TOGAF

(The Open Group

Architecture Framework)

Version 8.1

"Enterprise Edition"

Front matter

The Open Group Architectural Framework (TOGAF), Version 8.1
Document Number:

Any comments relating to the material contained in this document may be submitted to The Open Group using a feedback form, by electronic mail to OGArchFram@opengroup.org, or by post to:

The Open Group
44 Montgomery Street,
Suite 960
San Francisco, CA 94104
USA

or

The Open Group
Apex Plaza, Forbury
Road
Reading, RG1 1AX
UK

The Open Group

Overview

The Open Group is the leading vendor-neutral, international consortium for buyers and suppliers of technology. Its mission is to cause the development of a viable global information infrastructure that is ubiquitous, trusted, reliable, and easy-to-use. The Open Group creates an environment where all elements involved in technology development can cooperate to deliver less costly and more flexible IT solutions.

Formed in 1996 by the merger of the X/Open Company Ltd. (founded in 1984) and the Open Software Foundation (founded in 1988), The Open Group is supported by most of the world's largest user organizations, information systems vendors, and software suppliers. By combining the strengths of open systems specifications and a proven branding scheme with collaborative technology development and advanced research, The Open Group is well positioned to meet its new mission, as well as to assist user organizations, vendors, and suppliers in the development and implementation of products supporting the adoption and proliferation of systems which conform to standard specifications.

With more than 200 member companies, The Open Group helps the IT industry to advance technologically while managing the change caused by innovation. It does this by:

- Consolidating, prioritizing, and communicating customer requirements to vendors
- Conducting research and development with industry, academia, and government agencies to deliver innovation and economy through projects associated with its Research Institute
- Managing cost-effective development efforts that accelerate consistent multi-vendor deployment of technology in response to customer requirements
- Adopting, integrating, and publishing industry standard specifications that provide an essential set of blueprints for building open information systems and integrating new technology as it becomes available
- Licensing and promoting the Open Brand, represented by the "X" Device, that designates vendor products which conform to Open Group Product Standards
- Promoting the benefits of open systems to customers, vendors, and the public.

The Open Group operates in all phases of the open systems technology lifecycle including innovation, market adoption, product development, and proliferation. Presently, it focuses on seven strategic areas: open systems application platform development, architecture, distributed systems management, interoperability, distributed computing environment, security, and the information superhighway. The Open Group is also responsible for the management of the UNIX trademark on behalf of the industry.

Ordering Information

Full catalog and ordering information on all Open Group publications is available on the World-Wide Web at http://www.opengroup.org/pubs.

Go to Contents

Contents

PART I: Introduction

Preface

Welcome Structure Navigation Downloads

Welcome to TOGAF - The Open Group Architectural Framework.

TOGAF is a framework - a detailed method and a set of supporting tools - for developing an enterprise architecture. It is described in a set of documentation published by The Open Group on its public web server, and may be used freely by any organization wishing to develop an enterprise architecture for use within that organization. (See specific conditions of use.)

TOGAF was developed by The Open Group's own members, working within the Architecture Forum. The original development of TOGAF Version 1 in 1995 was based on the Technical Architecture Framework for Information Management (TAFIM), developed by the US Department of Defense. The DoD gave The Open Group explicit permission and encouragement to create TOGAF by building on the TAFIM, which itself was the result of many years of development effort and many millions of dollars of U.S. government investment.

Starting from this sound foundation, the members of The Open Group's Architecture Forum have developed successive versions of TOGAF each year and published each one on The Open Group's public web site.

If you are new to the field of enterprise architecture and/or TOGAF, you may find it worthwhile to read the set of Frequently Asked Questions, where you will find answers to questions such as:

- What is an architectural framework?
- What kind of "architecture" are we talking about?
- How does my organization benefit from using TOGAF?

The Structure of the TOGAF Document

There are four main parts to the TOGAF document.

- **PART I: Introduction** (this Part) provides a high-level introduction to some of the key concepts behind enterprise architecture and in particular the TOGAF approach.
- **PART II: Architecture Development Method** is the core of TOGAF. It describes the *TOGAF Architecture Development Method* - a step-by-step approach to developing an enterprise architecture.
- **PART III: Enterprise Continuum** describes the TOGAF Enterprise Continuum, a virtual repository of architecture assets, which includes the TOGAF Foundation Architecture, and the Integrated Information Infrastructure Reference Model.
- **PART IV: Resources** comprises the TOGAF Resource Base - a set of tools and techniques available for use in applying TOGAF and the TOGAF ADM.

Navigation

The TOGAF document set is designed for use with frames. To navigate around the document:

- In the main Contents frame at the top of the page, click the relevant hyperlink (Part I, Part II, etc.) to load the Contents List for that Part of the TOGAF document into the Secondary Index frame in the left margin.
- Then click in that Contents List to load a page into this main frame.
- Where an entry in a Contents List has the prefix *(Index to)*, that hyperlink will load a sub-contents list into the Secondary Index frame.

Downloads

Downloads of the TOGAF documentation, including a printable PDF file, are available under license from the TOGAF information web site. The license is free to any organization wishing to use TOGAF entirely for internal purposes (for example, to develop an information system architecture for use within that organization).

TOGAF - Frequently Asked Questions

1. What is an "Enterprise"? . . .

A good definition of "enterprise" in this context is any collection of organizations that has a common set of goals and/or a single bottom line. In that sense, an enterprise can be a government agency, a whole corporation, a division of a corporation, a single department, or a chain of geographically distant organizations linked together by common ownership.

The term "enterprise" in the context of "enterprise architecture" can be used to denote both an entire enterprise, encompassing all of its information systems, and a specific domain within the enterprise. In both cases, the architecture crosses multiple systems, and multiple functional groups with the enterprise.

Confusion also arises from the evolving nature of the term "enterprise". An "extended enterprise" nowadays frequently includes partners, suppliers and customers. If the goal is to integrate an "extended enterprise", then the enterprise comprises the partners, suppliers and customers, as well as internal business units.

Large corporations and government agencies may comprise multiple "enterprises," and hence there may well be separate enterprise architecture projects. However, there is often much in common about the information systems in each "enterprise", and there is usually great potential for gain in the use of a common architecture framework. For example, a common framework can provide a basis for the development of an architecture repository for the integration and re-use of models, designs, and baseline data.

. . . an architecture? . . .

The definition of an **architecture** used in ANSI/IEEE Std 1471-2000 is: “the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution”.

At the present time TOGAF embraces but does not strictly adhere to ANSI/IEEE Std 1471-2000 terminology. In TOGAF, “Architecture” has two meanings depending upon its contextual usage:

1. A formal description of a system, or a detailed plan of the system at component level to guide its implementation.
2. The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time.

In TOGAF we endeavor to strike a balance between promoting the concepts and terminology of ANSI/IEEE Std 1471-2000 - ensuring that our usage of terms defined by ANSI/IEEE Std 1471-2000 is consistent with the standard - and retaining other commonly accepted terminology that is familiar to the majority of the TOGAF readership. *More on terminology...*

. . . an architecture description?

An **architecture description** is a formal description of an information system, organized in a way that supports reasoning about the structural properties of the system. It defines the components or building blocks that make up the overall information system, and provides a plan from which products can be procured, and systems developed, that will work

together to implement the overall system. It thus enables you to manage your overall IT investment in a way that meets the needs of your business.

. . . an architecture framework?

An **architecture framework** is a tool which can be used for developing a broad range of different architectures. It should describe a method for designing an information system in terms of a set of building blocks, and for showing how the building blocks fit together. It should contain a set of tools and provide a common vocabulary. It should also include a list of recommended standards and compliant products that can be used to implement the building blocks.

Back to Top

---

2. Why do I need an enterprise architecture?

The primary reason for developing an enterprise architecture is to support the business by providing the fundamental technology and process structure for an IT strategy. This in turn makes IT a responsive asset for a successful modern business strategy.

Today's CEOs know that the effective management and exploitation of information through IT is the key to business success, and the indispensable means to achieving competitive advantage. An enterprise architecture addresses this need, by providing a strategic context for the evolution of the IT system in response to the constantly changing needs of the business environment.

Furthermore, a good enterprise architecture enables you to achieve the right balance between IT efficiency and business innovation. It allows individual business units to innovate safely in their pursuit of competitive advantage. At the same time, it assures the needs of the organization for an integrated IT strategy, permitting the closest possible synergy across the extended enterprise.

The technical advantages that result from a good enterprise architecture bring important business benefits, which are clearly visible in the bottom line:

- A more efficient IT operation
  - Lower software development, support, and maintenance costs
  - Increased portability of applications
  - Improved interoperability and easier system and network management
  - A better ability to address critical enterprise-wide issues like security
  - Easier upgrade and exchange of system components
- Better return on existing investment, reduced risk for future investment
  - Reduced complexity in IT infrastructure
  - Maximum return on investment in existing IT infrastructure
  - The flexibility to make, buy, or out-source IT solutions
  - Reduced risk overall in new investment, and the costs of IT ownership
- Faster, simpler, and cheaper procurement
  - Buying decisions are simpler, because the information governing procurement is readily available in a coherent plan.
  - The procurement process is faster - maximizing procurement speed and flexibility without sacrificing architectural coherence.

Back to Top

---

3. Why do I need a "Framework" for enterprise architecture?

Using an architecture framework will speed up and simplify architecture development, ensure more complete coverage of the designed solution, and make certain that the architecture selected allows for future growth in response to the needs of the business.

Architecture design is a technically complex process, and the design of heterogeneous, multi-vendor architectures is

particularly complex. TOGAF plays an important role in helping to “demystify” the architecture development process, enabling IT users to build genuinely open systems-based solutions to their business needs.

Why is this important?

Those IT customers who do not invest in enterprise architecture typically find themselves pushed inexorably to single-supplier solutions in order to ensure an integrated solution. At that point, no matter how ostensibly “open” any single supplier’s products may be in terms of adherence to standards, the customer will be unable to realize the potential benefits of truly heterogeneous, multi-vendor open systems.

Back to Top

---

4. What specifically would prompt me to develop an architecture?

Typically, an architecture is developed because key people have concerns that need to be addressed by the IT systems within the organization. Such people are commonly referred to as the *stakeholders* in the system. The role of the architect is to address these concerns, by identifying and refining the requirements that the stakeholders have, developing views of the architecture that show how the concerns and the requirements are going to be addressed, and by showing the trade-offs that are going to be made in reconciling the potentially conflicting concerns of different stakeholders.

Without the architecture, it is highly unlikely that all the concerns and requirements will be considered and met.

Back to Top

---

5. What is TOGAF?

TOGAF is an **architecture framework** - The Open Group Architecture Framework. It enables you to design, evaluate, and build the right architecture for your organization.

The key to TOGAF is the TOGAF Architecture Development Method (ADM) - a reliable, proven method for developing an ~~IT~~ enterprise architecture that meets the needs of your business.

Back to Top

---

6. What kind of "architecture" does TOGAF deal with?

There are four types of architecture that are commonly accepted as subsets of an overall Enterprise Architecture, all of which TOGAF is designed to support:

- A **business** (or **business process**) **architecture** - this defines the business strategy, governance, organisation, and key business processes.
- An **applications architecture** - this kind of architecture provides a blueprint for the individual application systems to be deployed, their interactions, and their relationships to the core business processes of the organization.
- A **data architecture** - this describes the structure of an organization's logical and physical data assets and data management resources.
- A **technology architecture** - this describes the software infrastructure intended to support the deployment of core, mission-critical applications. This type of software is sometimes referred to as "middleware".

Back to Top

---

7. Who would benefit from using TOGAF?

Any organization undertaking, or planning to undertake, the design and implementation of an enterprise architecture for the support of mission-critical business applications, using open systems building blocks.

Customers who design and implement enterprise architectures using TOGAF are ensured of a design and a procurement specification that will greatly facilitate open systems implementation, and will enable the benefits of open systems to accrue to their organizations with reduced risk.

Back to Top

---

8. What Specifically Does TOGAF Contain?

TOGAF provides a common sense, practical, prudent, and effective method of developing an enterprise architecture.

TOGAF consists of three main parts:

- The **TOGAF Architecture Development Method (ADM)**, which explains how to derive an organization-specific enterprise architecture that addresses business requirements. The ADM provides:
  - A reliable, proven way of developing the architecture
  - Architecture views which enable the architect to ensure that a complex set of requirements are adequately addressed
  - Linkages to practical case studies
  - Guidelines on tools for architecture development
- The **Enterprise Continuum**, which is a "virtual repository" of all the architecture assets - models, patterns, architecture descriptions, etc. - that exist both within the enterprise and in the IT industry at large, which the enterprise considers itself to have available for the development of architectures. At relevant places throughout the TOGAF ADM, there are reminders to consider which architecture assets from the Enterprise Continuum the architect should use, if any. TOGAF itself provides two reference models for consideration for inclusion in an enterprise's own Enterprise Continuum:
  - The **TOGAF Foundation Architecture** -- an architecture of generic services and functions that provides a foundation on which specific architectures and architectural building blocks can be built. This Foundation Architecture in turn includes:
    - the *TOGAF Technical Reference Model (TRM)*, which provides a model and taxonomy of generic platform services; and
    - The *TOGAF Standards Information Base (SIB)*, a database of open industry standards that can be used to define the particular services and other components of an enterprise-specific architecture
  - The **Integrated Information Infrastructure Reference Model**, which is based on the TOGAF Foundation Architecture, and is specifically aimed at helping the design of architectures that enable and support the vision of "Boundaryless Information Flow".
- The **TOGAF Resource Base**, which is a set of resources - guidelines, templates, background information, etc. - to help the architect in the use of the ADM.

Back to Top

---

9. Just how do you use TOGAF?

TOGAF is published by The Open Group on its public Web site, and may be reproduced freely by any enterprise wishing to use it to develop an enterprise architecture for use within that enterprise.

Basically, information about the benefits and constraints of the existing implementation, together with requirements for change, are combined using the methods described in the TOGAF ADM, resulting in a “target architecture” or set of target architectures.

The TOGAF Standards Information Base (SIB) provides a database of open industry standards that can be used to define the particular services and components required in the products purchased to implement the developed architecture. The SIB provides a simple and highly effective way to procure against an enterprise architecture.

Back to Top

10. How much Does TOGAF cost?

The Open Group operates as a not-for-profit consortium committed to delivering greater business efficiency by bringing together buyers and suppliers of information systems to lower the barriers of integrating new technology across the enterprise. Its goal is to realize the vision of "Boundaryless Information Flow".

TOGAF is a key part of its strategy for achieving this goal, and The Open Group wants TOGAF to be taken up and used in practical architecture projects, and the experience from its use fed back to help improve it.

The Open Group therefore publishes TOGAF on its public web server, and allows and encourages its reproduction and use free of charge by any organization wishing to use it internally to develop an enterprise architecture. (There are restrictions on its commercial exploitation, however: see specific conditions of use.)

Back to Top

11. Since TOGAF is freely available, why join The Open Group?

Organizations wishing to reduce the time, cost, and risk of implementing multi-vendor solutions that integrate within and between enterprises need The Open Group as their key partner.

The Open Group brings together the buyers and suppliers of information systems world-wide, and enables them to work together, both to ensure that information technology solutions meet the needs of customers, and to make it easier to integrate information technology across the enterprise.

The Open Group Architecture Framework is a key enabler in this task.

Yes, TOGAF itself is freely available. But how much will you spend on developing or updating your enterprise architecture using TOGAF? And how much will you spend on procurements based on that architecture?

The price of Open Group membership is insignificant in comparison with these amounts.

In addition to the general benefits of membership, as a member of The Open Group you will be eligible to participate in The Open Group Architecture Forum, which is the development program within which TOGAF is evolved, and in which TOGAF users come together to exchange information and feedback.

Members of the Architecture Forum gain:

- Immediate access to the fruits of the current year's TOGAF work program (not publicly available until publication of the next edition of the TOGAF document) - in effect, the latest information on TOGAF, as opposed to information that is up to a year old
- Exchange of experience with other customer and vendor organizations involved in enterprise architecture in general, and networking with architects using TOGAF in significant architecture development projects around the world.
- Peer review of specific architecture case study material

Back to Top

TOGAF as an Enterprise Architecture Framework

Introduction

The TOGAF FAQ explains that there are four kinds of "architecture" that are commonly accepted as subsets of an overall Enterprise Architecture:

- Business Architecture;
- Data Architecture;
- Application Architecture; and
- Technology Architecture.

The combination of Data Architecture and Application Architecture is also referred to as the *Information System Architecture*.

TOGAF was originally designed to support the last of these - the Technology Architecture. Over its years of evolution, however, it has acquired many of the facets of a framework and method for Enterprise Architecture. As of TOGAF Version 8, these different facets have been integrated, and TOGAF has undergone a major redevelopment, with the result that it is now a fully-fledged enterprise architecture framework.

With Version 8.1, the numbering scheme for successive releases of TOGAF has been modified, to include a major and minor release indicator. Version 8.1 builds on the base established in Version 8, by adding new information in a number of areas. See the Version 8.1 Release Notice for a summary of the changes in this Release of TOGAF.

The Role of TOGAF

TOGAF in its Enterprise edition remains what it has always been, namely an *architectural framework* - a set of methods and tools for developing a broad range of different IT architectures. It enables IT users to design, evaluate, and build the right architecture for their organization, and reduces the costs of planning, designing, and implementing architectures based on open systems solutions.

The key to TOGAF remains a reliable, practical method - the TOGAF Architecture Development Method (ADM) - for defining business needs and developing an architecture that meets those needs, utilizing the elements of TOGAF and other architectural assets available to the organization.

A number of enterprise architecture frameworks already exist and are widely recognized, each of which has its particular advantages and disadvantages, and relevance, for enterprise architecture. They are discussed in Part IV, *Other Architectures and Frameworks*.

Although a number of enterprise frameworks exist, there is no accepted industry standard *method* for developing an enterprise architecture. The Open Group's goal with TOGAF is to work towards making the TOGAF architecture development method just such an industry standard method, which is neutral towards tools and technologies, and can be used for developing the products associated with any recognized enterprise framework, such the Zachman Framework, FEAF, TEAF, C4ISR/DoD Framework, that the architect feels is appropriate for a particular architecture.

The TOGAF Architecture Development Method therefore does not *prescribe* any specific set of enterprise architecture deliverables - although it does *describe* a set, by way of example. Rather, TOGAF is designed to be used with whatever set of deliverables the TOGAF user feels is most appropriate. That may be the set of deliverables described in TOGAF itself; or it may be the set associated with another framework, such as Zachman Framework, FEAF, etc.

- In fact, TOGAF has always done this: it does not *prescribe* a specific set of "Architecture Views", but *describes* an example taxonomy of the kinds of views that an architect might consider developing, and why; and it provides

guidelines for making the choice, and for developing particular views, if chosen.

With the migration of TOGAF to an enterprise architecture framework, this flexibility becomes even more important. TOGAF is not intended to compete with these other frameworks: rather, it is intended to perform a unique role, in distilling what these other frameworks have to offer, and providing a generic Architecture Development Method that can be adapted for use with any of these other frameworks.

The Open Group's vision for TOGAF is as a vehicle and repository for practical, experience-based information on how to go about the process of enterprise architecture, providing a generic method with which specific sets of deliverables, specific reference models, and other relevant architectural assets, can be integrated.

TOGAF and Architecture Governance

As governance has become an increasingly visible requirement for organisational management, the adoption of governance into TOGAF aligns the framework with current business best practice and also ensures a level of visibility, guidance, and control that will support all architecture stakeholder requirements and obligations.

The benefits of Architectural Governance include:

- Increased transparency of accountability, and informed delegation of authority
- Controlled risk management
- Protection of existing asset base through maximizing re-use of existing architectural components
- Proactive control, monitoring and management mechanisms
- Process, concept, and component reuse across all organisational business units
- Value creation through monitoring, measuring, evaluation, and feedback
- Increased visibility supporting internal processes and external parties' requirements
    - In particular, increased visibility of decision-making at lower levels - ensuring oversight at an appropriate level within the enterprise of decisions that may have far-reaching strategic consequences for the organization
- Greater shareholder value
    - In particular, Enterprise Architecture increasingly represents the core Intellectual Property of the enterprise.
    - Studies have demonstrated a correlation between increased shareholder value and well governed enterprises.
- Integrates with existing processes and methodologies and complements functionality by adding control capabilities.

Further detail on Architecture Governance is given on Part IV of TOGAF.

Using TOGAF with Other Frameworks

Overview

Two of the key elements of any enterprise architecture framework are:

- a definition of the deliverables that the architecting activity should produce; and
- a description of the method by which this should be done.

With some exceptions, the majority of enterprise architecture frameworks focus on the first of these - the specific set of deliverables - and are relatively silent about the methods to be used to generate them (intentionally so, in some cases.)

Because TOGAF is a generic framework, as mentioned above, and intended to be used in a wide variety of environments, it does not prescribe a specific set of deliverables; rather it talks in general terms about the types of deliverable that need to be produced, and focuses instead on the methods by which these should be developed.

As a result, TOGAF may be used either in its own right, with the generic deliverables that it describes; or else these deliverables may be replaced by a more specific set, defined in any other framework that the user architect considers relevant.

In the latter case, the user architect will adapt and build on the TOGAF ADM in order to define a tailored method and process

for developing these more specific deliverables. Guidelines for adapting the TOGAF ADM in such a way are given under Adapting the ADM in Part II.

As a generic framework and method for Enterprise Architecture, TOGAF also complements other frameworks that are aimed at specific vertical business domains, specific horizontal technology areas such as Security or Manageability, or specific application areas such as e-Commerce. The concept of leveraging other relevant architectural assets in this way is known within TOGAF as the *Enterprise Architecture Continuum*.

The Enterprise Architecture Continuum

TOGAF embodies the concept of the *Enterprise Architecture Continuum* (described in Part III), to reflect different levels of abstraction in an architecture development process. In this way TOGAF facilitates understanding and co-operation between actors at different levels. It provides a context for the use of multiple frameworks, models, and architecture assets in conjunction with the TOGAF ADM. By means of the Enterprise Architecture Continuum, architects are encouraged to leverage all other relevant architectural resources and assets, in addition to the TOGAF Foundation Architecture, in developing an organization-specific IT architecture.

In this context, the TOGAF ADM can be regarded as describing the process of moving from the TOGAF Foundation Architecture to an organization-specific architecture (or set of architectures), leveraging the contents of the Enterprise Architecture Continuum along the way, including the TOGAF Foundation Architecture and other relevant architectural frameworks, models, components and building blocks.

Summary

TOGAF thus does not seek to compete with or duplicate other frameworks. What TOGAF does seek to provide is a practical, industry standard method of doing enterprise architecture, leveraging all relevant assets in the process, that is freely available and supported by a number of different architecting consultancies, and that is sufficient for an organization to use "as-is" or to adapt as the basis of an enterprise architecture method for other, deliverables-focused frameworks.

PART II: Architecture Development Method (ADM)

Introduction to the Architecture Development Method (ADM)

ADM Overview

The TOGAF Architecture Development Method (ADM) is the result of continuous contributions from a large number of architecture practitioners. It describes a method for developing an enterprise architecture, and forms the core of TOGAF. It integrates elements of TOGAF as well as other available architectural assets, to meet the business and information technology needs of an organization.

The TOGAF Architecture Development Method (ADM) forms the core of TOGAF. It is a method for developing an enterprise architecture to meet the business and information technology needs of an organization, utilizing the other elements of TOGAF described in this document, and other architectural assets available to the organization.

Relationship to Other Parts of TOGAF

There are two other main parts to TOGAF, besides the ADM:

- The **TOGAF Enterprise Continuum**, described in detail in Part III. This is a 'framework-within-a-framework' that provides context for the leveraging of relevant architecture assets and provides navigational help when discussions move between different levels of abstraction.

- The **TOGAF Resource Base**, described in Part IV. This is a set of resources - guidelines, templates, checklists, and other detailed materials supporting the TOGAF Architecture Development Method.

The ADM and the Enterprise Continuum

As mentioned above, the TOGAF Enterprise Continuum provides a framework and context for the leveraging of relevant architecture assets in executing the ADM. These assets may include architecture descriptions, models and patterns taken from a variety of sources, as explained in Part III. At relevant places throughout the ADM, there are reminders to consider which architecture assets from the Enterprise Continuum the architect should use, if any. In some cases, for example in the development of a Technology Architecture, this may be TOGAF's own Foundation Architecture (see Part III). In other cases, for example in the development of a business architecture, it may be a reference model for e-Commerce taken from the industry at large.

The practical implementation of the Enterprise Continuum will often take the form of a repository that includes reference architectures, models and patterns that have been accepted for use within the enterprise, and actual architectural work done previously within the enterprise. The architect would seek to reuse as much as possible from the Continuum that was relevant to the project at hand. (In addition to the collection of architecture source material, the repository would also contain architecture development work-in-progress.)

The criteria for including source materials in an organization's Enterprise Continuum will typically form part of the organization's IT governance process.

The Enterprise Continuum is thus a framework (a "framework-within-a-framework") for categorizing architectural source material – both the contents of the architecture working repository, and the set of relevant, available reference models in the industry.

In executing the ADM, the architect is not only developing the end result of an organization-specific architecture, s/he is also populating the organization's own Enterprise Architecture Continuum, with all the architectural assets identified and leveraged along the way -- including, but not limited to, the resultant enterprise-specific architecture.

Architecture development is an iterative, on-going process, and in executing the ADM repeatedly over time, the architect gradually populates more and more of the organization's Enterprise Continuum. Although the primary focus of the ADM is on the development of the enterprise-specific architecture, in this wider context the ADM can also be viewed as the process of

populating the enterprise's own Enterprise Continuum with relevant reusable building blocks.

In fact, the first execution of the ADM will often be the hardest, since the architecture assets available for re-use will be relatively few. Even at this stage of development, however, there will be architecture assets available from external sources such as TOGAF, as well as the IT industry at large, that could be leveraged in support of the effort.

Subsequent executions will be easier, as more and more architecture assets become identified; are used to populate the organization's Enterprise Continuum; and are thus available for future re-use.

The ADM is also useful to populate the Foundation Architecture of an enterprise. Business requirements of an enterprise may be used to identify the necessary definitions and selections in the Foundation Architecture. This could be a set of re-usable common models, policy and governance definitions, or even as specific as overriding technology selections (e.g. if mandated by law). Population of the Foundation Architecture follows similar principles as for an Enterprise Architecture, with the difference that requirements for a whole enterprise are restricted to the overall concerns and thus less complete than for a specific enterprise.

It is important to recognize that existing models from these various sources may not necessarily be integratable into a coherent enterprise architecture. "Integratability" of architecture descriptions is considered below, under Architecture Integration.

The ADM and the Resource Base

The TOGAF Resource Base is a set of resources - guidelines, templates, checklists, and other detailed materials - that support the TOGAF Architecture Development Method.

The individual sections of the Resource Base are described separately in Part IV so that they can be referenced from the relevant points in the ADM as necessary, rather than having the detailed text clutter the description of the ADM itself.

The Architecture Development Cycle

Key Points

The following are the key points about the ADM:

- The ADM is iterative, over the whole process, between phases, and within phases. For each iteration of the ADM, a fresh decision must be taken as to:
  - the breadth of coverage of the enterprise to be defined
  - the level of detail to be defined
  - the extent of the time horizon aimed at, including the number and extent of any intermediate time horizons
  - the architectural assets to be leveraged in the organization's Enterprise Continuum, including:
    - assets created in previous iterations of the ADM cycle within the enterprise
    - assets available elsewhere in the industry (other frameworks, systems models, vertical industry models, etc.)
- These decisions need to be made on the basis of a practical assessment of resource and competence availability, and the value that can realistically be expected to accrue to the enterprise from the chosen scope of the architecture work.
- As a generic method, the ADM is intended to be used by enterprises in a wide variety of different geographies and applied in different vertical sectors / industry types. As such, it may be, but does not necessarily have to be, tailored to specific needs. For example,
  - it may be used in conjunction with the set of deliverables of another framework, where these have been deemed to be more appropriate for a specific organization. (For example, many US federal agencies have developed individual frameworks that define the deliverables specific to their particular departmental needs.)
  - it may be used in conjunction with the well known Zachman Framework, which is an excellent classification scheme, but lacks an openly available, well defined methodology.

These issues are considered in detail under Adapting the ADM below.

In addition to the method itself being iterative, there is also iteration within the ADM cycle, both among the individual phases and among the steps within each phase.

Basic Structure

The basic structure of the ADM is shown in Figure 1.

Throughout the ADM cycle, there needs to be frequent validation of results against the original expectations, both those for the whole ADM cycle, and those for the particular phase of the process.

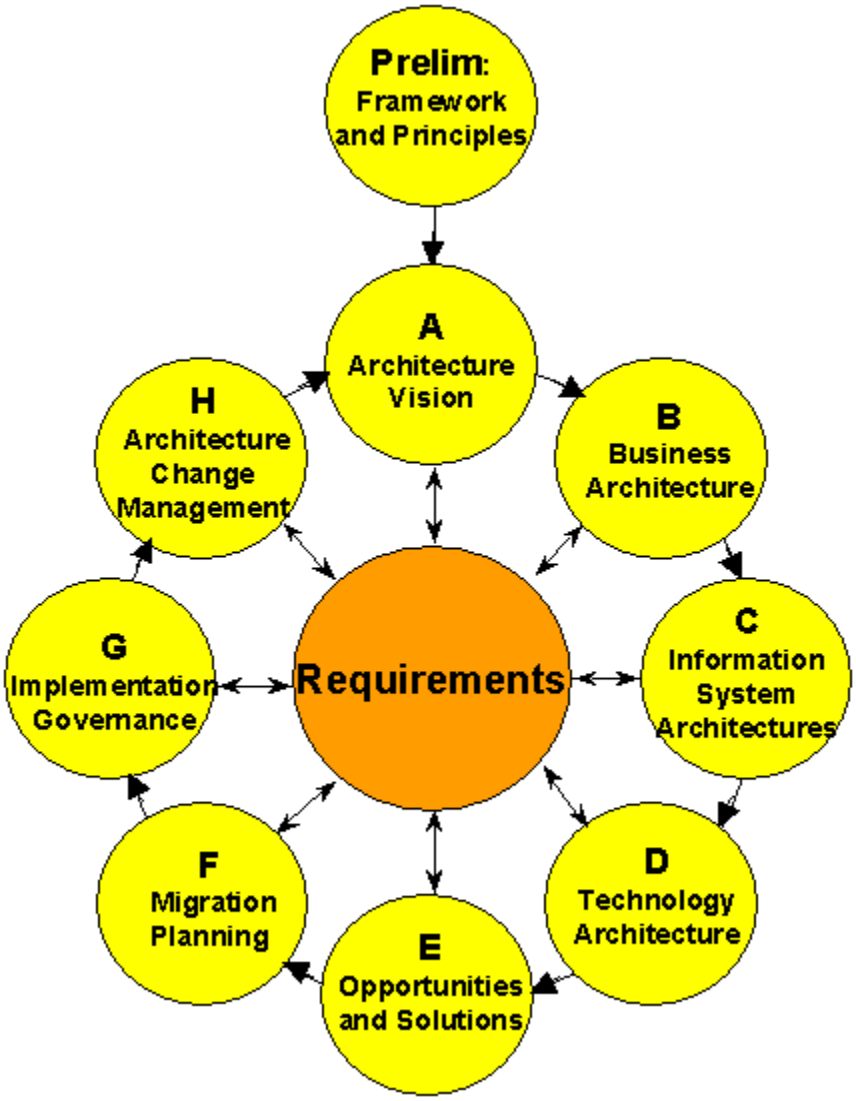


Figure 1: Architecture Development Cycle

The phases of the ADM cycle shown in Figure 1 are further divided into steps, such as the ones depicted by the expansion of the Technology Architecture phase in Figure 2.

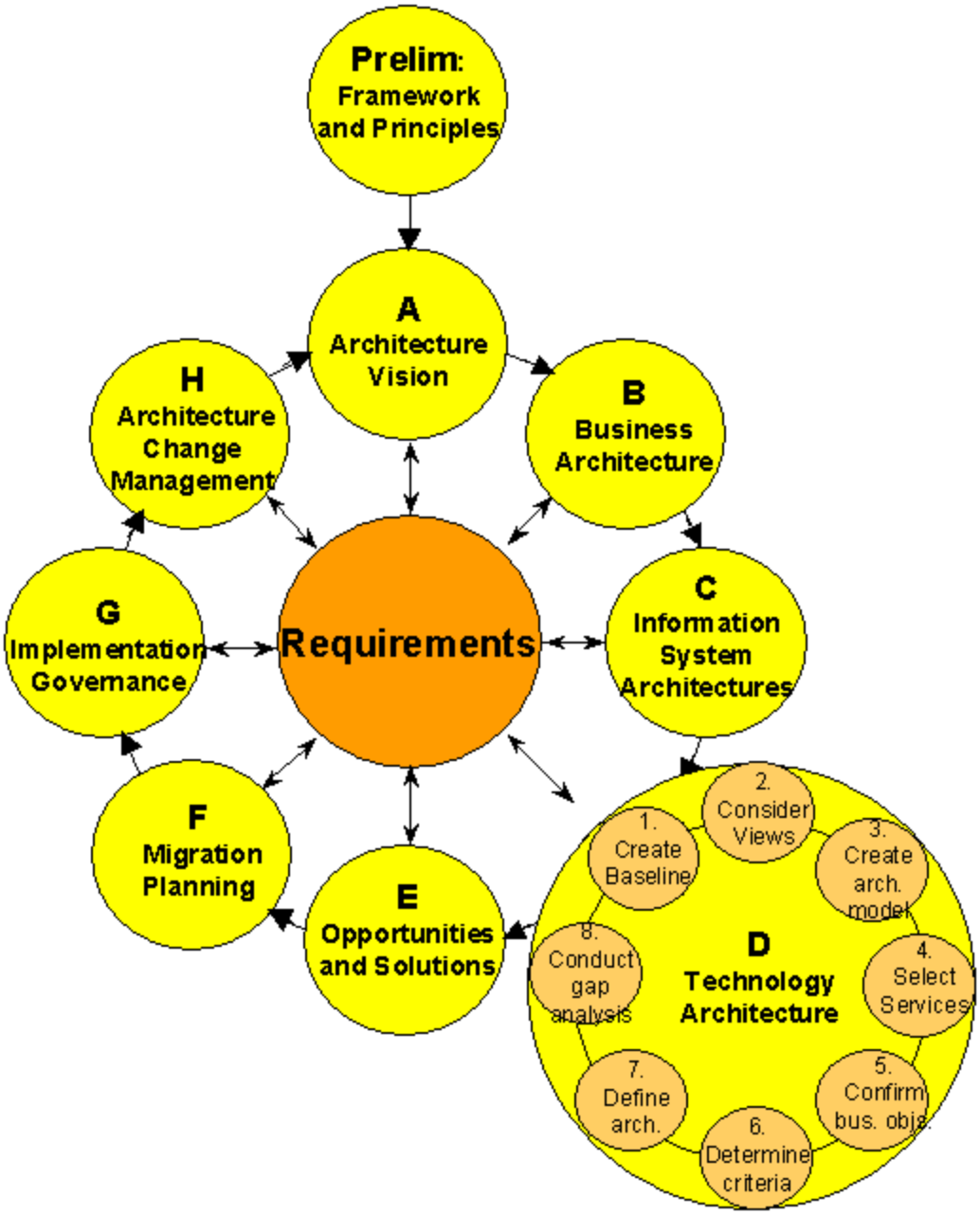


Figure 2: Architecture Development Cycle - Expansion

The phases of the cycle are described in detail in the following subsections. Phase D, the creation of a Technology Architecture, is described in even greater detail in a separate section, Target Architecture - Detailed Description.

Note that output is generated throughout the process, and that the output in an early phase may be modified in a later phase. The versioning of output is managed through version numbers.

Adapting the ADM

The ADM is a generic method for architecture development, which is designed to deal with most system and organizational requirements. However, it will often be necessary to modify or extend the ADM, to suit specific needs. One of the tasks before applying the ADM is to review its components for applicability, and then tailor them as appropriate to the circumstances of the individual enterprise. This activity may well produce an 'enterprise-specific' ADM.

One reason for wanting to adapt the ADM, which it is important to stress, is that the order of the phases in the ADM is to some extent dependent on the maturity of the architecture discipline within the enterprise concerned. For example, if the

business case for doing architecture at all is not well recognized, then creating an Architecture Vision is almost always essential; and a detailed Business Architecture often needs to come next, in order to underpin the Architecture Vision, detail the business case for remaining architecture work, and secure the active participation of key stakeholders in that work. In other cases a slightly different order may be preferred: for example, a detailed inventory of the baseline environment may be done before undertaking the Business Architecture.

The order of phases may also be defined by the business and architectural principles of an enterprise. For example, the business principles may dictate that the enterprise be prepared to adjust its business processes to meet the needs of a packaged solution, so that it can be implemented quickly to enable fast response to market changes. In such a case, the Business Architecture (or at least the completion of it) may well follow completion of the Information Systems Architecture or the Technology Architecture.

Another reason for wanting to adapt the ADM is if TOGAF is to be integrated with another enterprise framework (as explained in Part I, *TOGAF in the Enterprise*). For example, an enterprise may wish to use TOGAF and its generic ADM in conjunction with the well-known Zachman Framework, or another enterprise architecture framework that has a defined set of deliverables specific to a particular vertical sector: Government, Defense, e-Business, Telecommunications, etc. The ADM has been specifically designed with this potential integration in mind.

Other possible reasons for wanting to adapt the ADM include:

- The ADM is one of the many corporate processes that make up the corporate governance model. It is complementary to, and supportive of, other standard program management processes, such as those for: authorization; risk management; business planning and budgeting; development planning; systems development; and procurement.
- The ADM is to be used as a method for something other than enterprise architecture: for example, as a general program management method.
- The ADM is being mandated for use by a prime or lead contractor in an outsourcing situation, and needs to be tailored to achieve a suitable compromise between the contractor's existing practises and the contracting enterprise's requirements.
- The enterprise is a Small-to-Medium Enterprise, and wishes to use a "cut-down" method more attuned to the reduced level of resources and system complexity typical of such an environment.
- The enterprise is very large and complex, comprising many separate but interlinked "enterprises" within an overall collaborative business framework, and the architecture method needs to be adapted to recognize this. Different approaches to planning and integration may be used in such cases, including the following (possibly in combination):
    - *Top-down planning and development* - designing the whole interconnected meta-enterprise as a single entity (an exercise that typically stretches the limits of practicality)
    - *Development of a "generic" or "reference" architecture*, typical of the enterprises within the organization, but not representing any specific enterprise, which individual enterprises are then expected to adapt in order to produce an architecture "instance" suited to the particular enterprise concerned.
    - *Replication* - developing a specific architecture for one enterprise, implementing it as a proof of concept, and then taking that as a "reference architecture" to be cloned in other enterprises.
- In a vendor or production environment, a generic architecture for a family of related products is often referred to as a Product Line Architecture, and the analogous process to that outlined above is termed *(Architecture-based) Product Line Engineering*. The ADM is targeted primarily at architects in IT user enterprises, but a vendor organization whose products are IT-based might well wish to adapt it as a generic method for a product line architecture development.

Note: Readers interested in the concept of infrastructure architecture, and/or its use to support acquisitions and mergers, consolidation, etc., are referred to Part IV, Infrastructure Architecture, where an approach to infrastructure architecture using the TOGAF ADM is outlined.

Architecture Governance

The ADM, whether adapted by the organisation or used as documented here, is a key process to be managed in the same manner as other architecture artefacts in the Enterprise Continuum. The Architecture Board should be satisfied that the method is being applied correctly across all phases of an architecture development iteration. Compliance with the ADM is

fundamental to the governance of the architecture, to ensure that all considerations are made and all required deliverables are produced.

Process Management

The management of all architectural artefacts, governance, and related processes should be supported by a managed environment. Typically this would be based on one or more repositories supporting versioned object and process control and status.

Governance process management includes repository management, access, communication, training, and accreditation. This section is included to identify the major information areas managed by the governance repository. The repository initially consists of one or more data storage facilities that will contain the following types of information:

- Reference data (collateral from organization's own repositories / Enterprise Continuum, including external data: e.g. COBIT, ITIL)

  Used for guidance and instruction during project implementation. This includes the details of information outlined above. The reference data includes description of the governance procedures themselves.
- Process status

  All information regarding the state of any governance processes will be managed - examples of this include outstanding compliance requests, dispensation requests, and compliance assessments investigations.
- Audit information

  This will record all completed governance process actions and will be used to support:
  a. Key decisions and responsible personnel for any architecture project that has been sanctioned by the governance process.
  b. A reference for future architectural and supporting process developments, guidance and precedence.

The Governance artefacts and process are themselves part of the contents of the Enterprise Continuum.

Scoping the Architecture Activity

There are many reasons for wanting to limit the scope of the architecture activity to be undertaken, most of which come down to the availability of people, finance, and other resources. The scope chosen for the architecture activity is normally directly dependent on available resources, and in the final analysis is usually a question of feasibility.

Whatever the reasons for wanting or having to limit the scope of the architecture activity, there are four dimensions in which the scope may be defined and limited:

- **Enterprise scope or focus**: What is the full extent of the "enterprise", and how much of that extent should the architecting effort focus on?
  - Many enterprises are very large, effectively comprising a federation of organizational units that could validly be considered enterprises in their own right.
  - The modern "enterprise" increasingly extends beyond its traditional boundaries, to embrace a fuzzy combination of traditional business enterprise combined with suppliers, customers, and partners.
- **Architecture domains**: A complete enterprise architecture description should contain all four architecture domains (business, data, applications, infrastructure), but the realities of resource and time constraints often mean there is not enough time, funding, or resources to build a top-down, all-inclusive architecture description encompassing all four architecture domains, even if the enterprise scope is chosen to be less than the full extent of the overall enterprise.
- **Vertical scope, or level of detail**: To what level of detail should the architecting effort go? How much architecture is "enough"? What is the appropriate demarcation between the architecture effort and other, related activities (System Design, System Engineering, System Development)?
- **Time horizon**: What is the time horizon that needs to be articulated for the architectural vision, and does it make sense (in terms of practicality and resources) for the same horizon to be covered in the detailed architecture description? If not, how many intermediate target architectures are to be defined, and what are their time horizons?

These aspects are explored in detail below. In each case, particularly in large-scale environments where architectures are

necessarily developed in a federated manner, there is a danger of architects optimizing within their own scope of activity, instead of at the level of the overall enterprise. It is often necessary to sub-optimize in a particular area, in order to optimize at the enterprise level. The aim should always be to seek the highest level of commonality and focus on scalable and reusable modules in order to maximize re-use at the enterprise level.

Enterprise Scope / Focus

One of the key decisions is the focus of the architecture effort, in terms of the breadth of overall enterprise activity to be covered (which specific business sectors, functions, organizations, geographical areas, etc.).

One important factor in this context is the increasing tendency for large-scale architecture developments to be undertaken in the form of "federated architectures" - independently developed, maintained and managed architectures that are subsequently integrated within a meta architectural framework. Such a framework specifies the principles for interoperability, migration, and conformance. This allows specific business units to have architectures developed and governed as stand-alone architecture projects.

Complex architectures are extremely hard to manage, not only in terms of the architecture development process itself, but also in terms of getting buy-in from large numbers of stakeholders. This in turn requires a very disciplined approach to identifying common architectural components, and management of the commonalties between federated components – deciding how to integrate, what to integrate, etc.

There are two basic approaches to federated architecture development:

- The overall enterprise is divided up "vertically", into enterprise "segments", each representing an independent business sector within the overall enterprise, and each having its own enterprise architecture with potentially all four architecture domains (business, data, applications, infrastructure). These separate, multi-domain architectures can be developed with a view to subsequent integration, but they can also be implemented in their own right, possibly with interim target environments defined, and therefore represent value to the enterprise in their own right.

- The overall enterprise architecture is divided up "horizontally", into architectural "super-domains", in which each architecture domain (business, data, applications, infrastructure) covering the full extent of the overall enterprise is developed and approved as a major project independently of the others, possible by different personnel. For example, a business architecture for the complete overall enterprise would form one independent architecture project, and the other domains would be developed and approved in separate projects, with a view to subsequent integration.

The US government, and in particular the US Department of Defense, has undertaken and published leading work in the field of federated architectures, emphasizing the need for integrated repositories and metamodels to aid integration and ensure interoperability. This work is very much at the leading edge of the state of the art, however, and what works in practice is still very much a matter of debate.

The Introduction section in the *Federal Enterprise Architecture Framework* published by the US Federal CIO Council explains the choices in approach that faced the US government in the development of the Federal Enterprise Architecture Framework (FEAF):

> *"In developing the Federal Enterprise Architecture Framework, the CIO Council evaluated three approaches.*
>
> - ***Conventional Approach*** *- Requires a substantial initial investment in time and dollars. First, a framework must be developed that shows how to prepare an architecture description. Second, the current baseline must be described. Finally, a target architecture must be described. Only after these activities are completed, implementing needed architecture changes through design, development, and acquisition of systems can begin. Although this approach appears to be sound, it may result in "paralysis by analysis," because of the complexity of the Federal effort.*
> - ***Segment Approach*** *- Promotes the incremental development of architecture segments within a structured enterprise architecture framework. This approach focuses on major business areas (e.g., grants or common financial systems) and is more likely to succeed because the effort is limited to common functions or specific enterprises.*
> - ***Status Quo Approach*** *- Represents business as usual resulting in continued failure to share information and cope with the rapidly changing environment. This approach would result in business rework, decreased productivity, and lost and missed opportunities, as well as failure to comply with Clinger-Cohen Act requirements.*
>
> *.....To mitigate the risk of overreaching with minimal returns, curtail startup costs for a conventional architecture, and realize returns quickly, the CIO Council selected the segment approach.....*

The Federal Enterprise Architecture Framework allows critical parts of the overall Federal Enterprise, called architectural segments, to be developed individually, while integrating these segments into the larger Enterprise Architecture."

The FEAF approach thus seeks to do a "complete" enterprise architecture across a succession of selected individual business domains (or segments), and then to integrate these into a more comprehensive, overarching enterprise architecture.

Conversely, the "Practical Guide to Federal Architecture", also issued by the US Federal CIO Council, highlights the dangers of selecting too narrow an enterprise scope, particularly at the higher business levels:

> *It is critically important that EA development be approached in a top-down, incremental manner, consistent with the hierarchical architectural views that are the building blocks of proven EA frameworks.... In doing so, it is equally important that the scope of the higher level business views of the EA span the entire enterprise or agency. By developing this enterprise-wide understanding of business processes and rules, and information needs, flows, and locations, the agency will be positioned to make good decisions about whether the enterprise, and thus the EA, can be appropriately compartmentalized. Without doing so, scoping decisions about the EA run the risk of promoting "stove-piped" operations and systems environments, and ultimately sub-optimizing enterprise performance and accountability.* "A Practical Guide to Federal Architecture", Chief Information Officer Council, Version 1.0, February 2001.

Current experience does seem to indicate that, in order to cope with the increasingly broad focus and ubiquity of architectures, it is often necessary to have a number of different architectures existing across an enterprise, focused on particular time frames, business functions, or business requirements; and this phenomenon would seem to call into question the feasibility of a single enterprise-wide architecture for every business function or purpose. In such cases, the paramount need is to manage and exploit the 'federations' of architecture. A good starting point is to adopt a publish-and-subscribe model that allows Architecture to be brought under a governance framework. In such a model, architecture developers and architecture consumers in projects (the supply and demand sides of Architecture work) sign up to a mutually beneficial framework of governance that ensures that:

1. architectural material is of good quality, up to date, fit for purpose, and published (reviewed and agreed to be made public).

2. usage of architecture material can be monitored, and compliance with standards, models, and principles can be exhibited, via

i. a compliance assessment process that describes what the user is subscribing to, and assesses their level of compliance; and
ii. a dispensation process that may grant dispensations from adherence to Architecture standards and guidelines in specific cases (usually with a strong business imperative).

Publish and subscribe techniques are being developed as part of general IT governance and specifically for the Defense sphere."

Architecture Domains

A complete enterprise architecture should address all four architecture domains (business, data, applications, technology), but the realities of resource and time constraints often mean there is not enough time, funding, or resources to build a top-down, all-inclusive architecture description encompassing all four architecture domains.

Architecture descriptions will normally be built with a specific purpose in mind - a specific set of business drivers that drive the architecture development - and clarifying the specific issue(s) that the architecture description is intended to help explore, and the questions it is expected to help answer, is an important part of the initial phase of the ADM.

For example, if the purpose of a particular architecture effort is to define and examine technology options for achieving a particular capability, and the fundamental business processes are not open to modification, then a full business architecture well may not be warranted. However, because the data, applications and infrastructure architectures build on the business architecture, the business architecture still needs to be thought through and understood.

While circumstances may sometimes dictate building an architecture description not containing all four architecture domains, it should be understood that such an architecture can not, by definition, be a complete enterprise architecture. One of the risks is lack of consistency and therefore ability to integrate. Integration either needs to come later, with its own costs and risks; or the risks and trade-offs involved in not developing a complete and integrated architecture need to be articulated by the architect, and communicated to and understood by the enterprise management.

Vertical Scope / Level of Detail

Care should be taken to judge the appropriate level of detail to be captured, based on the intended use of the Enterprise Architecture and the decisions to be made based on it. It is important that a consistent and equal level of depth be completed in each architecture domain (business, data, applications, infrastructure) included in the architecture effort. If pertinent detail is omitted, the architecture may not be useful. If unnecessary detail is included, the architecture effort may exceed the time and resources available, and/or the resultant architecture may be confusing or cluttered.

It is also important to predict the future uses of the architecture so that, within resource limitations, the architecture can be structured to accommodate future tailoring, extension, or reuse. The depth and detail of the Enterprise Architecture needs to be sufficient for its purpose, and no more.

John Zachman advocates developing enterprise-wide architecture at an enormous level of detail, in the same way as an aerospace company needs blueprints for everything down to nuts and bolts. Some regard this is an extreme position in terms of vertical scope, but it can certainly be justified when compared with the lifetime costs of alternatives. Zachman's argument is that information systems are not special. In other industries where very expensive, complex things are built, and where there is an expectation of repair or change, models are kept at an enormous level of detail, with concurrent expense. Aeroplanes, buildings, and cars are built this way. Why are information systems different?

However, it is not necessary to aim to complete a detailed architecture description at the first attempt. Future iterations of the Architecture Development Method, in a further architecture life-cycle, will build on the artefacts and the competencies created during the current iteration.

The bottom line is: there is a need to document all the models in an enterprise, to whatever level of detail is affordable, within an assessment of the likelihood of change and the concomitant risk, and bearing in mind the need to integrate the components of the different architecture domains (business / data / applications / infrastructure). The key is to understand the status of the enterprise's architecture work, and what can realistically be achieved with the resources and competencies available, and then focus on identifying and delivering the value that is achievable. Stakeholder value is a key focus: too broad a scope may deter some stakeholders (no RoI).

Time Horizon

The ADM is described in terms of a single cycle of architecture vision, and a set of target architectures (business, data, applications, technology) that enable the implementation of the vision.

However, when the enterprise scope is large, and/or the target architectures particularly complex, the development of target architecture descriptions may encounter major difficulties, or indeed prove "mission impossible", especially if being undertaken from scratch.

In such cases, a wider view may be taken, whereby an enterprise is represented by several different architecture instances, each representing the enterprise at a particular point in time. One architecture instance will represent the current enterprise state (the "as-is", or baseline). Another architecture instance, perhaps defined only partially, will represent the ultimate target end-state (the "vision"). In-between, intermediate or "transitional" architecture instances may be defined, each comprising its own set of target architecture descriptions.

By this approach, the target architecture work is split into two or more discrete stages:

1. First develop **Target Architecture** descriptions for the overall (large-scale) system, demonstrating a response to stakeholder objectives and concerns for a relatively distant timeframe (for example, a six-year period)
2. Then develop one or more "**Transitional Architecture**" descriptions, as increments or plateaus, each in line with and converging on the Target Architecture descriptions, and describing the specificities of the increment concerned.

In such an approach, the Target Architectures are evolutionary in nature, and require periodic review and update according to evolving business requirements and developments in technology, whereas the Transitional Architectures are (by design) incremental in nature, and in principle should not evolve during the implementation phase of the increment, in order to avoid the "moving target" syndrome. This of course is only possible if the implementation schedule is under tight control and relatively short (typically less than two years).

The Target Architectures remain relatively generic, and because of that are less vulnerable than the Transitional Architectures to obsolescence. They embody only the key strategic architectural decisions, which should be blessed by the stakeholders from the outset, whereas the detailed architectural decisions in the Transitional Architectures are deliberately postponed as far as possible (i.e. just before implementation) in order to improve responsiveness vis-à-vis new technologies and products.

The enterprise evolves by migrating to each of these Transitional Architectures in turn. As each Transitional Architecture is implemented, the enterprise achieves a consistent, operational state on the way to the ultimate vision. However, this vision itself is periodically updated to reflect changes in the business and technology environment, and in effect may never actually be achieved, as originally described. The whole process continues for as long as the enterprise exists and continues to change.

Such a breakdown of the architecture description into a family of related architecture products of course requires effective management of the set and their relationships.

Architecture Integration

There is a need to provide an integration framework that sits above the individual architectures. This can be an 'enterprise framework' such as Zachman to position the various domains and artefacts, or it may be a meta-architectural framework (i.e. principles, models and standards) to allow interoperability, migration, and conformance between federated architectures. The purpose of this meta-architectural framework is to: 1) allow the architect to understand how components fit into the framework; 2) derive the architectural models that focus on enterprise level capabilities; 3) define the conformance standards that enable the integration of components for maximum leverage and reuse.

As described above, a significant number of scoping decisions need to be taken, in terms of enterprise focus, architecture scope, level of detail, time horizons, and choice of transitional architectures, any one of which may result in a less than complete architecture description being developed. A potential way of assessing the gaps in scope or level of detail is to use an enterprise architecture framework (e.g. Zachman) to understand the coverage of the artefacts.

There are varying degrees of architecture description “integratability.” At the low end, integratability means that different architecture descriptions (whether prepared by one organizational unit or many) should have a “look and feel” that is sufficiently similar to enable critical relationships between the descriptions to be identified, thereby at least indicating the need for further investigation. At the high end, integratability ideally means that different descriptions should be capable of being combined into a single logical and physical representation.

At the present time the state of the art is such that architecture integration can be accomplished only at the lower end of the integratability spectrum. Key factors to consider are the granularity and level of detail in each artefact, and the maturity of standards for the interchange of architectural descriptions.

As organizations address common themes (such as service oriented architecture, and integrated information infrastructure), and universal data models and standard data structures emerge, integration toward the high end of the spectrum will be facilitated. However, there will always be the need for effective standards governance to reduce the need for manual co-ordination and conflict resolution.

Summary

The TOGAF ADM defines a recommended sequence for the various phases and steps involved in developing an architecture, but it cannot recommend a scope: this has to be determined by the organization itself, bearing in mind that the recommended sequence of development in the ADM process is an iterative one, with the depth and breadth of scope and deliverables increasing with each iteration. Each iteration will add resources to the organization's Architecture Continuum.

The choice of scope is critical to the success of the architecting effort. The key factor here is the sheer complexity of a complete, horizontally and vertically integrated enterprise architecture, as represented by a fully populated instantiation of the Zachman Framework. Very few enterprise architecture developments today actually undertake such an effort in a single development project, simply because it is widely recognized to be at the limits of the state of the art, a fact that John Zachman himself recognizes:

> *"Some day, you are going to wish you had all these models ... However, I am not so altruistic to think that we have to have them all today ... or even that we understand how to build and manage them all today. But the very fact that we can identify conceptually where we want to get SOME day, makes us think more about what we are doing in the current time frame that might prevent us from getting to where we want to go in the future."*
> John Zachman, in email correspondence to George Brundage.

John Zachman himself likes to point out the alternatives available to those who can't countenance the amount of work implied

in developing the all models required in his framework. There are only three choices: trial and error ("knocking down the walls"); starting from new; or reverse engineering the architecture from the existing systems; all of which are risky and/or hugely expensive. What is necessary due to the pace of change is to have a set of readily deployable artifacts and a process for assembling them swiftly.

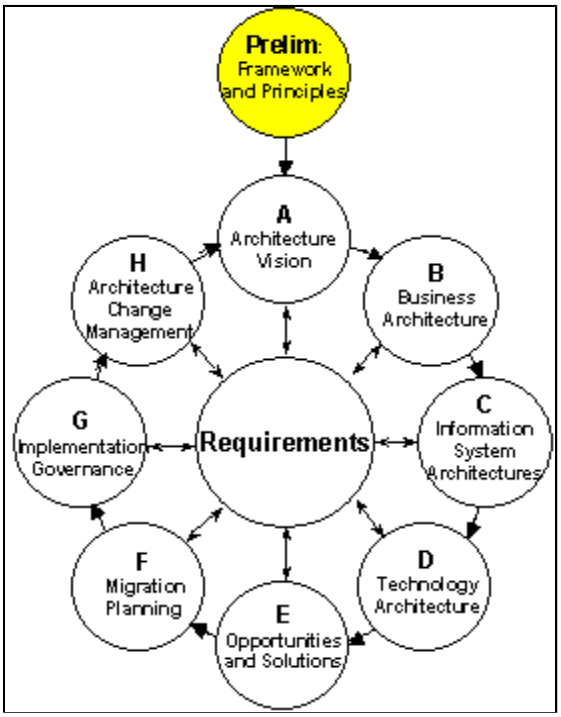
While such a complete framework is useful (indeed, essential) to have in mind as the ultimate long-term goal, in practice there is a key decision to be made as to the scope of a specific enterprise architecture effort. This being the case, it is vital to understand the basis on which scoping decisions are being made, and to set expectations aright for what is the goal of the effort.

The main guideline is to focus on what creates value to the enterprise, and to select horizontal and vertical scope, and time horizons, accordingly. Whether or not this is the first time around, understand that this exercise will be repeated, and that future iterations will build on what is being created in the current effort, adding greater width and depth.

Preliminary Phase: Framework and Principles

Objectives Approach Inputs Steps Outputs

Objectives



- To ensure that everyone who will be involved in, or benefit from this approach, is committed to the success of the architectural process.
- To define the architecture principles that will inform the constraints on any architecture work.
- To define the "architecture footprint" for the organization - the people reponsible for performing architecture work, where they are located, and their responsibilities.
- To define the scope and assumptions (particularly in a federated architecture environment)
- To define the framework and detailed methodologies that are going to be used to develop enterprise architectures in the organization concerned (typically, an adaptation of the generic ADM).
- To set up and monitor a process (normally including a pilot project) to confirm the fitness for purpose of the defined framework.
- If necessary, to define a set of criteria for evaluating architecture tools (an example set of criteria is given in Part IV), repositories and repository management processes to be used to capture, publish, and maintain architecture artifacts.

Approach

This preliminary phase is about defining "How we do Architecture" in the enterprise concerned. There are two main aspects: defining the framework to be used; and defining the architecture principles that will inform any architecture work.

The enterprise's approach to reuse of architecture assets is a key part of both the framework definition and architecture principles. (Typically the principles will state the policy on reuse; and the framework will explain how reuse is effected.)

In federated architectures (see Enterprise Scope / Focus in the Introduction to the Architecture Development Method), requirements from a higher level architecture are often manifested as "principles" in lower level ones.

Principles

This phase defines the architecture principles that will form part of the constraints on any architecture work undertaken in the enterprise. The issues involved in this are explained in Part IV, Architectural Principles.

Architecture work is informed by business principles as well as architecture principles. The architecture principles themsleves are also normally based in part on business principles. Defining business principles normally lies outside the scope of the Architecture function. However, depending on how such principles are defined and promulgated within the enterprise concerned, it may be possible for the set of architecture principles to also restate, or cross-refer to, a set of business principles, business goals, and strategic business drivers defined elsewhere within the enterprise. (Within an architecture project, the architect will normally need to ensure that the definitions of these business principles, goals, and strategic drivers are current, and to clarify any areas of ambiguity.)

The issue of architecture governance is closely linked to that of architecture principles. The body responsible for governance will also normally be responsible for approving the architecture principles, and for resolving architecture issues. This will normally be one of the principles cited. The issues involved in governance are explained in Part IV, IT Governance.

Framework

The TOGAF Architecture Development Method is a generic method, intended to be used by enterprises in a wide variety of industry types and geographies. It is also designed for use with a wide variety of other enterprise architecture frameworks, if

required (although it can be used perfectly well in its own right, without adaptation).

This phase therefore involves doing any necessary work to adapt the ADM to define an organization-specific framework, using either the TOGAF deliverables or the deliverables of another framework. The issues involved in this are discussed under Adapting the ADM in the *Introduction to the ADM*.

Inputs

The inputs to this phase are:

- TOGAF Architecture Development Method
- Other architecture framework(s), if required
- Business Strategy, Business Principles, Business Goals and Business Drivers (when pre-existing)
- IT Governance Strategy (when pre-existing)
- Architecture Principles (when pre-existing)
- Principles that are being subscribed to, arising from other, federated architectures

Steps

The TOGAF Architecture Development Method is a generic method, intended to be used by a wide variety of different enterprises, and in conjunction with a wide variety of other architecture frameworks, if required. It is not practical to define specific steps for adapting the ADM to such a wide variety of potential contexts. The issues involved are discussed in detail under Adapting the ADM in the *Introduction to the ADM*.

Outputs

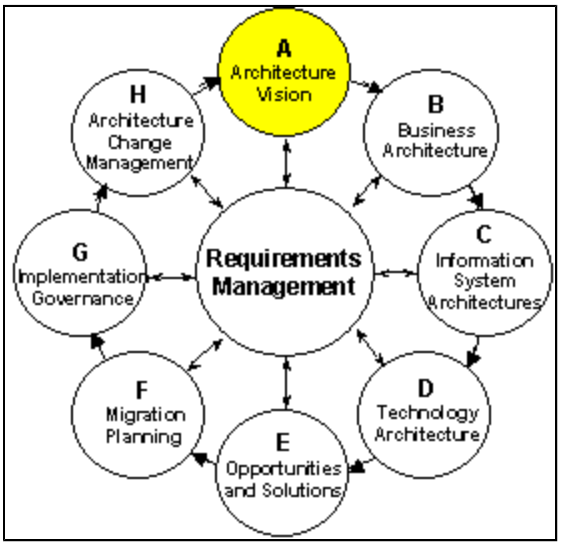
The outputs of this phase are:

- Framework Definition
- Architecture Principles
- Restatement of, or reference to, Business Principles, Business Goals and Business Drivers

Phase A: Architecture Vision

Objectives Approach Inputs Steps Outputs

Objectives



- To ensure that this evolution of the architecture development cycle has proper recognition and endorsement from the corporate management of the enterprise, and the support and commitment of the necessary line management.
- To validate the business principles, business goals, and strategic business drivers of the organization.
- To define the scope of, and to identify and prioritize the components of, the current architecture effort.
- To define the relevant stakeholders, and their concerns and objectives.
- To define the key business requirements to be addressed in this architecture effort, and the constraints that must be dealt with
- To articulate an architectural vision that demonstrates a response to those requirements and constraints.
- To secure formal approval to proceed.
- To understand the impact on, and of, other enterprise architecture development cycles going on in parallel.

Approach

General

The phase starts with receipt of a Request for Architecture Work from the sponsoring organization to the architecture organization.

The issues involved in ensuring proper recognition and endorsement from corporate management, and the support and commitment of line management, are discussed under IT Governance in Part IV.

This phase also defines what is in and what is outside the scope of the architecture effort and the constraints that must be dealt with. Scoping decisions need to be made on the basis of a practical assessment of resource and competence availability, and the value that can realistically be expected to accrue to the enterprise from the chosen scope of architecture work. The issues involved in this are discussed under Scoping the Architecture Activity in the section *Introduction to the ADM*.

The constraints will normally be informed by the business principles and architecture principles, developed as part of the preliminary phase, Framework and Principles.

Normally, the business principles, business goals, and strategic drivers of the organization are already defined elsewhere in the enterprise. If so, the activity in this phase is involved with ensuring that existing definitions are current, and clarifying any areas of ambiguity. Otherwise, it involves defining these essential items from scratch.

Similarly, the architecture principles that inform the constraints on architecture work will normally have been defined in the preliminary phase, Framework and Principles. The activity in this phase is concerned with ensuring that the existing principles definitions are current, and clarifying any areas of ambiguity. Otherwise, it entails defining the architecture principles from scratch, as explained in Part IV, Architectural Principles.

Creating the Architecture Vision

The Architecture Vision is essentially the architect's "elevator pitch" - the key opportunity to sell the benefits of the proposed development to the decision-makers within the enterprise. The goal is to articulate an architecture vision that enables the business goals, responds to the strategic drivers, conforms with the principles, and addresses the stakeholder concerns and objectives.

Clarifying and agreeing the purpose of the architecture effort is one of the key parts of this activity, and the purpose needs to be clearly reflected in the Vision that is created. Architecture projects are often undertaken with a specific purpose in mind - a specific set of business drivers that represent the RoI for the stakeholders in the architecture development. Clarifying that purpose, and demonstrating how it will be achieved by the proposed architecture development, is the whole point of the Architecture Vision.

Normally, key elements of the Architecture Vision such as the enterprise mission, vision, strategy, and goals have been documented as part of some wider business strategy or enterprise planning activity that has its own life-cycle within the enterprise. In such cases, the activity in this phase is concerned with verifying and understanding the documented business strategy and goals, and possibly bridging between the enterprise strategy and goals on the one hand, and the strategy and goals of the part of the enterprise that lies within the scope of the current architecture project.

In other cases, little or no business architecture work may have been done to date. In such cases, there will be a need for the architecture team to research, verify and gain buy-in to, the key business objectives and processes that the architecture is to support. This may be done as a free-standing exercise, either preceding architecture development, or as part of the ADM Initiation Phase (Phase A).

The Architecture Vision includes a first-cut, high-level description of the baseline ("as-is") and target ("to-be") environments, from both a business and a technical perspective. These outline descriptions are then built on in subsequent phases.

Business Scenarios are an appropriate and useful technique to discover and document business requirements, and to articulate an architectural vision that responds to those requirements. Business Scenarios are described in Part IV of TOGAF.

Once an Architectural Vision is defined and documented in the Statement of Architecture Work, it is critical to use it to build a consensus, as described in Part IV, IT Governance. Without this consensus it is very unlikely that the final architecture will be accepted by the organization as a whole. The consensus is represented by the sponsoring organization signing the Statement of Architecture Work.

Inputs

The inputs to this phase are:

- Request for Architecture Work
- Business Strategy, Business Principles, Business Goals and Business Drivers (when pre-existing)
- Architecture Principles (when pre-existing)
- Enterprise Continuum - existing architectural documentation (framework description, architectural descriptions, existing baseline descriptions, etc.)

Steps

Key steps in this phase include:

1. **Project Establishment**: Conduct the necessary (enterprise-specific) procedures to secure enterprise-wide recognition of the project, the endorsement of corporate management, and the support and commitment of the necessary line management. Include reference to the IT Governance framework for the enterprise, explaining how this project relates to that framework.
2. **Business Principles,Business Goals and Business Drivers**: Identify the business principles, business goals, and strategic drivers of the organization.
    - If these have already been defined elsewhere within the enterprise, ensure that the existing definitions are current, and clarify any areas of ambiguity. Otherwise, go back to the originators of the Statement of Architecture Work and work with them to define these essential items from scratch and secure their endorsement by corporate management.
3. **Architecture Principles**. Review the principles under which the current architecture is to be developed. Architecture Principles are normally based on the business principles developed as part of this Phase. They are explained, and an example set given, in Part IV, Architectural Principles. Ensure that the existing definitions are current, and clarify any

areas of ambiguity. Otherwise, go back to the body responsible for architecture governance and work with them to define these essential items from scratch and secure their endorsement by corporate management.

4. **Scope**: Define what is inside and what is outside the scope of the current architecture effort. The issues involved in this are discussed under Scope of the Architecting Activity in the section *Introduction to the ADM*. In particular, define:
    - the breadth of coverage of the enterprise
    - the level of detail to be defined
    - the specific architecture domains to be covered (Business, Data, Applications, Technology)
    - the extent of the time horizon aimed at, plus the number and extent of any intermediate time horizons
    - the architectural assets to be leveraged, or considered for use, from the organization's Enterprise Architecture Continuum
        - assets created in previous iterations of the ADM cycle within the enterprise
        - assets available elsewhere in the industry (other frameworks, systems models, vertical industry models, etc.)
5. **Constraints**: Define the constraints that must be dealt with, including enterprise-wide constraints and project-specific constraints (time, schedule, resources, etc.). The enterprise-wide constraints may be informed by the Business and Architectural Principles developed in the preliminary phase or clarified as part of this phase.
6. **Stakeholders and concerns, Business Requirements, and Architecture Vision**: Identify the key stakeholders and their concerns / objectives; define the key business requirements to be addressed in this architecture effort; and articulate an architecture vision that will address the requirements, within the defined scope and constraints, and conforming with the business and architectural principles.
    - Business Scenarios are an appropriate and useful technique to discover and document business requirements, and to articulate an architectural vision that responds to those requirements. Business Scenarios may also be used at more detailed levels of the architecture work (e.g. in Phase B, Business Architecture), and are described in Part IV of TOGAF.
    - If the Business Scenario technique is used, this step will generate the first, very high-level, definitions of the as-is (baseline) and to-be (target) environments, from both a business and technical perspective:
        - Business Baseline Architecture Version 1
        - Technical Baseline Architecture Version 1
        - Business Target Architecture Version 1
        - Technical Target Architecture Version 1
7. **Statement of Architecture Work and Approval**: Based on the purpose, focus, scope, and constraints, determine which architecture domains should be developed, to what level of detail, and which architecture views should be built. Estimate the resources needed, develop a roadmap and schedule for the proposed development, and document all these in the Statement of Architecture Work. Secure formal approval of the Statement of Architecture Work under the appropriate governance procedures.
    - A critical evaluation of the architectural starting point - the "as-is" environment described in the Architecture Vision - may be a key element of such a SoW, especially for an incremental approach, where neither the the architecture development nor the system implementation starts from scratch.

Business Scenarios

The ADM has its own method (a "method-within-a-method") for identifying and articulating the business requirements implied in new business functionality to address key business drivers, and the implied technical architecture requirements. This process is known as *Business Scenarios*, and is described in detail in Part IV. The technique may be used iteratively, at different levels of detail in the hierarchical decomposition of the Business Architecture. The generic Business Scenario process is as follows:

1. **Problem**: Identify, document and rank the problem that is driving the project.
2. **Business and technical environments**: Document, as high-level architecture models, the business and technical environment where the problem situation is occurring.
3. **Objectives and Measures of Success**: Identify and document desired objectives, the results of handling the problems successfully.
4. **Human Actors**: Identify human actors and their place in business model, the human participants and their roles.
5. **Computer Actors**: Identify computer actors and their place in technology model, the computing elements and their roles.
6. **Roles and Responsibilities**: Identify and document roles, responsibilities and measures of success per actor, the required scripts per actor, and the desired results of handling the situation properly.
7. **Refine**: Check for fitness for purpose of inspiring subsequent architecture work, and refine only if necessary.

Outputs

The outputs of this phase are:

- Approved Statement of Architecture Work / Project Definition, including in particular:
    - Scope and constraints

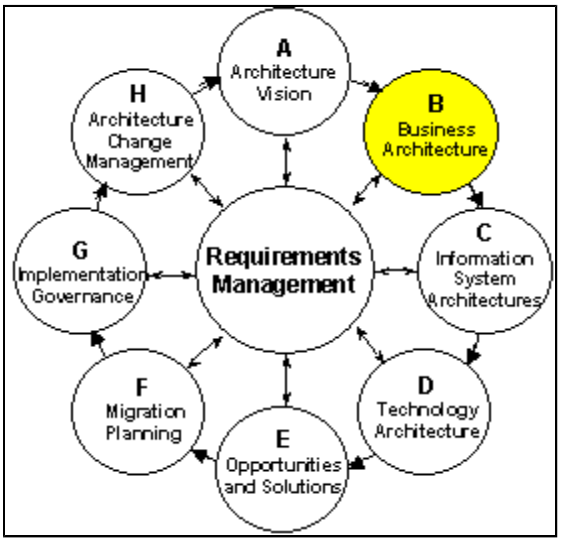
    - Plan for the architectural work
- Refined statements of Business Principles, Business Goals and Strategic Drivers
- Architecture Principles (if not previously existing)
- Architecture Vision
- Business Scenario, including:
    - Business Baseline Version 1
    - Technical Baseline Version 1
    - Business Architecture Version 1
    - Technical Architecture Version 1

Note: Multiple Business Scenarios may be used to generate a single Architecture Vision. In TOGAF terms, the Architecture Vision may also be referred to as a 'conceptual level architecture'.

Phase B: Business Architecture

Objectives Approach Inputs Steps Outputs

Objectives



- To describe the current baseline business architecture
- To develop a target Business Architecture, describing the product and/or service strategy, and the organizational, functional, process, information, and geographic aspects of the business environment, based on the business principles, business goals, and strategic drivers.
- To analyze the gaps between the baseline and target Business Architectures
- To select the relevant architectural viewpoints that will enable the architect to demonstrate how the stakeholder concerns are addressed in the Business Architecture.
- To select the relevant tools and techniques to be used in association with the selected viewpoints.

Approach

General

A knowledge of the business architecture is a prerequisite for architecture work in any other domain (data, applications, technology), and is therefore the first architecture activity that needs to be undertaken, if not catered for already in other organizational processes (enterprise planning, strategic business planning, business process re-engineering, etc.).

In practical terms, the business architecture is also often necessary as a means of demonstrating the business value of subsequent technical architecture work to key stakeholders, and the RoI to those stakeholders from supporting and participating in the subsequent work.

The extent of the work in this phase will depend to a large extent on the enterprise environment. In some case, key elements of the Business Architecture may be done in other activities; for example, the enterprise mission, vision, strategy and goals may be documented as part of some wider business strategy or enterprise planning activity that has its own life-cycle within the enterprise.

In such cases, there may be a need to verify and update the currently documented business strategy and plans, and/or to bridge between high-level business drivers, business strategy and goals on the one hand, and the specific business requirements that are relevant to this architecture development effort. (The business strategy typically defines what to achieve - the goals and drivers, and the metrics for success - but not how to get there. That is role of the Business Architecture.)

In other cases, little or no business architecture work may have been done to date. In such cases, there will be a need for the architecture team to research, verify and gain buy-in to, the key business objectives and processes that the architecture is to support. This may be done as a free-standing exercise, either preceding architecture development, or as part of the ADM Architecture Vision (Phase A).

In both of these cases, the Business Scenario technique of the TOGAF ADM, or any other method that illuminates the key business requirements and indicates the implied technical requirements for IT architecture, may be used.

A key objective is to reuse existing material as much as possible. In architecturally more mature environments, there will be existing architecture definitions, which (hopefully) will have been maintained since the last architecture development cycle. Where existing architectural descriptions exist, these can be used as a starting point, and verified and updated if necessary. (See Architecture Continuum, below.)

Gather and analyze only that information that allows informed decisions to be made relevant to the scope of this architecture effort. If this effort is focused on the definition of (possibly new) business processes, then this Phase will necessarily involve a

lot of detailed work. If the focus is more on the target architectures in other domains (data/information, application systems, infrastructure) to support an essentially existing business architecture, then it is important to build a complete picture in this Phase without going into unnecessary detail.

Developing the Baseline Description

In architecturally more mature environments, there will be existing architecture definitions, which (hopefully) will have been maintained since the last architecture development cycle. Where these existing architectural descriptions exist, they can be used as a starting point, and verified and updated if necessary. Any such existing descriptions will already have been used in Phase A in developing an Architecture Vision, and this work should provide a sound basis for the Baseline Description, and may even be sufficient in itself.

Where no such existing descriptions exist, information will have to be gathered in whatever format comes to hand.

The normal approach to Target Architecture development is top-down.In Baseline Description, however, the analysis of the current state often has to be done bottom-up, particularly where little or no existing architecture assets exist. In such a case, the architect simply has to document the working assumptions about high-level architectures, and the process is one of gathering evidence to turn the working assumptions into fact, until the law of diminishing returns sets in.

Business processes that are not to be carried forward have no intrinsic value. However, when developing baseline descriptions in other architecture domains, architectural components (principles, models, standards, and current inventory) that are not to be carried forward may still have an intrinsic value, and an inventory may be needed in order to understand the residual value (if any) of those components.

Whatever the approach, the goal should be to reuse existing material as much as possible, and to gather and analyze only that information that allows informed decisions to be made regarding the Target Business Architecture. It is important to build a complete picture without going into unnecessary detail.

Business Modeling

A variety of modeling tools and techniques may be employed, if deemed appropriate (bearing in mind the above caution not to go into unnecessary detail). For example:

- **Activity Models** (also called **Business Process Models**) describe the functions associated with the enterprise's business activities, the data and/or information exchanged between activities (internal exchanges), and the data and/or information exchanged with other activities that are outside the scope of the model (external exchanges). Activity Models are hierarchical in nature. They capture the activities performed in a business process, and the ICOMs (inputs, controls, outputs, and mechanisms / resources used) of those activities. Activity Models can be annotated with explicit statements of business rules, which represent relationships among the ICOMs. For example, a business rule can specify who can do what under specified conditions, the combination of inputs and controls needed, and the resulting outputs. One technique for creating Activity Models is the IDEF (Integrated Computer Aided Manufacturing (ICAM) DEFinition) modeling technique.
    - The Business Process Management Initiative (BPMI.org) is an organization that is defining standards for business process modeling, including a language with which to specify business processes, their tasks/steps, and the documents produced.

- **Use Case Models** can describe either business processes or systems functions, depending on the focus of the modeling effort. A Use Case Model describes the business processes of an enterprise in terms of use cases and actors corresponding to business processes and organizational participants (people, organizations, etc.). The Use Case Model is described in Use Case Diagrams and Use Case Specifications.

- **Class Models** are similar to logical data models. A Class Model describes static information and relationships between information. A Class Model also describes informational behaviors. Like many of the other models, it also can be used to model various levels of granularity. Depending on the intent of the model, a Class Model can represent business domain entities or systems implementation classes. A business domain model represents key business information (domain classes), their characteristics (attributes), their behaviors (methods or operations), and relationships (often referred to as multiplicity, describing how many classes typically participate in the relationship), and cardinality (describes required or optional participation in the relationship). Specifications further elaborate and detail information that cannot be represented in the class diagram.

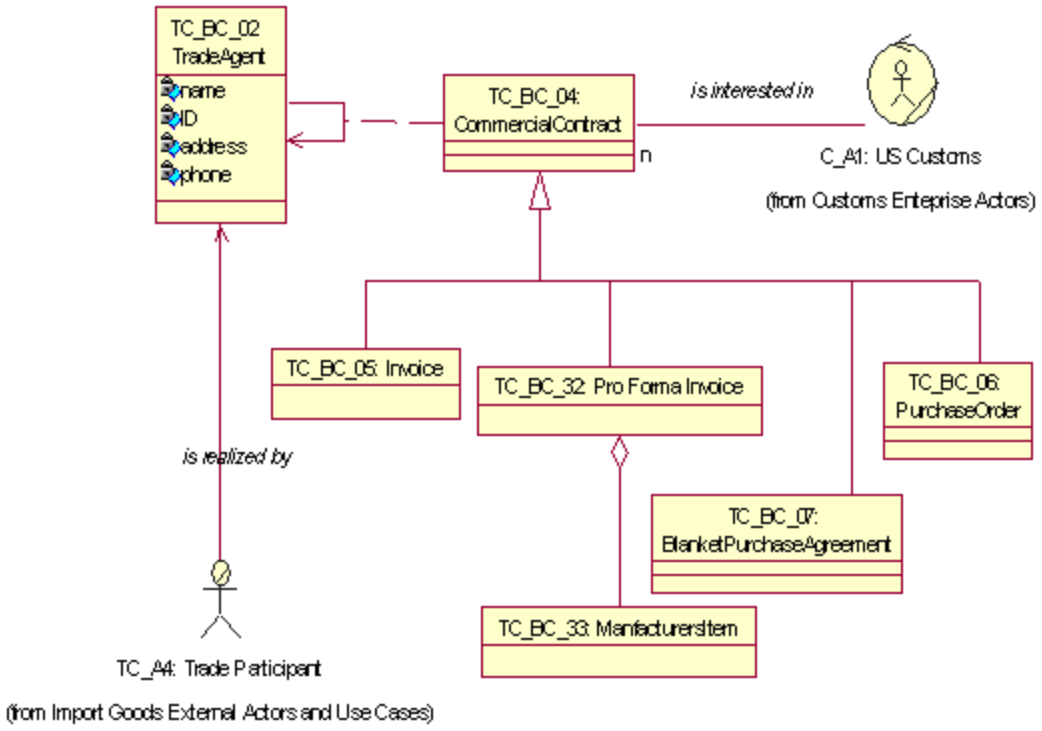


Figure 1: UML Business Class Diagram, Trade Class Model (Commercial View) (taken from "A Practical Guide to Federal Enterprise Architecture", CIO Council, February 2001)

All three types of model types above can be represented in Unified Modeling Language (UML), and a variety of tools exists for generating such models.

Certain industry sectors have modeling techniques specific to that sector concerned. For example, the Defense sector uses the following models:

- The **Node Connectivity Diagram** describes the business locations (nodes), the "needlines" between them, and the characteristics of the information exchanged. Node connectivity can be described at three levels: conceptual, logical, and physical. Each needline indicates the need for some kind of information transfer between the two connected nodes. A node can represent a role (e.g., a CIO); an organizational unit; a business location or facility; and so on. An arrow indicating the direction of information flow is annotated to describe the characteristics of the data or information - for example, its content; media; security or classification level; timeliness; and requirements for information system interoperability.

- The **Information Exchange Matrix** documents the Information Exchange Requirements for an Enterprise Architecture. Information Exchange Requirements express the relationships across three basic entities (activities, business nodes and their elements, and information flow), and focus on characteristics of the information exchange, such as performance and security. They identify who exchanges what information with whom, why the information is necessary, and in what manner.

Although originally developed for use in the Defense sector, these models are finding increasing use in the other sectors of government, and may also be considered for use in non-government environments.

Enterprise Continuum

As part of this Phase, the architecture team will need to consider what relevant Business Architecture resources are available from the Enterprise Continuum: in particular,

- Generic business models relevant to the organization's industry sector. (These are "Industry Architectures", in terms of the Enterprise Continuum.) For example:
  - The Object Management Group (OMG) has a number of vertical Domain Task Forces developing business models relevant to specific vertical domains such as Healthcare, Transportation, Finance, etc.

    - The TeleMangement Forum (TMF) has developed detailed business models relevant to the Telecommunications Industry;
    - Government departments and agencies in different countries have reference models and frameworks mandated for use, intended to promote cross-departmental integration and interoperability. An example is the Federal Enterprise Architecture Business Reference Model, which is a function-driven framework for describing the business operations of the Federal Government independent of the agencies that perform them.

- Business models relevant to common high-level business domains, such as electronic commerce, supply chain management, etc., that are published within the IT industry. (These are "Common Systems Architectures", in terms of the Enterprise Continuum.) For example:
    - The Resource-Event-Agent (REA) business model was originally created by William E. McCarthy of Michigan State University, mainly for modeling of accounting systems. It has proved so useful for better understanding of business processes that it has become one of the major modeling frameworks for both traditional enterprises and e-commerce systems. In particular, it has been extended by Robert Haugen and McCarthy for supply chain management.
    - The STEP Framework (STandard for the Exchange of Product model data) is concerned with product design and supply chain interworking. STEP is an ISO standard (ISO 10303). Implementation of the STEP standard has been lead by some large aerospace manufacturers, and has also been taken up in other industries that have a need for complex graphic and process data, such as the construction industry.
    - The Open Applications Group (OAG) is focused on defining a framework for allowing heterogeneous business applications to communicate together. Its OAGIS integration model and specification address the needs of traditional ERP integration, as well as supply chain management and electronic commerce.
    - RosettaNet is a consortium created by leading companies in the computer, electronic component, and semiconductor manufacturing supply chains. Its mission is to develop a complete set of standard e-Business processes for these supply chains, and to promote and support their adoption and use.

- Enterprise-specific building blocks (process components, business rules, job descriptions, etc.)

Gap Analysis

A key step in validating an architecture is to consider what may have been forgotten. The architecture must support all of the essential information processing needs of the organization. The most critical source of gaps that should be considered is stakeholder concerns that have not been addressed in prior architectural work.

Other potential sources of gaps:

- People gaps (e.g., cross-training requirements)
- Process gaps (e.g., process inefficiencies)
- Tools gaps (e.g., duplicate or missing tool functionality)
- Information gaps
- Measurement gaps
- Financial gaps
- Facilities gaps (buildings, office space, etc.)

Gap analysis highlights services and/or functions that have been accidentally left out, deliberately eliminated, or are yet to be developed or procured. Table 1 in Phase D illustrates an example of a gap analysis matrix. The suggested steps are as follows:

- Draw up a matrix with all the business architecture building blocks of the current architecture on the vertical axis, and all the business architecture building blocks of the target Business Architecture on the horizontal axis. In creating the matrix it is imperative to use terminology that is accurate and consistent.
- Add to the Current Architecture axis a final row labeled 'New business architecture building blocks', and to the Target Architecture axis a final column labeled 'Eliminated business architecture building blocks'.
- Where a business architecture building block is available in both the current and target architectures, record this with 'Included' at the intersecting cell.
- Where a business architecture building block from the current architecture is missing in the target architecture, each must be reviewed. If it was correctly eliminated, mark it as such in the appropriate 'Eliminated' cell. If it was not, you have uncovered an accidental omission in your new architecture that must be addressed by reinstating the business architecture building block in the next iteration of the architecture design - mark it as such in the appropriate 'Eliminated' cell.
- Where a business architecture building block from the target architecture cannot be found in the current architecture, mark it at the intersection with the 'New' row, as a gap that needs to filled, either by developing or procuring the building block.

When the exercise is complete, anything under 'Eliminated Services' or 'New Services' is a gap, which should either be explained as correctly eliminated, or marked as to be addressed by reinstating or developing/procuring the function.

Inputs

The inputs to this phase are:

- Request for Architecture Work
- Approved Statement of Architecture Work / Project Definition
- Refined statements of Business Principles, Business Goals and Strategic Drivers
- Architecture Principles
- Enterprise Continuum (as described above)
- Architecture Vision / Business Scenario, including:
  - Business Baseline Version 1
  - Technology Baseline Version 1
  - Business Architecture Version 1
  - Technology Architecture Version 1

Steps

The level of detail addressed in this phase will depend on the scope and goals of the overall architecture effort.

New business processes being introduced as part of this effort will need to be defined in detail during this phase. Existing business processes to be carried over and supported in the target environment may already have been adequately defined in previous architectural work; but, if not, they too will need to be defined in this phase.

Key steps in this phase include the following:

Note: the order of the following steps should be adapted to the situation at hand: in particular, determine whether in this situation it is appropriate to do baseline description or target architecture development first, as described in the ADM Introduction.

1. **Business Baseline Description**. Develop a baseline description of the existing business architecture, to the extent necessary to support the target Business Architecture. The scope and level of detail to be defined will depend on the extent to which existing business elements are likely to be carried over into the target business architecture, and on whether existing architectural descriptions exist, as describe under Approach, above. To the extent possible, identify the relevant business architecture building blocks, drawing on the Architecture Continuum.
2. **Reference Models, Viewpoints and Tools:**
   i. **Select relevant business architecture resources (reference models, patterns, etc.) from the Architecture Continuum, on the basis of the business drivers, and the stakeholders and concerns.**
   ii. **Select relevant business architecture viewpoints** (e.g., Operations; Management; Financial) - i.e., those that will enable the architect to demonstrate how the stakeholder concerns are being addressed in the Business Architecture.
   iii. **Identify appropriate tools and techniques** to be used for capture, modeling, and analysis, in association with the selected viewpoints. Depending on the degree of sophistication warranted, these may comprise simple documents or spreadsheets, or more sophisticated modeling tools and techniques such as activity models, business process models, use case models, etc.
3. **Architecture Model(s).**
   i. For each viewpoint, create the model for the specific view required, using the selected tool or method.
   ii. Assure that all stakeholder concerns are covered. If they are not, create new models to address concerns not covered, or augment existing models (see above). Business Scenarios are a useful technique to discover and document business requirements, and may be used iteratively, at different levels of detail in the hierarchical decomposition of the Business Architecture. Business Scenarios are described in Part IV of TOGAF. Other techniques may be used, if required. Create models of the following:
      i. *Organization structure*. Document the organization structure, identifying business locations and relating them to organizational units.
      ii. *Business goals and objectives*. Document business goals and objectives for each organizational unit.
      iii. *Business functions*. Identify and define business functions. This is a detailed, recursive step involving successive decomposition of major functional areas into sub-functions.
      iv. *Business Services - the services that each of the enterprise units provides to its customers, both internally and externally.*
      v. *Business processes, including measures and deliverables.*
      vi. *Business roles, including development and modification of skills requirements.*
      vii. *Correlation of organization and functions*. Relate business functions to organizational units in the form of a

matrix report.

iii. **information requirements.** Identify for each business function: when, where, how often, and by whom the function is performed; what information is used to perform it, and its source(s); and what opportunities exist for improvements. **Include information that needs to be created, retrieved, updated, and deleted.** The level of detail to be defined will depend on the scope and focus of the current architecture effort, as describe under Approach, above. Focus on what will be worthwhile collecting for the purpose at hand.

iv. Perform trade-off analysis to resolve conflicts (if any) among the different views.
   - One method of doing this is CMU / SEI's Architecture Trade-off Analysis (ATA) Method.

v. Validate that the models support the principles, objectives and constraints.

vi. Note changes to the viewpoint represented in the selected models from the Architecture Continuum, and document.

vii. Test architecture models for completeness against requirements.

4. **Select business architecture building blocks** (e.g., business services)
   i. Identify required building blocks and check against existing library of building blocks, re-using as appropriate.
   ii. Where necessary, define new business architecture building blocks.
5. **Conduct formal checkpoint review** of the architecture model and building blocks with stakeholders.
6. **Review non-functional (qualitative) criteria** (e.g., performance, costs, volumes). Use to specify required service levels (for example, via formal Service Level Agreements).
7. **Complete the business architecture**.
   i. Select standards for each of the architecture building blocks, reusing as much as possible from the reference models selected from the Architecture Continuum.
   ii. Fully document each architecture building block.
   iii. Final cross check of overall architecture against business goals. Document rationale for building block decisions in architecture document.
   iv. Document final requirements traceability report.
   v. Document final mapping of the architecture within the Architecture Continuum. From the selected architecture building blocks, identify those that might be reused (working practices, roles, business relationships, job descriptions, etc.), and publish via the architecture repository.
   vi. Document rationale for building block decisions in architecture document.
   vii. Prepare Business Architecture Report. Generate the Business Architecture document, comprising some or all of:
      - a business footprint (a high-level description of the people and locations involving with key business functions);
      - a detailed description of business functions and their information needs;
      - a management footprint (showing span of control and accountability);
      - standards, rules and guidelines showing working practices, legislation, financial measures, etc.; and
      - a skills matrix and set of job descriptions.

   If appropriate, use reports and/or graphics generated by modeling tools to demonstrate key views of the architecture. Route the Business Architecture document for review by relevant stakeholders, and incorporate feedback.

   viii. *Checkpoint*: Check the original motivation for the architecture project and the Statement of Architecture Work against the proposed Business Architecture, asking if it is fit for the purpose of supporting subsequent work in the other architecture domains. Refine the proposed Business Architecture only if necessary.
8. **Gap analysis (see under Approach, above) and report**
   i. Create gap matrix as described above.
   ii. Identify building blocks to be carried over, classifying as either changed or unchanged.
   iii. Identify eliminated building blocks.
   iv. Identify new building blocks.
   v. Identify gaps and classify as those that should be developed and those that should be procured.

Outputs

The outputs of this phase are:

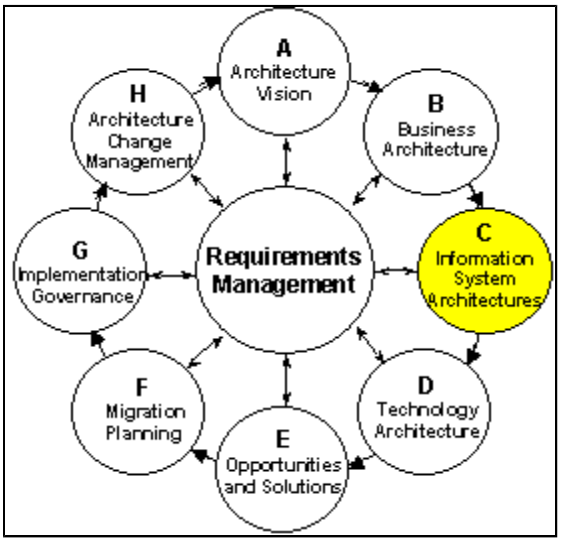
- Statement of Architecture Work (updated if necessary)
- Validated Business Principles, business goals, and strategic drivers
- Target Business Architecture - Version 2 (detailed), including:
  - *Organization structure*. identifying business locations and relating them to organizational units.
  - *Business goals and objectives*. for each organizational unit.
  - *Business functions*. a detailed, recursive step involving successive decomposition of major functional areas into sub-functions.
  - *Business Services* - the services that each enterprise unit provides to its customers, both internally and externally.
  - *Business processes,* including measures and deliverables
  - *Business roles*, including development and modification of skills requirements.

    - *Correlation of organization and functions*. Relate business functions to organizational units in the form of a matrix report.
- Business Baseline - Version 2 (detailed) - if appropriate
- Views corresponding to the selected viewpoints addressing key stakeholder concerns
- Gap analysis results
- Technical requirements (drivers for the Technical Architecture work): identifying, categorising and prioritising the implications for work in the remaining architecture domains; for example, by a dependency/ priority matrix. (For example, guiding trade-off between speed of transaction processing and security.) List the specific models that are expected to be produced (for example, expressed as primitives of the Zachman Framework).
- Business Architecture Report
- Updated business requirements

Phase C: Information System Architectures (Introduction)

Objective Approach Inputs Steps Outputs

Objective



The objective of this phase is to develop target architectures covering either or both (depending on project scope) of the Data and Application Systems domains.

The scope of the business processes supported in this phase is limited to those that are supported by information technology, and the interfaces of those IT-related processes to non-IT-related processes.

Approach

Development

This phase involves some combination of Data and Applications Architecture, in either order. Advocates exist for both sequences.

For example, Spewak's Enterprise Architecture Planning recommends a data-driven approach.

On the other hand, major applications systems such as those for Enterprise Resource Planning, Customer Relationship Management, etc., often provide a combination of technology infrastructure and business application logic, and some organizations take an application-driven approach, whereby they recognize certain key applications as forming the core underpinning of the mission-critical business processes, and take the implementation and integration of those core applications as the primary focus of architecture effort (the integration issues often constituting a major challenge).

Implementation

Implementation of these architectures may not necessarily follow the same order. For example, one common implementation approach is top-down design and bottom-up implementation:

- Design:
  1. Business Architecture design
  2. Data (or Applications) Architecture design
  3. Applications (or Data) Architecture design
  4. Technical Architecture design
- Implementation:
  1. Technical Architecture implementation
  2. Applications (or Data) Architecture implementation
  3. Data (or Applications) Architecture implementation
  4. Business Architecture implementation

An alternative approach is a data-driven sequence, whereby application systems that create data are implemented first, then applications that process the data, and finally applications that archive data.

Inputs

Inputs to this phase are:

- Applications Principles (if existing)
- Data Principles (if existing)
- Request for Architecture Work
- Statement of Architecture Work

- Architecture Vision
- Enterprise Continuum
- Business Baseline - Version 2 (detailed) - if appropriate
- Target Business Architecture - Version 2 (detailed)
- Relevant technical requirements that will apply to this phase
- Gap analysis (from Business Architecture)
- Re-usable building blocks (from organization's Architecture Continuum, if available)

Steps

Detailed steps for this phase are given separately for each architecture domain:

- Data Architecture
- Applications Architecture

Outputs

The main outputs are as follows:

- Statement of Architecture Work (updated if necessary)
- Target Data Architecture
- Target Applications Architecture
- Data Architecture Views corresponding to the selected viewpoints addressing key stakeholder concerns
- Applications Architecture Views corresponding to the selected viewpoints addressing key stakeholder concerns
- Data Architecture Report, summarizing what was done and the key findings
- Applications Architecture Report, summarizing what was done and the key findings
- Gap analysis
    - Areas where the Business Architecture may need to change to cater for changes in the Data and/or Applications Architecture
    - Constraints on the Technology Architecture about to be designed.
- Impact Analysis
- Updated business requirements (if appropriate)

Phase C: Information System Architecture - Data

Objective Approach Inputs Steps Outputs Mapping to Zachman Framework

Objective

The objective here is to define the major types and sources of data necessary to support the business, in a way that is

- understandable by stakeholders
- complete and consistent
- stable

It is important to note that this effort is NOT concerned with database design. The goal is to define the data entities relevant to the enterprise, not to design logical or physical storage systems. (However, linkages to existing files and databases may be developed, and may demonstrate significant areas for improvement.)

Approach

Enterprise Continuum

As part of this Phase, the architecture team will need to consider what relevant data architecture resources are available in the organization's Enterprise Continuum; in particular, generic data models relevant to the organization's industry "vertical" sector. For example:

- ARTS has defined a Data Model for the Retail Industry
- POSC has defined a Data Model for the Petrotechnical industry

Gap Analysis

A key step in validating an architecture is to consider what may have been forgotten. The architecture must support all of the essential information processing needs of the organization. The most critical source of gaps that should be considered is stakeholder concerns that have not been addressed in architectural work.

Types of data gap:

- data not located where it is needed
- not the data that is needed
- data not available when needed
- data not created
- data not consumed
- data relationship gaps
- etc.

Gap analysis highlights shortfalls in data services and/or data elements that have been accidentally left out, deliberately eliminated, or are yet to be defined. Table 1 in Phase D illustrates an example of a gap analysis matrix. The suggested steps are as follows:

- Draw up a matrix with all the Data architecture building blocks of the current architecture on the vertical axis, and all the Data architecture building blocks of the target Data Architecture on the horizontal axis. In creating the matrix it is imperative to use terminology that is accurate and consistent.
- Add to the Current Architecture axis a final row labeled 'New Data architecture building blocks', and to the Target Architecture axis a final column labeled 'Eliminated Data architecture building blocks'.
- Where a Data architecture building block is available in both the current and target architectures, record this with 'Included' at the intersecting cell.
- Where a Data architecture building block from the current architecture is missing in the target architecture, each must be reviewed. If it was correctly eliminated, mark it as such in the appropriate 'Eliminated' cell. If it was not, you have

uncovered an accidental omission in your new architecture that must be addressed by reinstating the Data architecture building block in the next iteration of the architecture design - mark it as such in the appropriate 'Eliminated' cell.
- Where a Data architecture building block from the target architecture cannot be found in the current architecture, mark it at the intersection with the 'New' row, as a gap that needs to filled, either by defining or inheriting the building block.

When the exercise is complete, anything under 'Eliminated Services' or 'New Services' is a gap, which should either be explained as correctly eliminated, or marked as to be addressed by reinstating or developing/procuring the function.

Table 1 in Phase D illustrates an example of a gap analysis matrix.

Inputs

Inputs to this phase are:

- Data Principles (if existing)
- Request for Architecture Work
- Statement of Architecture Work
- Architecture Vision
- Relevant technical requirements that will apply to this phase
- Gap analysis (from Business Architecture)
- Business Baseline - Version 2 (detailed) - if appropriate
- Target Business Architecture - Version 2 (detailed)
- Re-usable building blocks (from organization's Enterprise Continuum, if available)
    - In particular, definitions of current data

Steps

1. **Data Baseline Description.** Develop a baseline description of the existing Data architecture, to the extent necessary to support the target Data Architecture. The scope and level of detail to be defined will depend on the extent to which existing Data elements are likely to be carried over into the target Data architecture, and on whether existing architectural descriptions exist, as describe under Approach, above. To the extent possible, identify the relevant Data architecture building blocks, drawing on the Architecture Continuum, and review / verify the following primitives from the Zachman Framework:
    - Conceptual data model (entities, attributes, and relationships)
        - Entity-Relationship diagrams illustrating views of the data architecture to address the concerns of stakeholders.
    - Logical data model (logical views of the actual data of interest)
    - Data Management Process models, including:
        i. Data dissemination view
        ii. Data lifecycle view
        iii. Data security view
        iv. Data model management view
    - Data entity / business function matrix in the Business Architecture
2. **Principles, Reference Models, Viewpoints and Tools:**
    i. Review and validate (or generate, if necessary) the set of data principles.
        - These will normally form part of an overarching set of Architecture Principles. Guidelines for developing and applying principles, and a sample set of data principles, are given in Part IV, Architecture Principles.
    ii. Select relevant Data architecture resources (reference models, patterns, etc.) from the Architecture Continuum, on the basis of the business drivers, and the stakeholders and concerns.
    iii. Select relevant Data architecture viewpoints (for example, stakeholders of the data - regulatory bodies, users, generators, subjects, auditors, etc.; various time dimensions - real-time, reporting period, event-driven, etc.; locations; business processes) - i.e., those that will enable the architect to demonstrate how the stakeholder concerns are being addressed in the Data Architecture.
    iv. Identify appropriate tools and techniques ~~(including forms)~~ to be used for ~~data~~ capture, modeling, and analysis, in association with the selected viewpoints. Depending on the degree of sophistication warranted, these may comprise simple documents or spreadsheets, or more sophisticated modeling tools and techniques such as data management models, data models, etc. Examples of data modeling techniques are:
        - IDEF
        - Object Role Modeling
3. **Architecture Model(s).**
    i. For each viewpoint, create the model for the specific view required, using the selected tool or method.
        i. Examples of logical data models are:
            - the C4ISR Architecture Framework Logical Data Model
            - ARTS has defined a Data Model for the Retail Industry

        - POSC has defined a Data Model for the Petrotechnical industry
    ii. Assure that all stakeholder concerns are covered. If they are not, create new models to address concerns not covered, or augment existing models (see above). Model the following:
        i. Conceptual data model (entities, attributes, and relationships)
            - Draw Entity-Relationship diagrams to illustrate views of the data architecture to address the concerns of stakeholders.
        ii. Logical data model (logical views of the actual data of interest)
        iii. Data Management Process models, including:
            i. Data dissemination view
            ii. Data lifecycle view
            iii. Data security view
            iv. Data model management view
        iv. Relate data entities to business functions in the Business Architecture, indicating which of the CRUD operations (Create, Reference, Update, and Delete) are performed by which functions.
            - Relate each lowest level business function in the Business Architecture to the set of data entities, indicating which of the CRUD operations (Create, Reference, Update, and Delete) are performed by the function concerned.
            - Generate Entity-Business Function matrices tabulating all the relationships.
            - Review and validate the Entity-Business Function matrices, checking that each entity is created by at least one function, and referenced or updated by at least one other function.
            - Time permitting, relate entities to the application systems described in the Baseline Description.
    iii. Ensure that all information requirements in the Business Architecture are met.
    iv. Perform trade-off analysis to resolve conflicts (if any) among the different views.
        - One method of doing this is CMU / SEI's Architecture Trade-off Analysis (ATA) Method.
    v. Validate that the models support the principles, objectives and constraints.
    vi. Note changes to the viewpoint represented in the selected models from the Architecture Continuum, and document.
    vii. Test architecture models for completeness against requirements.
4. **Select Data architecture building blocks** (e.g., metamodels)
    i. Identify required building blocks and check against existing library of building blocks, re-using as appropriate.
    ii. Where necessary, define new Data architecture building blocks.
5. **Conduct a formal checkpoint review** of the architecture model and building blocks with stakeholders.
    - Review the Entity-Business Function matrices generated in step 3, and the Business Architecture generated in Phase B.
6. **Review the qualitative criteria** (e.g., security, availability, accuracy, performance, costs, volumes), providing as many measurable criteria as possible (e.g., privacy / confidentiality, reliability, minimum tolerable data losses, maximum data volumes at peak times, etc.). Use to specify required service levels for data services (for example, via formal Service Level Agreements).
    - The goal here is to guide the Applications and Technology Architecture efforts as to the qualities required in the applications, and the underlying technology, that manage and process the data.
7. **Complete the Data architecture**.
    i. Select standards for each of the architecture building blocks, reusing as much as possible from the reference models selected from the Architecture Continuum.
    ii. Fully document each architecture building block.
    iii. Final cross check of overall architecture against business requirements. Document rationale for building block decisions in architecture document.
    iv. Document final requirements traceability report.
    v. Document final mapping of the architecture within the Architecture Continuum. From the selected architecture building blocks, identify those that might be reused, and publish via the architecture repository.
    vi. Document rationale for building block decisions in architecture document.
    vii. Prepare Data Architecture Report. Generate the Data Architecture document, comprising some or all of:
        - Conceptual data model
        - Logical data model
        - Data management process model
        - Data entity / business function matrix
        - Data interoperability requirements (e.g., XML schema, security policies)

    If appropriate, use reports and/or graphics generated by modeling tools to demonstrate key views of the architecture. Route the Data Architecture document for review by relevant stakeholders, and incorporate feedback.

    viii. Checkpoint/Impact analysis: Check the original motivation for the architecture project and the Statement of Architecture Work against the proposed Data Architecture. Conduct an impact analysis, to:
        i. Identify any areas where the Business Architecture (e.g., business practices) may need to change to cater for changes in the Data Architecture (for example, changes to forms or procedures, application systems or database systems).
            - If the impact is significant, this may warrant the Business Architecture being revisited.
        ii. Identify any areas where the Application Architecture (if generated at this point) may need to change to

cater for changes in the Data Architecture (or to identify constraints on the Application Architecture about to be designed).
        - If the impact is significant, this may warrant the Application Architecture being revisited, if already developed in this cycle).
    iii. Identify any constraints on the Technology Architecture about to be designed.
    iv. Refine the proposed Data Architecture only if necessary.
8. **Gap analysis** (see under Approach, above) and report
    i. Create gap matrix as described above.
    ii. Identify building blocks to be carried over, classifying as either changed or unchanged.
    iii. Identify eliminated building blocks.
    iv. Identify new building blocks.
    v. Identify gaps and classify as those that should be defined and those inherited.

Outputs

The outputs of this phase are:

- Statement of Architecture Work (updated if necessary)
- Data Baseline Description - if appropriate
- Validated Data Principles, or new Data Principles (if generated here)
- Target Data Architecture
    - Conceptual data model
    - Logical data model
    - Data Management Process models
    - Data entity / business function matrix
    - Data interoperability requirements
- Viewpoints addressing key stakeholder concerns.
- Views corresponding to the selected viewpoints; e.g.:
    - Data dissemination view
    - Data lifecycle view
    - Data security view
    - Data model management view
- Gap analysis results
- Relevant technical requirements that will apply to this evolution of the architecture development cycle
- Data Architecture Report, summarizing what was done and the key findings
- Impact Analysis
    - Areas where the Business Architecture may need to change to cater for changes in the Data Architecture
    - Identify any areas where the Applications Architecture (if generated at this point) may need to change to cater for changes in the Data Architecture.
    - Constraints on the Technology Architecture about to be designed.
- Updated business requirements (if appropriate)

Phase C: Information System Architecture - Applications

Objective Approach Inputs Steps Outputs

Objective

The objective here is to define the major kinds of application system necessary to process the data and support the business.

It is important to note that this effort is NOT concerned with applications systems design. The goal is to define what kinds of application systems are relevant to the enterprise, and what those applications need to do in order to manage data and to present information to the human and computer actors in the enterprise.

The applications are not described as computer systems, but as logical groups of capabilities that manage the data objects in the data architecture and support the business functions in the Business Architecture. The applications and their capabilities are defined without reference to particular technologies. The applications are stable and relatively unchanging over time, whereas the technology used to implement them will change over time, based on the technologies currently available and changing business needs.

Approach

Enterprise Continuum

As part of this Phase, the architecture team will need to consider what relevant applications architecture resources are available in the Enterprise Continuum.

In particular:

- Generic business models relevant to your organization's industry "vertical" sector. For example:
  - the TeleMangement Forum (TMF) has developed detailed applications models relevant to the Telecommunications Industry;
  - the Object Management Group (OMG) has a number of vertical Domain Task Forces developing software models relevant to specific domains such as Healthcare, Transportation, Finance, etc.
- Application models relevant to common high-level business functions, such as electronic commerce, supply chain management, etc.

The Open Group has a Reference Model for Integrated Information Infrastructure that focuses on the application-level components and services necessary to provide an integrated information instructure.

In addition the ebXML Initiative aims to provide an open, XML-based infrastructure enabling global use of electronic business information in an interoperable, secure and consistent manner. UML is used for modelling aspects and XML for syntax aspects. The initiative was formed as a joint venture by the UN/CEFACT community and the OASIS consortium, with ANSI X.12 also fully participating.

Gap Analysis

A key step in validating an architecture is to consider what may have been forgotten. The architecture must support all of the essential information processing needs of the organization. The most critical source of gaps that should be considered is stakeholder concerns that have not been addressed in architectural work.

Gap analysis highlights shortfalls in applications services and/or applications components that have been accidentally left out, deliberately eliminated, or are yet to be defined. Table 1 in Phase D illustrates an example of a gap analysis matrix. The suggested steps are as follows:

1. Draw up a matrix with all the applications in the current Applications Architecture on the vertical axis, and all the applications in the Target Applications Architecture on the horizontal axis.
2. Add to the Current Architecture axis a final row labeled 'New Applications', and to the Target Architecture axis a final

column labeled 'Eliminated Applications'.

3. Where an application exists in both the current and target architectures, record this fact with 'Retained' at the intersecting cell.
4. Where an application in the current architecture is missing in the target architecture, the current and target architectures must be reviewed.
    - If the current application was correctly eliminated, mark it as such in the appropriate 'Eliminated Applications' cell.
    - If the current application is to be replaced, wholly or partly, by one or more applications in the target architecture, make a note to this effect in the corresponding intersecting cell(s).
    - If the current application was unintentionally eliminated in the target architecture, note this fact in the appropriate 'Eliminated Applications' cell. The omission will need to be addressed in an iteration of the Target Applications Architecture design.
5. Where an application in the Target Applications Architecture cannot be found in the current architecture, mark it at the intersection with the 'New' row, as a gap that needs to filled, either by developing or procuring the application.
6. When the exercise is complete, anything under 'Eliminated Applications' or 'New Applications' is a potential gap, which should either be explained as correctly eliminated, or marked as to be addressed, either by reinstating in the target architecture, or by developing/procuring the application.
7. Check that the applications gap analysis is complete.

Table 1 in Phase D illustrates an example of a gap analysis matrix.

Inputs

Inputs to this phase are:

- Applications Principles (if existing)
- Request for Architecture Work
- Statement of Architecture Work
- Architecture Vision
- Relevant technical requirements that will apply to this phase
- Gap analysis (from Business Architecture)
- Business Baseline - Version 2 (detailed) - if appropriate
- Target Business Architecture - Version 2 (detailed)
- Re-usable building blocks (from organization's Enterprise Continuum, if available)
- Data Baseline Description - if appropriate, and if available
- Target Data Architecture - if available

Steps

1. **Applications Baseline Description.** Develop a baseline description of the existing Applications architecture, to the extent necessary to support the target Applications Architecture. The scope and level of detail to be defined will depend on the extent to which existing Application components are likely to be carried over into the target Applications architecture, and on whether existing architectural descriptions exist, as describe under Approach, above. Define for each application:
    - Name (short and long)
    - Who maintains
    - Owner(s) / business unit(s) responsible for requirements
    - Other users
    - Plain language description of what the application does (not how it does it)
    - Status (planned, operational, obsolete)
    - Business functions supported
    - Organizational units supported
    - Hardware / software platform(s) on which it runs
    - Networks used
    - Precedent and successor applications

    To the extent possible, identify the relevant Applications architecture building blocks, drawing on the Architecture Continuum.

2. **Principles, Reference Models, Viewpoints and Tools:**
    i. Review and validate (or generate, if necessary) the set of Applications principles.
        - These will normally form part of an overarching set of Architecture Principles. Guidelines for developing and applying principles, and a sample set of Applications principles, are given in Part IV, Architecture Principles.

    ii. Select relevant Applications architecture resources (reference models, patterns, etc.) from the Architecture

Continuum, on the basis of the business drivers, and the stakeholders and concerns.

iii. Select relevant Applications architecture viewpoints (for example, stakeholders of the Applications - viewpoints relevant to functional and individual users of applications, Software Engineering View, Application-to-application Communication View, Software Distribution View, Enterprise Manageability View, etc.) - i.e., those that will enable the architect to demonstrate how the stakeholder concerns are being addressed in the Applications Architecture.

iv. Identify appropriate tools and techniques to be used for capture, modeling, and analysis, in association with the selected viewpoints. Depending on the degree of sophistication warranted, these may comprise simple documents or spreadsheets, or more sophisticated modeling tools and techniques.

- Consider using platform-independent descriptions of business logic. For example, the OMG's Model-Driven Architecture offers an approach to modeling application architectures that preserves the business logic from changes to the underlying platform and implementation technology.

3. **Architecture Model(s)**.

i. For each viewpoint, create the model for the specific view required, using the selected tool or method. Examples of Applications models are:

- the TeleMangement Forum (TMF) has developed detailed applications models relevant to the Telecommunications Industry;
- the Object Management Group (OMG) has a number of vertical Domain Task Forces developing software models relevant to specific domains such as Healthcare, Transportation, Finance, etc.

ii. Assure that all stakeholder concerns are covered. If they are not, create new models to address concerns not covered, or augment existing models (see above). Model at least the following:

- Common Applications services view - both those being consumed, and those being produced for others to consume.
- Applications Interoperability view, assumptions, dependencies, and standards (an example is the LISI Interoperability Model). Also The Open Group has a Reference Model for Integrated Information Infrastructure that can be used as a basis for this.

iii. Relate the application systems to the business functions in the Business Architecture.

i. For each application, identify each lowest level business function that it supports in the Business Architecture.

ii. Generate Application-Business Function matrices tabulating all the relationships.

iii. Review and validate the Application-Business Function matrices. In cases where a business function is supported by more than one application, check for any unnecessary duplication of functionality (and if found, eliminate it by redefining the applications concerned).

iv. Identify any business functions not supported by an applicaiton, and rationalize: either provide application support by amending or updating the set of application definitions, iterating steps 1 - 3; or else document the reason why no application support is warranted.

v. Use the Application-Business Function matrices generated under step (ii) above, and the business function-to-organizational-unit mappings (business functions cross-linked to the organizational units that perform them) contained in the Business Architecture, to relate the application systems to the organizational units that they support.

iv. Ensure that all information requirements in the Business Architecture are met.

v. Perform trade-off analysis to resolve conflicts (if any) among the different views.

- One method of doing this is CMU / SEI's Architecture Trade-off Analysis (ATA) Method.

vi. Validate that the models support the principles, objectives and constraints.

vii. Note changes to the viewpoint represented in the selected models from the Architecture Continuum, and document.

viii. Test architecture models for completeness against requirements.

4. **Identify candidate application systems.**

1. Review the re-usable architecture building blocks and re-usable solution building blocks from the enterprise's Architecture Continuum, the Business Scenario description, and the Baseline Description, and list all the potential application systems.
2. If available, review the Entity-to-business-function matrices from the Data Architecture, and identify potential applications to perform the required data management functions, and/or to automate particular business functions.
3. Even if a complete Data Architecture is not available, review whatever lists of data exist
4. Develop a user location / applications matrix
5. Brainstorm other potential application systems based on innovative use of new developments in technology.
6. Merge all lists into a single, deduplicated list of candidate application systems, including for each a brief description of business function(s) supported and data / information managed.
7. Create application definitions for all candidate application systems. For each application:

i. Assign a unique name and identifier.

ii. Write a brief description of what the application does (not how it works).

iii. Write a brief description of the business benefits arising from the application.

iv. Simplify complicated applications by decomposing them into two or more applications.

v. Ensure that the set of application definitions is internally consistent, by removing duplicate functionality as far as possible, and combining similar applications into one.

vi. Identify technology requirements and candidate technology building blocks, where this affects the

applications design, including re-usable solution building blocks from the enterprise's Architecture Continuum, and external software packages.
   vii. Identify any critical infrastructure dependencies (e.g., operating system and network services required).
   viii. Identify any critical application dependencies, and minimize as far as possible.
   ix. Time permitting, relate the applications to the files and databases described in the Baseline Description, and/or to the data entities defined in the Data Architecture (if available).
   x. Time permitting, draw simple diagrams to illustrate views of the applications architecture relevant to different stakeholders.
5. **Conduct a formal checkpoint review** of the architecture model and building blocks with stakeholders.
   - Review the Application-Business Function matrices generated in step 3, and the Business Architecture generated in Phase B.
6. **Review the qualitative criteria** (e.g., security, availability, performance, costs), providing as many measurable criteria as possible (e.g., privacy / confidentiality, reliability, minimum tolerable outages, cycle requirements and transaction volume requirements at peak and mean times, numbers and locations of users, etc.). Use to specify required service levels for Applications services (for example, via formal Service Level Agreements).
   - The goal here is to guide the Data and Technology Architecture efforts as to the qualities required in the data, and the underlying technology, that support and are processed by the application.
7. **Complete the Applications architecture.**
   i. Select standards for each of the architecture building blocks, reusing as much as possible from the reference models selected from the Architecture Continuum.
   ii. Fully document each architecture building block.
   iii. Final cross check of overall architecture against business requirements. Document rationale for building block decisions in architecture document.
   iv. Document final requirements traceability report.
   v. Document final mapping of the architecture within the Architecture Continuum. From the selected architecture building blocks, identify those that might be reused, and publish via the architecture repository.
   vi. Document rationale for building block decisions in architecture document.
   vii. Prepare Applications Architecture Report. Generate the Applications Architecture document. If appropriate, use reports and/or graphics generated by modeling tools to demonstrate key views of the architecture. Route the Applications Architecture document for review by relevant stakeholders, and incorporate feedback.
   viii. Checkpoint/Impact analysis: Check the original motivation for the architecture project and the Statement of Architecture Work against the proposed Applications Architecture**.** Conduct an impact analysis, to:
      i. Identify any areas where the Business Architecture (e.g., business practices) may need to change to cater for changes in the Applications Architecture (for example, changes to forms or procedures, application systems or database systems).
         - If the impact is significant, this may warrant the Business Architecture being revisited.
      ii. Identify any areas where the Data Architecture (if generated at this point) may need to change to cater for changes in the Applications Architecture (or to identify constraints on the Data Architecture, if about to be designed).
         - If the impact is significant, this may warrant the Data Architecture being revisited, if already developed in this cycle).
      iii. Identify any constraints on the Technology Architecture about to be designed.
      iv. Refine the proposed Applications Architecture only if necessary.
8. **Gap analysis** (see under Approach, above) and report
   i. Create gap matrix as described above.
   ii. Identify building blocks to be carried over, classifying as either changed or unchanged.
   iii. Identify eliminated building blocks.
   iv. Identify new building blocks.
   v. Identify gaps and classify as those that should be developed and those inherited.

Outputs

The outputs of this phase are:

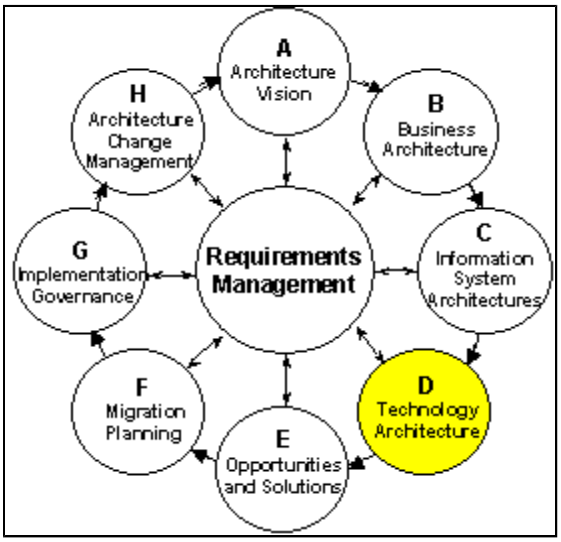
- Statement of Architecture Work (updated if necessary)
- Applications Baseline Description - if appropriate
- Validated Applications Principles, or new Applications Principles (if generated here)
  - ZF: Scope/Data
- Target Applications Architecture
  - Process Systems Model
  - Place Systems Model
  - Time Systems Model
  - People Systems Model
  - Applications interoperability requirements
- Viewpoints addressing key stakeholder concerns.
- Views corresponding to the selected viewpoints; e.g.:

    - Common Applications services view
    - Applications Interoperability view
    - Applications / Information View
    - Applications / User locations View
- Gap analysis results
- Applications Architecture Report, summarizing what was done and the key findings
- Impact Analysis
    - Areas where the Business Architecture may need to change to cater for changes in the Applications Architecture
    - Identify any areas where the Data Architecture (if generated at this point) may need to change to cater for changes in the Applications Architecture.
    - Constraints on the Technology Architecture about to be designed.
- Updated business requirements (if appropriate)

Phase D: Technology Architecture (Introduction)

Objective Approach Inputs Steps Outputs

Objective



The objective of this phase is to develop a technology architecture that will form the basis of the following implementation work.

Approach

General

Detailed guidelines for this phase, including Inputs, Steps, and Outputs, are given under Target Technology Architecture - Detailed Description.

Architecture Continuum

As part of this Phase, the architecture team will need to consider what relevant technology architecture resources are available in the Architecture Continuum.

In particular:

- The TOGAF Technical Reference Model
- Generic technology models relevant to your organization's industry "vertical" sector. For example:
  - the TeleManagement Forum (TMF) has developed detailed technology models relevant to the Telecommunications Industry;
- Technology models relevant to common systems architectures. For example:
  - The Open Group has a Reference Model for Integrated Information Infrastructure that focuses on the application-level components and underlying services necessary to provide an integrated information infrastructure.

Inputs

Inputs to this phase are:

- Technical Principles (if existing)
- Request for Architecture Work
- Statement of Architecture Work
- Architecture Vision
- Relevant technical requirements from previous phases
- Gap analysis (from Data Architecture)
- Gap analysis (from Applications Architecture)
- Business Baseline - Version 2 (detailed) - if appropriate
- Data Baseline Description - if appropriate
- Applications Baseline Description - if appropriate
- Target Business Architecture - Version 2 (detailed)
- Re-usable building blocks (from organization's Enterprise Continuum, if available)
- Target Data Architecture
- Target Applications Architecture

Steps

1. **Technology Baseline Description.**
   i. **Review Business Baseline**, **Data Baseline and Application Systems Baseline**, to the degree necessary to inform decisions and subsequent work.
   ii. Develop a baseline description of the existing Technology Architecture, to the extent necessary to support the

target Technology Architecture. The scope and level of detail to be defined will depend on the extent to which existing Technology components are likely to be carried over into the target Technology architecture, and on whether existing architectural descriptions exist, as described under Approach, above. Define for each major hardware or software platform type:

- Name (short and long)
- Physical location
- Owner(s)
- Other users
- Plain language description of what the hardware / software platform is and what it is used for
- Business functions supported
- Organizational units supported
- Networks accessed
- Applications and data supported
- System inter-dependencies (for example, fall-back configurations)

iii. To the extent possible, identify and document candidate technology architecture building blocks (potential re-usable assets).

iv. Draft the Technology Baseline Report: summarize key findings and conclusions, developing suitable graphics and schematics to illustrate baseline configuration(s). If warranted, provide individual Technology Baseline Descriptions as Annexes.

v. Route the Technology Baseline Report for review by relevant stakeholders, and incorporate feedback. Refine the Baseline Description only if necessary.

2. **Target Technology Architecture.** See detailed steps. Detailed activities for this step, including Inputs, Activities, and Outputs, are given under Target Technology Architecture - Detailed Description.

Outputs

The outputs of this phase are:

- Statement of Architecture Work (updated if necessary)
- Technology Baseline Description - if appropriate
- Validated Technology Principles, or new Technology Principles (if generated here)
- Technology Architecture Report, summarizing what was done and the key findings
- Target Technology Architecture Version 1
- Technology Architecture - gap report
- Viewpoints addressing key stakeholder concerns.
- Views corresponding to the selected viewpoints.

Target Technology Architecture - Introduction

Introduction

This is the detailed description of the process to develop the Target Technology Architecture.

Overview

The steps involved in development of the Target Technology Architecture are illustrated in Figure 2 below.

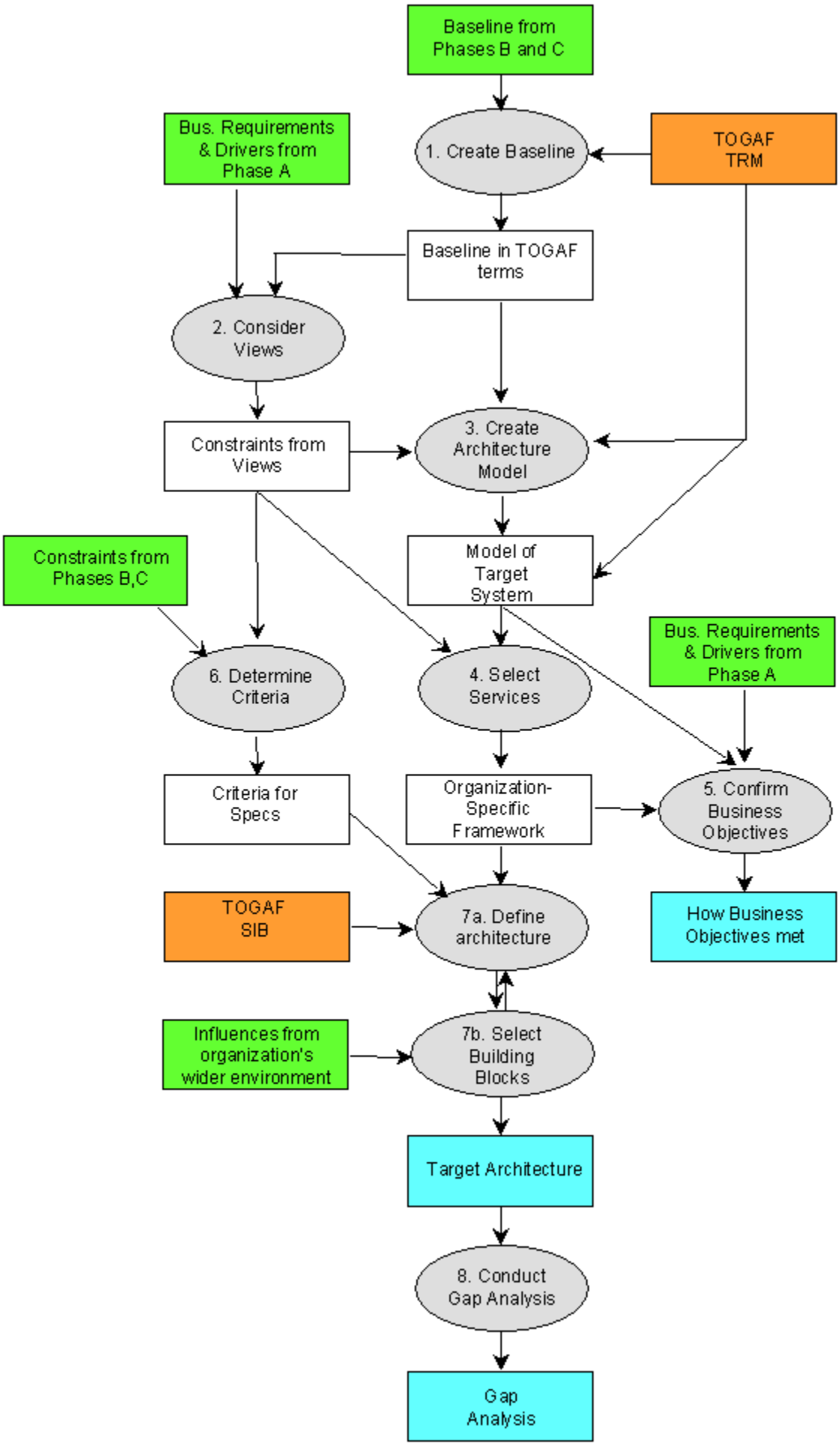

Baseline from Phases B and C
Bus. Requirements & Drivers from Phase A
1. Create Baseline
TOGAF TRM
Baseline in TOGAF terms
2. Consider Views
Constraints from Views
3. Create Architecture Model
Constraints from Phases B,C
Model of Target System
Bus. Requirements & Drivers from Phase A
6. Determine Criteria
4. Select Services
Criteria for Specs
Organization-Specific Framework
5. Confirm Business Objectives
TOGAF SIB
7a. Define architecture
How Business Objectives met
Influences from organization's wider environment
7b. Select Building Blocks
Target Architecture
8. Conduct Gap Analysis
Gap Analysis

Figure 2: Technology Architecture Development

The boxes and ovals in Figure 2 are color-coded as follows:

- **green** boxes represent inputs to the Technology Architecture development process from earlier phases in the architecture development cycle. This implies that some previous implementation exists for which an architecture may or may not have been defined previously.
- **orange** boxes represent inputs to the process from TOGAF.
- white boxes represent inputs and outputs internal to the Technology Architecture development process.
- **blue** boxes represent outputs to later phases in the architecture development cycle.
- **gray** ovals represent phases within the Technology Architecture development process.

It is possible that an organization creating or adapting a Technology Architecture already mandates the use of a list of approved suppliers / products for that organization. Such a list would be an input to the definition of the organization specific architecture framework, as shown in Figure 2. The architectures can then be used as procurement tools to govern future growth and the development of the organization's IT infrastructure.

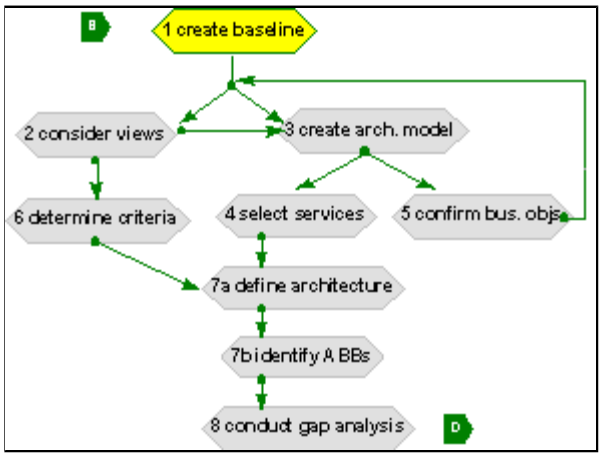
The steps outlined in Figure 2 are expanded in the following subsections.

Step 1: Create baseline description in TOGAF format

1. Create a baseline description in the TOGAF format

Objective Approach Inputs Activities Outputs

Objective



The objective of this step is to convert the description of the existing system into services terminology using the organization's foundation architecture (e.g., the TOGAF Foundation Architecture's Technical Reference Model). The rationale behind this is to structure the existing system description in a way which makes it compatible with the breakdown of standards and the descriptions used within your foundation architecture.

Approach

This step is intended to facilitate moving from product documentation to a service-oriented description. The step will aid in specifying standards for the target architecture in step 4. An additional step, step 3, oriented to defining building blocks, provides the means to cross check the architectural definition process in the form of implementation related decisions.

Additionally this step captures relevant parts of the existing architecture (using the scope definition established in Phase A) as candidates for re-usable building blocks, along with inhibitors to meeting business requirements using the existing system. The existing architecture is assessed against the business architecture, identifying the key inhibitors and opportunities for re-use. Finally the existing architecture assessment ends with the capture of implied or explicit architecture principles that should be carried forward and imposed on this architecture exercise.

Begin by converting the description of the existing environment into the terms of your organization's foundation architecture (e.g., the TOGAF Foundation Architecture's Technical Reference Model). This will allow the team developing the architecture to gain experience with the model and to understand its component parts. The team may be able to take advantage of a previous architectural definition, but it is assumed that some adaptation may be required to match the architectural definition techniques described as part of this process. Another important task is to set down a list of key questions which can be used later in the development process to measure the effectiveness of the new architecture.

A key process in the creation of a broad architectural model of the target system is the conceptualization of architectural building blocks (ABBs). ABBs are not intended to be solutions, but depictions of how the architecture might be looked on in implementable terms. Their functionality is clearly defined, but without the detail introduced by specific products. The method of defining ABBs, along with some general guidelines for their use in creating an architectural model, is described in Part IV under Building Blocks and the ADM, and is illustrated in detail in the Building Blocks Example in Part IV.

It is recommended that architecture building blocks be documented (e.g., with an architecture description language) and stored (e.g., in a repository or information base), in order to maximize reuse potential.

Applying the ABB method introduces application space into the architectural process. This is the means of linking services, which address functionality that must be considered on an enterprise basis, with applications, which may or may not address global functionality. The Building Blocks Example in Part IV provides insight into both application specific and more global considerations in defining building blocks in order to illustrate this.

Inputs

The inputs to this step are:

- Technical Principles (if existing)
- Request for Architecture Work
- Statement of Architecture Work
- Architecture Vision
- Relevant technical requirements from previous phases
- Gap analysis (from Data Architecture)
- Gap analysis (from Applications Architecture)
- Business Baseline - Version 2 (detailed) - if appropriate

- Data Baseline Description - if appropriate
- Applications Baseline Description - if appropriate
- Business Architecture - Version 2 (detailed)
- Re-usable building blocks (from organization's Enterprise Continuum, if available)
- Target Data Architecture
- Target Applications Architecture
- Re-usable architecture building blocks (from organization's Architecture Continuum, if available)
- Re-usable solutions building blocks (from organization's Solutions Continuum, if available)

Activities

Key activities in this step include:

1. Collect data on current system.
2. Document all constraints
3. Review and validate (or generate, if necessary) the set of Technology Architecture principles.
    - These will normally form part of an overarching set of Architecture Principles. Guidelines for developing and applying principles, and a sample set of Technology Architecture principles, are given in Part IV, Architecture Principles.
4. List distinct functionality
5. Produce affinity groupings of functionality using TOGAF TRM service groupings (or your business' foundation architecture)
6. Analyze relationships between groupings
7. Sanity check functionality to assure all of current system considered
8. Identify interfaces
9. Produce Technology Architecture model
10. Verify Technology Architecture model
11. Document key questions to test merits of Technology Architecture
12. Document criteria for selection of service portfolio architecture

Outputs

The outputs of this step are:

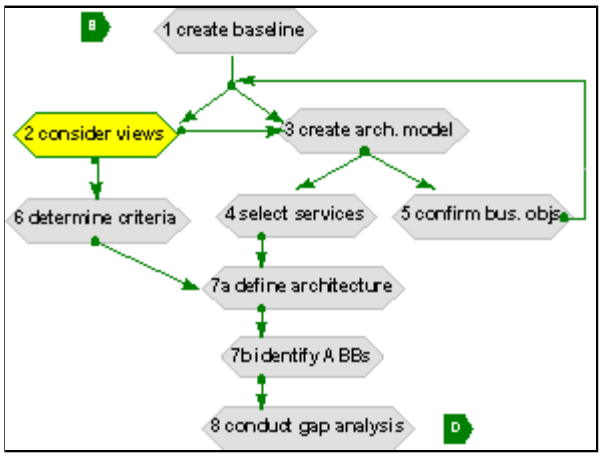
- Technical Principles (if not existing)
- Technology Architecture Version 0.1:
    - Technology Architecture - Constraints
    - Technology Architecture - Architecture Principles
    - Technology Architecture - Requirements Traceability - key questions list
    - Technology Architecture - Requirements Traceability - criteria for selection of service portfolio
    - Technology Architecture Model - Version 0.1

Step 2: Consider Views

2. Consider different architecture reference models, viewpoints, and tools

Objective Approach Inputs Activities Outputs

Objective



The objective of this step is to perform an analysis of the Technology Architecture from a number of different concerns (requirements) or viewpoints and to document each relevant viewpoint. The purpose of considering these viewpoints is to ensure that all **relevant** stakeholder concerns will have been considered in the final Technology Architecture, so ensuring that the target system will meet all the requirements put on it.

Approach

The Business Architecture is used to select the most relevant viewpoints for the project. It is important to recognize that in practice it will be rarely possible to reach 100% coverage of stakeholder concerns.

Pertinent viewpoints are created first from the existing system to identify the salient elements of the current systems requirements that the stakeholders confirm must also be satisfied in the target system. A comprehensive set of stakeholder viewpoints must also be created for the target system. The corresponding views of the existing system will be compared with the views of the target system to identify elements of the existing system that are intended for replacement or improvement.

If a set of viewpoints is carefully chosen, it will expose the most important aspects of the existing architecture and the requirements of the target system.

Several different viewpoints may be useful. Architecture viewpoints and views are described in greater detail in Part IV, *Architecture Views*. The viewpoints presented there should not be considered an exhaustive set, but simply as a starting point. In developing an Technology Architecture, it is very likely that some of the viewpoints given there will not be useful, while others not given there will be essential. Again use the Business Architecture as a guide in selecting the pertinent viewpoints.

Inputs

The inputs to this step are:

- Request for Architecture Work
- Statement of Architecture Work
- Business Architecture Version 2
- Technology Architecture Version 0.1

Activities

Key activities in this step include:

1. **Select relevant Technology Architecture resources** (reference models, patterns, etc.) from the Architecture Continuum, on the basis of the business drivers, and the stakeholders and concerns.
2. **Select relevant Technology Architecture viewpoints** - i.e., those that will enable the architect to demonstrate how the stakeholder concerns are being addressed in the Technology Architecture. (See Part IV, Architecture Views, for examples).
    - Document the selected viewpoints, if not already documented.
        - Consider using ANSI/IEEE Std. 1471-2000 as a guide for documenting a viewpoint.
    - A primary reference model will be the TOGAF Technical Reference Model. Other Reference Models will be taken from the Architecture Continuum.
    - Consider developing at least the following views:
        - Networked Computing/ Hardware View
        - Communications View
        - Processing View

        - Cost View
        - Standards View
    - Brainstorm and document technical constraints deriving from analysis of the concerns, and ensure they are covered by the viewpoints.
3. **Identify appropriate tools and techniques** to be used for capture, modeling, and analysis, in association with the selected viewpoints. Depending on the degree of sophistication warranted, these may comprise simple documents or spreadsheets, or more sophisticated modeling tools and techniques
4. **Perform trade-off analysis** to resolve conflicts (if any) among the different viewpoints.
    - One method of doing this is CMU / SEI's Architecture Trade-off Analysis (ATA) Method.

Outputs

The outputs of this step are:

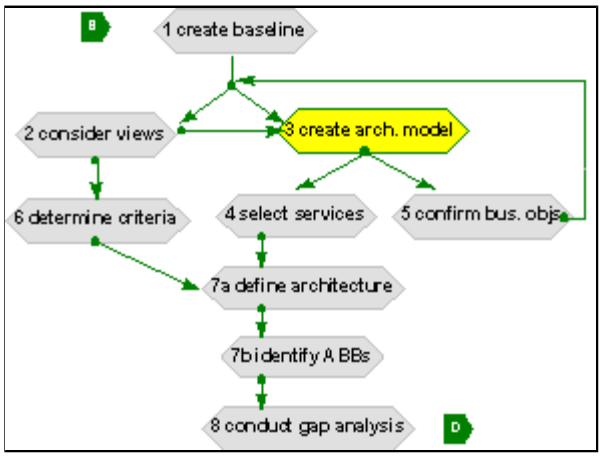
- Technology Architecture Version 0.2
    - Technology Architecture - Architecture Viewpoints
        - Networked Computing/Hardware View
        - Communications View
        - Processing View
        - Cost View
        - Standards View
    - Technology Architecture - Constraints

Step 3: Create architectural model of building blocks

3. Create an architectural model of building blocks

Objective Approach Inputs Activities Outputs

Objective



The reason for selecting viewpoints in the previous step is to be able to develop views for each of those viewpoints in this step. The architectural model created in this step comprises those several views.

The objective of this step is to broadly determine how the services required in the target system will be grouped after considering all pertinent viewpoints of the architecture's use. This differs from step 1 in that step 1 dealt mainly with the required functionality of the system, whereas here we are considering many viewpoints that are not expressed explicitly as required functionality.

The rationale behind this is to enable the services required within the system to be selected during the next step, through the creation of an architecture model that clearly depicts the required services.

Approach

At this step, the purpose of examining different viewpoints in step 2 becomes clear. The constraints defined and the unique system insights gained through an examination of the viewpoints pertinent to the current system and the target system can be used to validate the ability of the broad architectural model to accommodate the system requirements.

The broad architectural model starts as a TOGAF TRM-based model (or a model based upon the organization's Foundation Architecture), derived from the service to function mapping carried out as part of the service examination in step 1. An architecture based exactly on the TOGAF TRM may not be able to accommodate the stakeholder needs of all organizations. If the examination of different viewpoints identifies architectural features that cannot be expressed in terms of the TOGAF TRM, changes and amendments to the TOGAF TRM should be made to create an organization-specific TRM.

Once the baseline description has been established and appropriate views described, it is possible to make decisions about how the various elements of system functionality should be implemented. This should only be in broad terms, to a level of detail which establishes how the major business functions will be implemented - for example as a transaction processing application or using a client/server model.

Therefore this step defines the future model of Building Blocks (e.g. collections of functions and services generated from previous steps). It is here that reuse of building blocks from your business' architecture continuum is examined carefully, assuring that maximum reuse of existing material is realized.

Once the architecture model of building blocks is created, the model must be tested for coverage and completeness of the required technical functions and services. For each building block decision, completely follow through its impact and note the rationale for decisions, including the rationale for decisions not to do something.

Inputs

The inputs to this step are:

- Business Architecture Version 2
- Technology Architecture Version 0.2
  - Technology Architecture - Viewpoints
  - Technology Architecture - Constraints
- Re-usable architecture building blocks (from organization's Architecture Continuum, if available)

Activities

Key activities in this step include:

1. To the extent possible, identify the relevant Technology Architecture building blocks, drawing on the Architecture Continuum.
2. For each viewpoint, create the model for the specific view required, using the selected tool or method. Consider developing at least the following views:
    - Networked Computing/ Hardware View
    - Communications View
    - Processing View
    - Cost View
    - Standards View
3. Assure that all stakeholder concerns are covered. If they are not, create new models to address concerns not covered, or augment existing models.
4. Ensure that all information requirements in the Business Architecture, Data Architecture, and Applications Architecture are met.
5. Perform trade-off analysis to resolve conflicts (if any) among the different views.
    - One method of doing this is CMU / SEI's Architecture Trade-off Analysis (ATA) Method.
6. Validate that the models support the principles, objectives and constraints.
7. Note changes to the viewpoint represented in the selected models from the Architecture Continuum, and document.
8. Identify solution building blocks that would be used to implement the system, and create a model of building blocks.
9. Check building blocks against existing library of building blocks and re-use as appropriate.
10. Test architecture models for completeness against requirements.
11. Document rationale for building block decisions in architecture document.

Outputs

The outputs of this step are:

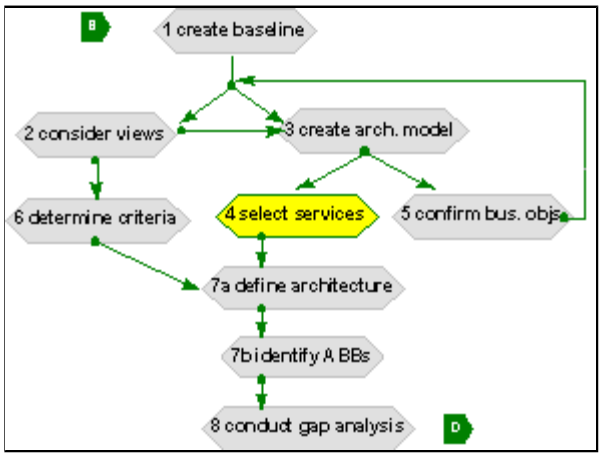
- Technology Architecture Version 0.3
    - Technology Architecture Model
        - Networked Computing/ Hardware View
        - Communications View
        - Processing View
        - Cost View
        - Standards View
    - Technology Architecture - change requests and/or extensions or amendments to be incorporated in an organization-specific architecture continuum.

Step 4: Select services portfolio per building block

4. Select the services portfolio required per building block

Objective Approach Inputs Activities Outputs

Objective



The objective of this step is to select services portfolios for each building block generated in the preceding step.

Approach

The services portfolios are combinations of basic services from the service categories in the TOGAF Technical Reference Model that do not conflict. The combination of services are again tested to ensure support for the applications. This is required as a pre-requisite to the later step of defining the architecture fully.

The constraints output from step 2 can provide more detailed information about:

- requirements for organization-specific elements or pre-existing decisions (as applicable)
- pre-existing and unchanging organizational elements (as applicable)
- inherited external environment constraints

Where requirements demand definition of specialized services that are not identified in TOGAF, consideration should be given to how these might be replaced if standardized services become available in the future.

For each Architectural Building Block build up a service description portfolio as a set of non-conflicting services. The set of services must be tested to ensure that the functionality provided meets application requirements.

Inputs

The inputs to this step are:

- Business Architecture Version 2
- Technology Architecture Version 0.3
- TOGAF Technical Reference Model
- TOGAF Standards Information Base

Activities

Key activities in this step include:

1. Produce affinity grouping of services
2. Cross-check affinity groups against needs
3. Document service description portfolio for each Architectural Building Block cross-checking for non-conflicting services
4. Document change requests to architectures in architecture continuum

Outputs

The outputs of this step are:

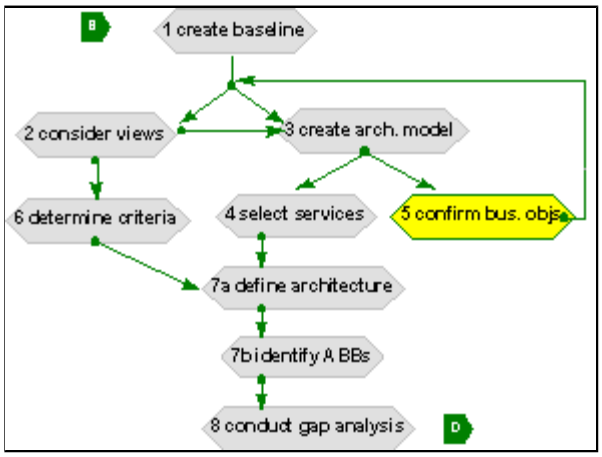
- Technology Architecture Version 0.4
  - Technology Architecture - target services (a description of the service portfolios required also known as an Organization Specific Framework)
  - Technology Architecture - change requests and/or extensions or amendments to be incorporated in an organization-specific architecture continuum

Step 5:Confirm business goals and objectives are met

5. Confirm that the business goals and objectives are met

Objective Approach Inputs Activities Outputs

Objective



The objective of this step is to clarify and check the business goals and other objectives of implementing the architecture. This is required as a cross check that the Technology Architecture meets these objectives.

Approach

The key question list is used to pose questions against the architecture model and service description portfolio to test its merit and completeness.

Inputs

The inputs to this step are:

- Business Architecture (business goals) Version 2
- Technology Architecture Version 0.4

Activities

Key activities in this step include:

1. Conduct a formal checkpoint review of the architecture model and building blocks with stakeholders, validating that business goals are met. Utilizing the key questions list, ensure that the architecture addresses each question.

2. Document findings

Outputs

The outputs of this step are:

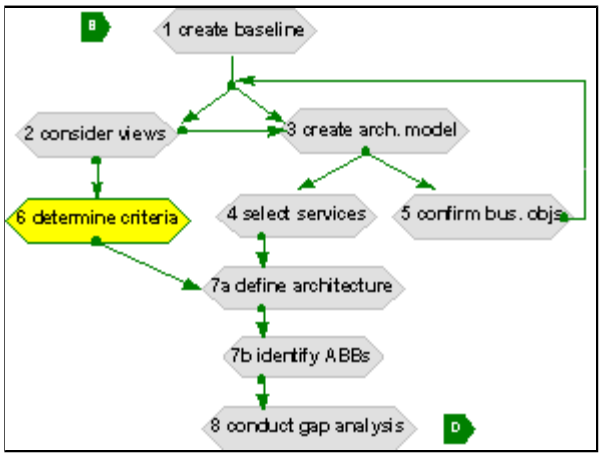
- Technology Architecture Version 0.5
  - Technology Architecture - Requirements Traceability (business objectives criteria)

Step 6: Determine criteria for specification selection

6. Determine criteria for specification selection

Objective Approach Inputs Activities Outputs

Objective



The objective of this step is to develop a set of criteria for choosing specifications and portfolios of specifications.

Approach

Choosing the right criteria is vital if the final architecture is to meet its objectives. These criteria will depend on the existing system and the overall objectives for the new architecture. The overall objectives should be developed from the organization's business goals, so it is hard to give specific advice here, but some example objectives are listed in Part IV, Business Scenarios.

Here are some example criteria, selected by a large government organization with the intention of building a stable and widely applicable architecture:

A standard or specification:

- Must meet the organization's requirements
- Must meet legal requirements
- Should be a publicly available specification
- Should have been developed by a process which sought a high level of consensus from a wide variety of sources
- Should be supported by a range of readily available products
- Should be complete
- Should be well understood, mature technology
- Should be testable, so that components or products can be checked for conformance
- Should support internationalization
- Should have no serious implications for ongoing support of legacy systems
- Should be stable
- Should be in wide use
- Should have few, if any problems or limitations

A high level of consensus is often considered the most important factor by large organizations because standards and specifications chosen have to accommodate a wide range of user needs. For example, in determining the level of consensus for standards in their architecture, the Application Portability Profile or APP, the US National Institute for Standards and Technology prefers to use international standards for the basis of specifications. The process through which these international standards have evolved requires a very high level of consensus. A number of US Federal Information Processing Standards (FIPS) specified in the APP are based on approved international standards. The use of international standards has significant benefits for any organization which works or trades with organizations in other countries.

Inputs

The inputs to this step are:

- Business Architecture Version 2
- Technology Architecture Version 0.5
- Standards Information Base

Activities

Key activities in this step include:

1. Brainstorm criteria for choosing specifications and portfolios of specifications relying on previously used criteria for existing system and extrapolating for new architectural elements.

2. Meet with sponsors and present current state to negotiate a continue request from sponsors.

Outputs

The outputs of this step are:

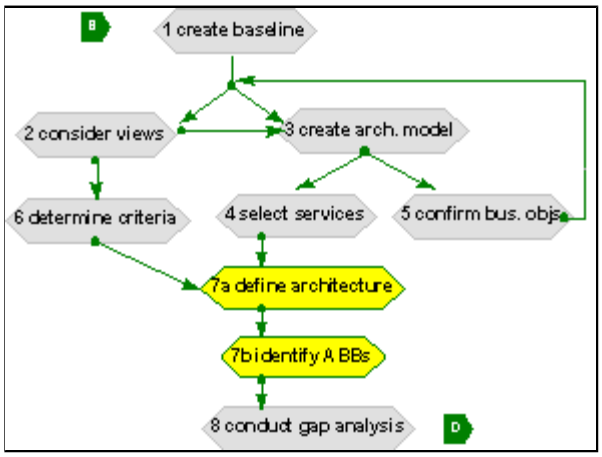
- Technology Architecture Version 0.6
  - Technology Architecture - Requirements Traceability (standards selection criteria)

Step 7: Complete architecture definition

7. Complete the architecture definition

Objective Approach Inputs Activities Outputs

Objective



The objective of this step is to fully specify the Technology Architecture. This is a complex and iterative process in which the selection of building blocks and interfaces has a big impact on how the original requirements are met. Figure 2 shows this as two boxes, captioned as steps 7a and 7b, but in reality the process is more complicated. See Part IV, Building Block Example, for further details.

Approach

Completion of the architecture definition may be achieved in two steps, by defining an intermediate Transition Architecture in addition to the final target architecture, if complexity of migration requires it.

The specification of building blocks as a portfolio of services is an evolutionary process:

- The earliest building block definitions start as relatively abstract ones, defined by standards and services that map most easily to the architectural framework. These building blocks are most probably architectural building blocks.
- At this stage a model and a portfolio of services have been established. The next step is to select the set of specifications that provide the services and that can be combined as required to create the building blocks.
- During this final step in the development of building blocks it must be verified that the organization-specific requirements will be met. The development process must include recognition of dependencies and boundaries for functions and should take account of what products are available in the marketplace. There are architectural and related solution oriented building blocks.
- An example of how this might be expressed can be seen in the Building Blocks Example in Part IV. Building blocks can be defined at a number of levels matching the degree of integration that best defines the architecture of the system at any stage.
    - Fundamental Functionality and Attributes - semantic, unambiguous including security capability and manageability
    - Interfaces - chosen set, supplied (APIs, data formats, protocols, hardware interfaces, ...standards)
    - Dependent BBs with required functionality and named used interfaces
    - Map to business / organizational entities and policies
- Finally the building blocks become more implementation specific as Solution Building Blocks and their interfaces become the detailed architecture specification. Solution Building Blocks are a means to determining how portions of the target architecture might be procured, developed, or reused. The Solution Building Blocks architecture should have separate elements for developed, reused and procured building blocks, each described in terms of their minimum specification.

A full list of standards and specifications recommended by The Open Group can be found in Part III, Standards Information Base.

Inputs

The inputs to this step are:

- Business Architecture Version 2
- Technology Architecture Version 0.6
- Re-usable architecture building blocks (from organization's Architecture Continuum, if available)
- TOGAF Standards Information Base

Activities

Key activities in this step include:

1. Ensure clear documentation of all interfaces for each building block (APIs, data formats, protocols, hardware interfaces).
2. Select standards for each of the architecture building blocks, reusing as much as possible from the reference models selected from the Architecture Continuum.
3. Fully document each architecture building block.
4. Final cross check of overall architecture against business requirements. Document rationale for building block decisions in architecture document.
5. Document final requirements traceability reports
6. Document final mapping of the architecture within the Architecture Continuum. From the selected architecture building blocks, identify those that might be reused, and publish via the architecture repository.
7. Document rationale for building block decisions in architecture document.
8. Generate the Technology Architecture document.
9. Prepare Technology Architecture Report. If appropriate, use reports and/or graphics generated by modeling tools to demonstrate key views of the architecture. Route the Technology Architecture document for review by relevant stakeholders, and incorporate feedback.
10. Checkpoint/Impact analysis: Check the original motivation for the architecture project and the Statement of Architecture Work against the proposed Technology Architecture. Conduct an impact analysis, to:
    i. Identify any areas where the Business Architecture (e.g., business practices) may need to change to cater for changes in the Technology Architecture.
        - If the impact is significant, this may warrant the Business Architecture being revisited.
    ii. Identify any areas where the Data Architecture may need to change to cater for changes in the Technology Architecture.
        - If the impact is significant, this may warrant the Data Architecture being revisited.
    iii. Identify any areas where the Applications Architecture may need to change to cater for changes in the Technology Architecture.
        - If the impact is significant, this may warrant the Applications Architecture being revisited.
    iv. Refine the proposed Technology Architecture only if necessary.

Outputs

The outputs of this step are:

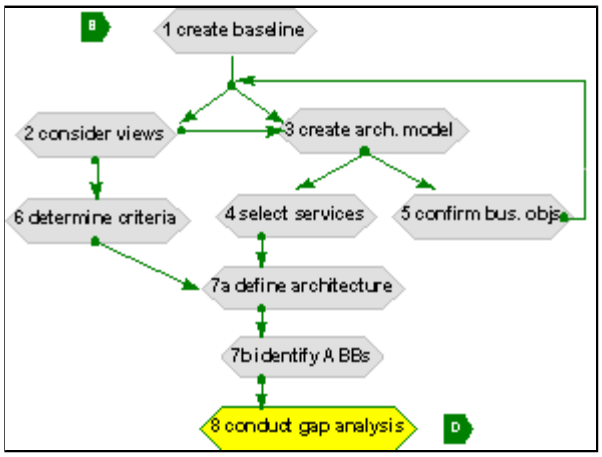
- Technology Architecture Version 0.7
    - Technology Architecture - Architecture Specification
    - Technology Architecture - Requirements Traceability
    - Technology Architecture - Mapping of the architectures in the architecture continuum.
- Technology Architecture Report

Step 8: Conduct gap analysis

8. Conduct a gap analysis

Objective Approach Inputs Activities Outputs

Objective



The objective of this step is to identify areas of the current and target system for which provision has not been made in the Technology Architecture. This is required in order to identify projects to be undertaken as part of the implementation of the target system.

Approach

A key step in validating an architecture is to consider what may have been forgotten. The architecture must support all of the essential information processing needs of the organization, as driven by the required applications. The most critical source of gaps that should be considered is stakeholder concerns that have not been addressed in subsequent architectural work.

Gap analysis highlights services and/or functions that have been accidentally left out, deliberately eliminated, or are yet to be developed or procured:

- Draw up a matrix with all the business functions of the current architecture on the vertical axis, and all the business functions of the target Technology architecture on the horizontal axis. In creating the matrix it is imperative to use terminology that is accurate and consistent.
- Add to the Current Architecture axis a final row labeled 'New Services', and to the Target Architecture axis a final column labeled 'Eliminated Services'.
- Where a function is available in both the current and target architectures, record this with 'Included' at the intersecting cell.
- Where a function from the current architecture is missing in the target architecture (in the example, "broadcast services" and "shared screen services"), each must be reviewed. If it was correctly eliminated, mark it as such in the appropriate 'Eliminated Services' cell. If it was not, you have uncovered an accidental omission in your new architecture that must be addressed by reinstating the function in the next iteration of the design - mark it as such in the appropriate 'Eliminated Services' cell.
- Where a function from the target architecture cannot be found in the current architecture (in the example, "mailing list services"), mark it at the intersection with the 'New' row, as a gap that needs to filled, either by developing or procuring the function.

When the exercise is complete, anything under 'Eliminated Services' or 'New Services' is a gap, which should either be explained as correctly eliminated, or marked as to be addressed by reinstating or developing/procuring the function.

Table 1 shows an example from the Network Services category when functions from the current architecture are missing from the target architecture:

Target architecture → / Current architecture ↓	Video conferencing services	Enhanced telephony services	Mailing list services	Eliminated services ↓
Broadcast services				Intentionally eliminated
Video conferencing services	Included			
Enhanced telephony services		Potential match		
Shared screen services				Unintentionally excluded - a gap in target architecture
New →		Gap: **Enhanced** services to be developed or procured	Gap: To be developed or procured	

Table 1: A Gap Analysis Matrix

Inputs

The inputs to this step are:

- Business Architecture Version 2
- Data Architecture
- Applications Architecture
- Technology Architecture Version 0.7

Activities

Key activities in this step include:

1. Create gap matrix as described under Approach, above.
2. Identify building blocks to be carried over, classifying as either changed or unchanged.
3. Identify eliminated building blocks.
4. Identify new building blocks.
5. Identify gaps and classify as those that should be developed, those that should be procured, and those inherited.

Outputs

The output of this step is:

- Technology Architecture Version 1
  - Technology Architecture - gap report

Postscript

Postscript

The Technology Architecture development process described above includes iterations. Financial and timing constraints should explicitly limit the number of iterations within steps 1 through 8, and drive to implementation. After that, a new cycle of architecture evolution may ensue.

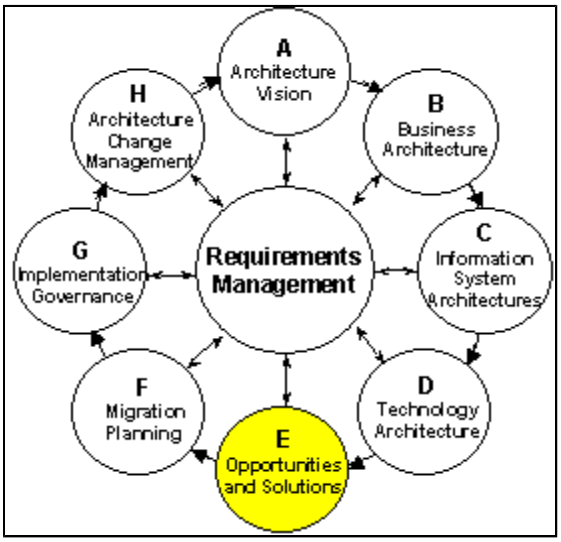
Choosing the scope of an architecture development cycle carefully will accelerate the pay-back. In contrast, an excessively large scope is unlikely to lead to successful implementation.

"How do you eat an elephant? - One bite at a time".

Phase E: Opportunities and Solutions

Objective Approach Inputs Steps Outputs

Objective



The objectives of this phase are to:

- evaluate and select among the implementation options identified in the development of the various target architectures (for example, build-versus-buy-versus-reuse options, and sub-options within those major options)
- identify the strategic parameters for change, and the top-level work packages or projects to be undertaken in moving from the current environment to the target;
- assess the dependencies, costs, and benefits of the various projects; and
- generate an overall implementation and migration strategy and a detailed implementation plan.

Approach

This phase identifies the parameters of change, the major phases along the way and the top-level projects to be undertaken in moving from the current environment to the target. The output of this phase will form the basis of the implementation plan required to move to the target architecture. This phase also attempts to identify new business opportunities arising from the architecture work in previous phases.

Sometimes the process of identifying implementation opportunities allows a business to identify new applications, and in this case it may be necessary to iterate between the Opportunities and Solutions phase and previous phases. Iteration must be limited by time or money to avoid wasting effort for the search of a perfect architecture.

The Opportunities and Solutions phase is the first phase which is directly concerned with implementation. The task is to identify the major work packages or projects to be undertaken.

An effective way to do this is to use the gap analysis on the business functions between the old environment and the new, created in Phase D. Any functions appearing as "new" items will have to be implemented (developed or purchased and deployed).

Slightly harder to identify are the projects required to update or replace existing functions which must be done differently in the new environment. One of the options to be considered here is leaving an existing system in place and coexisting with the new environment.

During this final step in the specification of building blocks it must be verified that the organization-specific requirements will be met. Key to this is reason checking against the business scenario driving the scope of this project. It is important to note that the ensuing development process must include recognition of dependencies and boundaries for functions and should take account of what products are available in the marketplace. An example of how this might be expressed can be seen in the Building Blocks Example in Part IV.

Coexistence appears on the surface to be easy. After all, the original system is left in place, largely unchanged. Unfortunately, it is not always as easy as it looks. The main problems with coexistence are:

- **user interfaces**: Combining user interfaces to the old and new applications in a single unit on the users' desks can be difficult, if not impossible.
- **access to data**: Often the new applications need to share some data with the old applications, and some kind of data

sharing must be established. This can be difficult unless the old and new systems use the same database technology.
- **connectivity**: This may involve expenditure on software and gateway equipment. In difficult cases, equipment simply may not be available in a useful time scale. Often this happens because the old system is simply too out of date for connectivity solutions to be still on the market.

The most successful strategy for the Opportunities and Solutions phase is to focus on projects that will deliver short-term pay-offs and so create an impetus for proceeding with longer-term projects.

Inputs

Inputs to this phase are:

- Request for Architecture Work
- Statement of Architecture Work
- Business Architecture
- Data Architecture
- Applications Architecture
- Technology Architecture
- Re-usable architecture building blocks (from organization's Enterprise Continuum, if available)
- Product information

Steps

Key steps in this phase include:

1. Identify the key business drivers constraining the sequence of implementation (for example, reduction of costs; consolidation of services; introduction of new customer services; etc.)
2. Review the gap analysis generated in Phase D.
3. Brainstorm technical requirements from a functional perspective
4. Brainstorm co-existence and interoperability requirements
5. Perform architecture assessment and gap analysis
6. Identify major work packages or projects, and classify as new development, purchase opportunity, or reuse of existing system.

Outputs

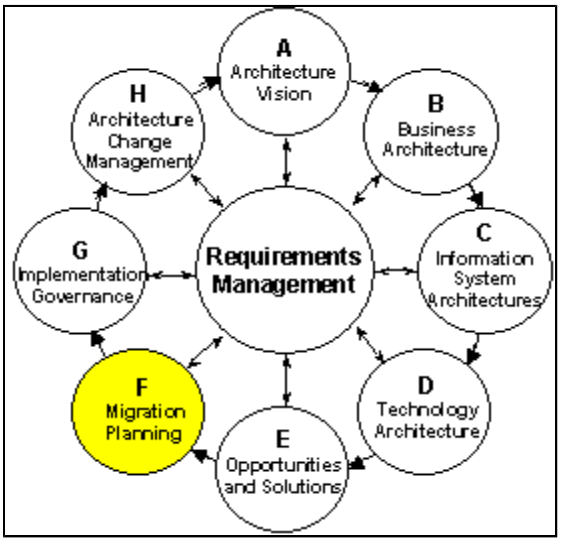
The outputs of this phase are:

- Implementation and migration strategy
- High-level implementation plan
- Impact Analysis - Project list

Phase F: Migration Planning

Objective Approach Inputs Steps Outputs

Objective



The objective of this phase is to sort the various implementation projects into priority order. Activities include assessing the dependencies, costs and benefits of the various migration projects. The prioritized list of projects will go on to form the basis of the detailed implementation and migration plan.

Approach

There are some important questions to be asked before embarking on a migration exercise:

- What are the implications of this project on other projects and activities?
- What are the dependencies between this project and other projects and activities?
- What products are needed?
- What components must be developed?
- Does the organization have the resources needed to develop such components?
- What standards are the products or components built on?
- When will they be available?
- Will the products stand the test of time, both because of the technology they use and also because of the viability of the supplier?
- What is the cost of retraining the users?
- What is the likely cultural impact on the user community, and how can it be controlled?
- What is the total cost of the migration, and what benefits will it deliver? It is important to look at actual benefits, and not presumed benefits. Is the funding available?
- Is the migration viable?

Many things affect the answers to these questions, including the current and future architectures, the size of the organization and its complexity, and the value of technology to the core functions of the organization. Other things to consider are the asset value of the current systems, and the level of risk associated with changing the solution and/or the supplier.

Most organizations find that a change of architecture has too much impact on the organization to be undertaken in a single phase. Migration often requires consideration of a number of technical issues, not the least of which are those associated with the means of introducing change to operational systems.

Issues requiring special consideration may include:

- parallel operations
- choices of proceeding with phased migration by subsystem or by function
- the impact of geographical separation on migration

The decisions resulting from these considerations should be incorporated in the implementation plan.

There are a number of strategies for developing the migration and implementation plan.

The most successful basic strategy is to focus on projects that will deliver short-term pay-offs and so create an impetus for proceeding with longer-term projects.

One common approach is to implement business functions in a data-driven chronological sequence: i.e, create the applications and supporting technology that create data before those that process the data, before those that simply store, archive or delete data.

For example, the following detailed description of this approach is taken from: SPE 68794, "Implementing Enterprise Architecture - Putting Quality Information in the Hands of Oil and Gas Knowledge Workers", G. A. Cox, R. M. Johnston, SPE, and R. M. Palermo, Aera Energy LLC, Copyright 2001, Society of Petroleum Engineers Inc.

1. Determine the future disposition of current systems. Each current system is classified as:
   - *Mainstream systems* - part of the future information system.
   - *Contain systems* - expected to be replaced or modified in the planning horizon (next 3 years).
   - *Replace systems* - to be replaced in the planning horizon.

   The current system disposition decisions should be made by business people, not IT people.

2. Applications should be combined or split into parts to facilitate sequencing and implementation. This rearrangement of applications creates a number of projects, a project being equivalent to an application or to combinations or parts of applications.

3. Develop the data sequence for the projects as described in the data architecture. Using the CRUD (Create / Read / Update / Delete) matrix developed as part of the Data Architecture, sequence the projects such that projects that create data precede projects that read or update that data.

4. Develop an estimated value to the business for each project. To do this, first develop a matrix based on a value index dimension and a risk index dimension. The value index includes the following criteria: principles compliance, which includes financial contribution, strategic alignment, and competitive position. The risk index includes the following criteria: size and complexity, technology, organizational capacity, and impact of a failure. Each of the criteria has an individual weight. The index and its criteria and weighting are developed and approved by senior management early in the project. It is important to establish the decision-making criteria before the options are known.

In addition, there will be key business drivers to be addressed that will also tend to dictate the sequence of implementation, such as:

- Reduction of costs
- Consolidation of services
- Ability to handle change
- A goal to have a minimum of “interim” solutions (they often become long-term / strategic!)

Another, possibly complementary, approach is for the individual projects or work packages to be group-sorted into a series of plateaus, each of which can be achieved in a realistic time scale.

The following description assumes a Target Architecture with only a single time horizon.

Inputs

Inputs to this phase are:

- Request for Architecture Work
- Statement of Architecture Work
- Business Architecture - Version 2
- Technology Architecture
- Impact Analysis - Project list

Steps

Key steps in this phase include:

1. Prioritize projects
2. Estimate resource requirements and availability
3. Perform cost / benefit assessment of the various migration projects
4. Perform risk assessment
5. Generate implementation roadmap (time-lined)
6. Document the Migration Plan

Outputs

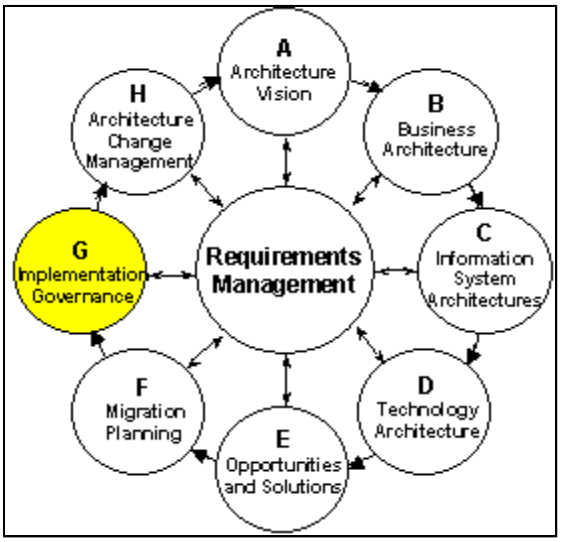
The output of this phase is:

- Impact Analysis - Detailed Implementation and Migration Plan, including
  - Architecture Implementation Contract (if appropriate)

Phase G: Implementation Governance

Objectives Approach Inputs Steps Outputs

Objective



The objectives of this phase are to:

- formulate recommendations for each implementation project,
- construct an architecture contract to govern the overall implementation and deployment process, and
- perform appropriate governance functions while the system is being implemented and deployed; in particular,
- ensure conformance with the defined architecture by implementation projects and other projects

Approach

It is here that one brings together all the information for successful management of the various implementation projects. Note that in parallel with this phase there is the execution of an organizational specific development process, where the actual development happens.

This phase establishes the connection between architecture and implementation organization, through the Architecture Contract.

Project details are developed, including:

- Name, description and objectives
- Scope, deliverables and constraints
- Measures of effectiveness
- Acceptance criteria
- Risks and issues

Implementation governance is closely allied to overall Architecture Governance, which is discussed in Part IV, Architecture Governance.

A key aspect of this phase is ensuring compliance with the defined Architecture(s), not only by the implementation projects but also by other on-going projects within the enterprise. The considerations involved with this are explained detail in Part IV, Architecture Compliance.

Inputs

Inputs to this phase are:

- Request for Architecture Work
- Statement of Architecture Work
- Re-usable solutions building blocks (from organization's Solutions Continuum, if available)
- Impact Analysis - Detailed Implementation and Migration Plan (including Architecture Implementation Contract, if appropriate)

Steps

Key steps in this phase include:

1. Project recommendation formulation, for each separate implementation project do the following:
    - document scope of individual project in Impact Analysis
    - document strategic requirements (from the architectural perspective) in Impact Analysis
    - document change requests (such as support for a standard interface) in Impact Analysis
    - document rules for conformance in Impact Analysis
    - document time-line requirements from road-map in Impact Analysis
2. Document Architecture Contract
    - obtain signature from all developing organizations and sponsoring organization
3. On-going implementation governance and architecture compliance review.

Outputs

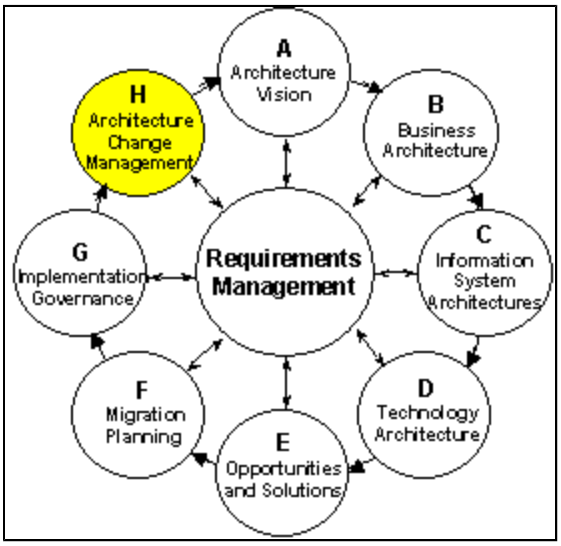
The output of this phase is:

- Impact Analysis - Implementation recommendations
- Architecture Contract, as recommended in Part IV, Architecture Contracts
- The architecture-compliant implemented system
    - Note: the implemented system is actually an output of the development process. However, given the importance of this output, it is stated here as an output of the architecture development method. The direct involvement of architecture staff in implementation will vary according to organizational policy, as described in Part IV, Architecture Governance.

Phase H: Architecture Change Management

Objective Approach Inputs Steps Outputs

Objective



The objective of this phase is to establish an Architecture Change Management process for the new Enterprise Architecture baseline that is achieved with completion of the Implementation Governance phase. This process will typically provide for the continual monitoring of such things as new developments in technology and changes in the business environment, and for determining whether to formally initiate a new architecture evolution cycle.

This phase also provides for changes to the Framework and Principles set up in the Preliminary Phase.

Approach

The goal of an Architecture Change Management process is to ensure that changes to the architecture are managed in a cohesive and architected way, and to establish and support the implemented Enterprise Architecture as a *dynamic* architecture - that is, one having the flexibility to evolve rapidly in reponse to changes in the technology and business environment.

The Change Management process once established will determine:

- The circumstances under which the Enterprise Architecture, or parts of it, will be permitted to change after implementation; and the process by which that will happen
- The circumstances under which the Enterprise Architecture development cycle will be initiated again to develop a new architecture

The Architecture Change Management process is very closely related to the architecture governance processes of the enterprise, and to the management of the Architecture Contract between the Architecture function and the business users of the enterprise.

In this phase it is critical that the governance body establish criteria to judge whether a change request warrants just an architecture update or whether it warrants starting a new cycle of the architecture development method. It is especially important to avoid "creeping elegance", and the governance body must continue to look for changes that relate directly to business value.

Guidelines for establishing these criteria are difficult to prescribe, as many companies accept risk differently, but as the architecture development method is exercised, the maturity level of the governance body will improve, and criteria will become clear for specific needs.

The Drivers for Change

There are many technology related drivers for architecture change requests. For example:

- New technology reports
- Asset Management cost reductions
- Technology withdrawal
- Standards initiatives

This type of change request is normally manageable primarily through an enterprise's change management and architecture governance processes.

In addition there are business drivers for architecture change, including:

- Business-As-Usual developments
- Business Exceptions
- Business Innovations
- Business Technology Innovations
- Strategic Change

This type of change request often results in a complete re-development of the architecture, or at least in an iteration of a part of the architecture development cycle, as explained below.

The Change Management Process

The Change Management process needs to determine how changes are to be managed, what techniques are to be applied, and what methodologies used. The process also needs a filtering function that determines which phases of the architecture development process are impacted by requirements. For example, changes that affect only migration may be of no interest in the architecture development phases.

There are many valid approaches to change management, and various management techniques and methodologies that can be used to manage change: for example, project management methods such as PRINCE 2, service management methods such as ITIL, management consultancy methods such as Catalyst, and many others. An enterprise that already has a change mangement process in place in a field other than Architecture (for example, in Systems Development or Project Management) may well be able to adapt it for use in relation to architecture.

The following describes an approach to change management, aimed particularly at the support of a dynamic enterprise architecture, which may be considered for use if no similar process currently exists.

The approach is based on classifying required architectural changes into one of three categories:

- **Simplification change**: A simplification change can normally be handled via change management techniques.
- **Incremental change**. An incremental change may be capable of being handled via change management techniques, or it may require partial re-architecting, depending on the nature of the change. See below for guidelines.
- **Re-architecting change**. A re-architecting change requires putting the whole architecture through the architecture development cycle again.

Another way of looking at these three choices is to say that a simplification change to an architecture is often driven by a requirement to reduce investment; an incremental change, by a requirement to derive additional value from existing investment; and a re-architecting change, by a requirement to increase investment in order to create new value for exploitation.

To determine whether a change is simplification, incremental, or re-architecting, the following activities are undertaken:

1. Registration of all events that may impact the architecture
2. Resource allocation and management for architecture tasks.
3. The process or role responsible for architecture resources has to make assessment of what should be done
4. Evaluation of impacts

Guidelines for Maintenance versus Architecture Redesign

A good rule-of-thumb is:

- If the change impacts two stakeholders or more, then it is likely to require an architecture re-design and re-entry to the ADM
- If the change impacts only one stakeholder, then it is more likely to ba a candidate for change management
- If the change can be allowed under a dispensation, then it is more likely to ba a candidate for change management

For example:

- If the impact is significant for the business strategy, then there may be a need to redo the whole enterprise architecture - thus a re-architecting approach.
- If a new technology or standards emerge, then there may be a need to refresh the Technology Architecture, but not the whole Enterprise Architecture - thus an incremental change.

- If the change is at an infrastructure level - for example, ten systems reduced or changed to one system - this may not change the architecture above the physical layer, but it will change the baseline description of the Technoloogy Architecture. This would be a simplification change handled via change management techniques.

In particular, a refreshment cycle (partial or complete re-architecting) may be required if:

- the Foundation Architecture needs to re-aligned with the business strategy
- Substantial change is required to components and guidelines for use in deployment of the architecture
- Significant Standards used in the product architecture are changed which have significant end user impact; e.g. regulatory changes

If there is a need for a refreshment cycle, then a New Request for Architecture Work must be issued (to move to another cycle).

Inputs

Inputs to this phase are:

- Request for Architecture Change - technology changes
  - New technology reports
  - Asset Management cost reduction initiatives
  - Technology withdrawal reports
  - Standards initiatives
- Request for Architecture Change - business changes
  - Business developments
  - Business exceptions
  - Business innovations
  - Business technology innovations
  - Strategic change developments

Steps

Key steps in this phase include:

- Ongoing monitoring of technology changes
- Ongoing monitoring of business changes
- Assessment of changes and development of position to act
- Meeting of Architecture Board (or other governing council) to decide on handling changes (technology and business)

Outputs

The outputs of this phase are:

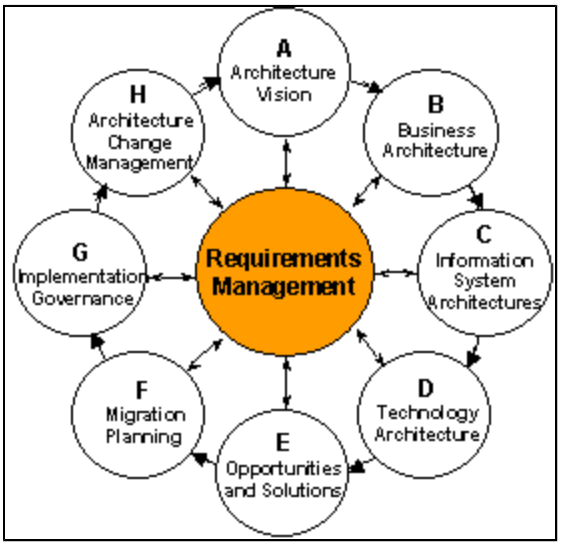
- Architecture updates
- Changes to Architecture Framework and Principles
- New Request for Architecture Work (to move to another cycle)

ADM Architecture Requirements Management

Objectives Approach Inputs Steps Outputs

Objectives

- To define a process whereby requirements for enterprise architecture are identified, stored, and fed into and out of the relevant ADM phases



Approach

General

As indicated by the Requirements Management circle at the centre of the ADM graphic, the ADM is continuously driven by the Requirements Management process.

It is important to note that the Requirements Management circle denotes, not a static set of requirements, but a dynamic process whereby requirements for enterprise architecture *and subsequent changes to those requirements* are identified, stored, and fed into and out of the relevant ADM phases.

The ability to deal with changes in requirements is crucial. Architecture is an activity that by its very nature deals with uncertainty and change - the "grey area" between what stakeholders aspire to and what can be specified and engineered as a solution. Architecture requirements are therefore invariably subject to change in practice. Moreover, Architecture often deals with drivers and constraints, many of which by their very nature are beyond the control of the enterprise (changing market conditions, new legislation, etc.), and which can produce changes in requirements in an unforeseen manner.

Note also that the Requirements Management process itself does not dispose of, address, or prioritize any requirements: this is done within the relevant phase of the ADM. It is merely the process for managing requirements throughout the overall ADM.

Resources

The world of Requirements Engineering is rich with emerging recommendations and processes for Requirements Management. TOGAF does not mandate or recommend any specific process or tool: it simply states what an effective Requirements Management process should achieve (i.e., the "requirements for requirements", if you like).

Business Scenarios

One effective technique that is described in TOGAF itself is Business Scenarios, which are an appropriate and useful technique to discover and document business requirements, and to articulate an architectural vision that responds to those requirements. Business Scenarios are described in detail in Part IV of TOGAF.

Volere Requirements Specification Template

Architecture requirements is very much a niche area within the overall requirements field. One useful resource is the Volere Requirements Specification Template, available from the Volere web site hosted by the Atlantic Systems Guild. While not designed with architecture requirements in mind, this is a very useful requirements template, which is freely available and may be modified or copied (for internal use, provided the copyright is appropriately acknowledged).

One interesting item in this template is the "Waiting Room", which is a hold-all for requirements in waiting. There are often requirements identified which, as a result of the prioritization activity that forms part of the Requirements Management process (see below), are designated as beyond the planned scope, or the time available, for the current iteration of the architecture. The "Waiting Room" is a repository of future requirements. Having the ability to store such requirements helps avoid the perception that they are simply being discarded, while at the same time helping to manage expectations about what

will be delivered.

Requirements Tools

There is a large, and increasing, number of commercial off-the-shelf (COTS) tools available for the support of requirements management, albeit not necessarily designed for architecture requirements. The Volere web site has a very useful list of leading requirements tools.

Inputs

The inputs to the Requirements Management process are the requirements-related outputs from each ADM phase.

The first high-level requirements are articulated as part of the Architecture Vision, generated by means of the Business Scenario or analogous technique.

Each architecture domain then generates detailed design requirements specific to that domain, and potentially to other domains (for example, areas where already designed architecture domains may need to change to cater for changes in this architecture domain; constraints on other architecture domains still to be designed.)

Deliverables in later ADM phases also contain mappings to the design requirements, and may also generate new types of requirements (for example, conformance requirements; time windows for implementation).

Steps

Key steps in the Requirements Management process include:

	Requirements Management Steps	ADM Phase steps
1		Identify / document requirements - use Business Scenarios, or an analogous technique.
2	Baseline requirements: a) Determine priorities arising from current phase of ADM. b) Confirm stakeholder buy-in to resultant priorities. c) Record requirements priorities and place in Requirements Repository.	
3	Monitor baseline requirements	
4		Identify changed requirement: a) Remove or re-assess priorities. b) Add requirements and re-assess priorities. c) Modify existing requirements.
5	Identify changed requirement and record priorities: a) Identify changed requirements and ensure the requirements are prioritized by the architect(s) responsible for the current phase, and by the relevant stakeholders. b) Record new priorities. c) Ensure that any conflicts are identified and managed through the phases to a successful conclusion and prioritization. d) Generate Requirements Impact Statement for steering the architecture team. Notes: • Changed requirements can come in through any route. To ensure that the requirements are properly assessed and prioritized, this process needs to direct the	

	ADM phases and record the decisions related to the requirements. • The Requirements Management phase needs to determine stakeholder satisfaction with the decisions. Where there is dissatisfaction, the phase remains accountable to ensure the resolution of the issues and determine next steps.	
6		a) Assess impact of changed requirements on current (active) phase. b) Assess impact of changed requirements on previous phases. c) Determine whether to implement change, or defer to later ADM cycle. If decision is to implement, assess timescale for change management implementation. d) Issue Requirements Impact Statement version n+1.
7		Implement requirements arising from Architecture Change Management phase. • The architecture can be changed through its lifecycle by the Architecture Change Management phase. The Requirements Management process ensures that new or changing requirements that are derived from the Architecture Change Management phase are managed accordingly.
8	Update the Requirements Repository with information relating to the changes requested, including stakeholder views affected.	
9		Implement change in the current phase.
10		Assess and revise Gap Analysis for past phases. The Gap Analysis in the ADM phases B through D identifies the gaps between baseline and target architectures. Certain types of gap can give rise to gap requirements. The ADM describes two kinds of gap: 1) something that is present in the baseline, but not in the target (i.e., eliminated - by accident or design); 2) something not in the baseline, but present in the target (i.e., new). A "gap requirement" is anything that has been eliminated by accident, and therefore requires a change to the target architecture. If the gap analysis generates gap requirements, then this step will ensure that they are addressed, documented, and recorded in the requirements repository, and that the target architecture is revised accordingly.

Outputs

The output of the Requirements Management process itself is:

- A **Structured Requirements Statement**, including:
  - Changed Requirements
  - Requirements Impact Statement
    - The Requirements Repository contains the current requirements for the target architecture. When new requirements arise, or existing ones are changed, a Requirements Impact Statement is generated, which identifies the phases of the ADM that need to be revisited to address the changes. The Statement goes through various iterations until the final version, which includes the full implications of the requirements (e.g., costs, timescales, business metrics) on the architecture development.

ADM Input and Output Descriptions

Introduction Input Descriptions Output Descriptions

Introduction

This section provides example descriptions of input and output items referenced in the Architecture Development Method.

Note that not all the content described here need be contained in a particular input or output. Rather, it is recommended that external references be used where possible: for example, the strategic plans should not be copied into the Request for Architecture Work, but rather the title of the strategic plans should be referenced.

Also, it is not suggested that these descriptions should be followed to the letter. However, each element should be considered carefully: ignoring any input or output item may cause problems downstream.

Finally, note that versioning is used to indicate that input or output items may undergo change as the ADM is executed. As an input or output item is updated, a new version may be produced. The initial version is named Version 1, and subsequent versions are named Version 2, Version 3, etc.

Major Input Descriptions

Request for Architecture Work

Organization sponsors
Organization's mission statement
Business goals (and changes)
Strategic plans of the business
Time limits
Changes in the business environment
Organizational constraints
Budget information, financial constraints
External constraints, business constraints
Current business system description
Current architecture/IT system description
Description of developing organization
Description of resources developing organization has available

Architecture Principles

See Part IV, Architecture Principles, for guidelines and a detailed set of generic Architecture Principles, including:

- Business Principles
- Data Principles
- Application Principles
- Technical Principles

Re-usable Architecture Building Blocks

Architecture documentation and models from the enterprise's Architecture Continuum.

Product Information

Functional descriptions of products that are candidates for the implementation

Architectural descriptions of elements that are candidates for the implementation

New Technology Reports

New developments in potentially relevant technology

Major Output Descriptions

Statement of Architecture Work

Statement of work title
Project request and background
Project description and scope
Architecture vision
Managerial approach
Change of scope procedures
Responsibilities and deliverables
Acceptance criteria and procedures
Project plan and schedule
Support of the enterprise continuum
Signature approvals

Business Scenario / Architecture Vision

Problem Description
- Purpose of Scenario

Detailed Objectives

Environment and Process Models
- Process Description
- Process Steps Mapped to Environment
- Process Steps Mapped to People
- Information Flow

Actors and Their Roles and Responsibilities
- Human Actors and Roles
- Computer Actors and Roles
- Requirements

Resulting Architecture Model
- Constraints
- IT Principles
- Architecture Supporting the Process
- Requirements Mapped to Architecture

Business Architecture

Baseline Business Architecture

Business Goals, Objectives and Constraints
- Business requirements and key system and architecture drivers
- Business return given required changes
- Assumptions (e.g. business, financial, organizational, or required technical functionality)
- Business Architecture principles

Business Architecture Models
- Organization structure
- Business functions
- Business roles
- Correlation of organization and functions
- Business Architecture building blocks list (e.g., business services)
- Business Architecture building blocks models

Candidate solution building blocks list
Candidate solution building blocks models
Relevant business process descriptions, including measures and deliverables

Technical requirements (drivers for other Architecture work)

Technology Architecture

Baseline Technology Architecture

Objectives and Constraints

Technology requirements and key system and architecture drivers
Assumptions (e.g. business, financial, organizational, or required technical functionality)

Technical architecture model(s)

Architecture building block models of views (minimally model of functions and a model of services)
Architecture building block models of service portfolios (enterprise specific framework)

Technical architecture specification

Per architecture building block:

Details of the technical functionality
Fully defined list of all the standards
Description of building block at the levels necessary to support implementation, enterprise wide strategic decision making, and further iterations of the architectural definition process
Rationale for decision taken that relate to the building block, including rationales for decisions not to do something
Specification identifying the inter-working with other building blocks and how they do so
Guidelines for procuring

Standards summary list

Requirements traceability

Acceptance criteria
Criteria for choosing specifications
Criteria for selection of portfolios of specifications
Criteria to test merits of architecture (key question list)
Report on cost/benefit analyses
Report on how the proposed architecture meets the business goals and objectives
Criteria response answers to key question list to test merits of architecture

Gap report

Report on gap analysis
Report of gap analysis matrix

Mapping of the architectures in the Enterprise Continuum

Change requests for extensions or amendments to related architectures

Impact Analysis

Project list

Name, description and objectives of each impacted project
Prioritized list of impacted projects to implement the proposed architecture

Time oriented migration plan

Benefits of migration, determined [including mapping to business requirements]
Estimated costs of migration options

Implementation recommendations

Criteria measures of effectiveness of projects
Risks and issues
Solutions building blocks - description and model

Architecture Contract

Typical contents of an Architecture Design and Development Contract:

Introduction & Background
The nature of the agreement
Scope of the Architecture
Architecture and Strategic principles and requirements
Conformance requirements
Architecture development and management process and roles
Target architecture measures
Defined Phases of deliverables

Prioritised Joint Workplan
Time window(s)
Architecture delivery and business metrics

Typical contents of a Business Users' Architecture Contract:

Introduction & Background
The nature of the agreement
Scope
Strategic requirements
Conformance requirements
Architecture adopters
Time window
Architecture business metrics
Service Architecture (includes Service Level Agreement)

This contract is also used to manage changes to the Enterprise Architecture in Phase H, Architecture Change Management.

Requirements Impact Statement

Reference to specific Requirements
Stakeholder Priority of the Requirements to date
Phases to be revisited
Phase to lead on requirements prioritisation
Results of Phase investigations and revised priorities
Recommendations on Management of Requirements
Repository reference number

Building Blocks and the Architecture Development Method

Principles Specification Process Levels of Modelling

Basic Principles

This subsection focuses on the use of building blocks in the ADM. General considerations and characteristics of Building Blocks are described under Introduction to Building Blocks.

Building Blocks in Architecture Design

An architecture is a set of building blocks depicted in an architectural model, and a specification of how those building blocks are connected to meet the overall requirements of an information system.

The various building blocks in an architecture specify the services required in an enterprise-specific system.

There are some general principles underlying the use of building blocks in the design of specific architectures:

- An architecture need only contain building blocks to implement those services that it requires.
- Building blocks may implement one, more than one, or only part of a service identified in the architecture framework.
- Building blocks should conform to standards relevant to the services they implement.

Building Block Design

The process of identifying building blocks includes looking for collections of functions which require integration to draw them together or make them different:

- Consider three classes of building blocks
  - re-usable building blocks such as legacy items
  - building blocks to be the subject of development such as new applications
  - building blocks to be the subject of purchase, i.e. commercial off-the-shelf applications
- Use the desired level of integration to bind or combine functions into building blocks. For instance legacy elements could be treated as large building blocks to avoid breaking them apart.

In the early stages and during views of the highest level enterprise, the Building Blocks are often kept at a broad integration definition. It is during these exercises that the services definitions can often be best viewed. As implementation considerations are addressed, more detailed views of Building Blocks can often be used to address implementation decisions, focus on the critical strategic decisions or aid in assessing the value and future impact of commonality and reusability.

Building Block Specification Process in the ADM

The process of building block definition takes place gradually as the Architecture Development Method is followed, mainly in Phases A, B, C, and D. It is an iterative process because as definition proceeds, detailed information about the functionality required, the constraints imposed on the architecture and the availability of products may affect the choice and the content of building blocks.

The key parts of the ADM at which building blocks are designed and specified are summarized below.

The major work in these steps consists of identifying the architectural building blocks (ABBs) required to meet the business goals and objectives. The selected set of ABBs is then refined in an iterative process to arrive at a set of Solution Building Blocks (SBBs) which can either be bought off the shelf or custom developed.

The specification of building blocks using the ADM is an evolutionary and iterative process. The key Phases and Steps of the

ADM at which building blocks are evolved and specified are summarized below, and illustrated in Figure 1.

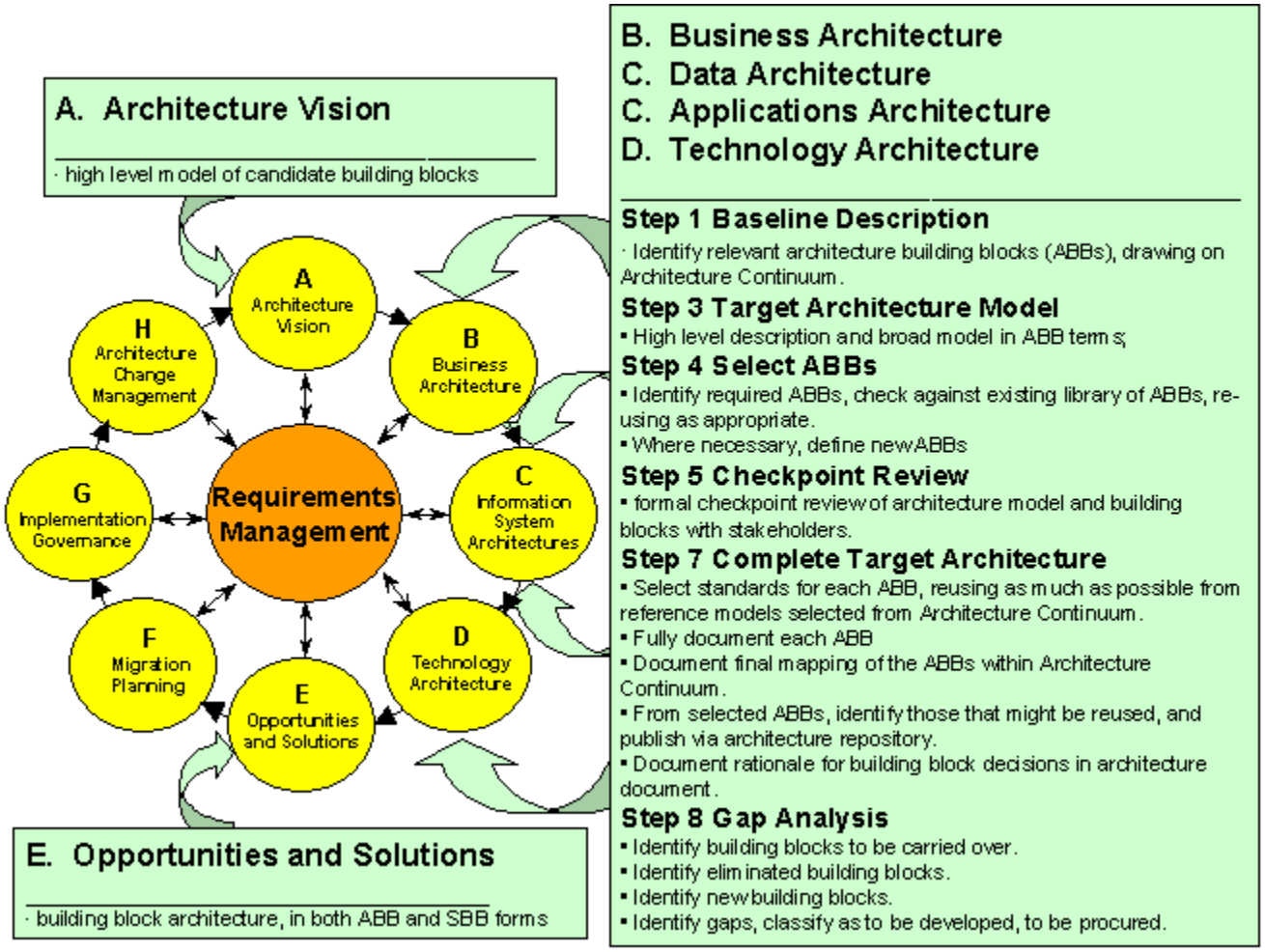


Figure 1: Key Phases / Steps of ADM at which Building Blocks are Evolved / Specified

In Phase A the earliest building block definitions start as relatively abstract entities within the Architecture Vision.

In Phases B, C, and D, building blocks within the Business, Data, Applications, and Technology Architectures are evolved to a common pattern of steps:

- Step 1, Baseline Description produces:
  - A list of candidate building blocks, from the analysis of the baseline.
- Step 3, Target Architecture Model, takes this list and high level model as inputs, and evolves them iteratively into a definition of the target architecture, specified in terms of architectural building blocks. Specifically, Step 3 produces:
  - a high level description and broad model of the target system in terms of Architectural Building Blocks.
  - a rationale for each building block decision
- Step 4 produces:
  - for each Architectural Building Block, a service description portfolio, built up as a set of non-conflicting services.
- Step 5 produces:
  - a confirmation of the merit and completeness of the model and service description portfolio, and a description of how the emerging target architecture meets the objectives of the architecture development
- Step 7 produces:
  - A target architecture, fully specified in terms of Architectural Building Blocks
  - a fully defined (by service) list of all the standards that make up the target architecture, and all the Architectural building blocks that will be used to implement it.
  - a diagrammatic depiction of the building blocks at the levels needed to describe the strategic and implementation aspects of the architecture.
- Step 8 produces:
  - a gap analysis of eliminated building blocks, carried over building blocks, and new building blocks

Finally in Phase E, Opportunities and Solutions, the building blocks become more implementation specific as Solution Building

Blocks, and their interfaces become the detailed architecture specification. The output of Phase E is the building block architecture, both in Architectural (i.e. functionally defined) and Solution (i.e. product-specific) Building Block forms.

The minimum contents of an Architectural Building Block specification and a Solution Building Block specification are described under Introduction to Building Blocks.

Levels of Modelling

Defining and developing the *context* for a set of building blocks takes place at two levels:

- The **business process** level (colored green in the diagrams in this section). This deals only at the highest level with what has to happen for a business process to be carried out.
- The **technical functionality and constraints** level (colored blue in the diagrams). This level deals with the component activities that form part of the business process and whether they can be supported or not.

Defining and developing an actual set of building blocks also takes place at two levels:

- The **architectural model** level (colored red in the diagrams). This level identifies the systems and components that will implement the technical functionality and expresses the relationships between them. This level introduces the idea of a notation to describe the architectural elements and relationships.
- The **solution model** level (colored black in the diagrams). This is the level where the individual products and/or product components that will implement the architecture are identified.

Working through the four levels is an iterative process. Figure 2 shows how considerations at any level can result in change at any or all of the other levels.

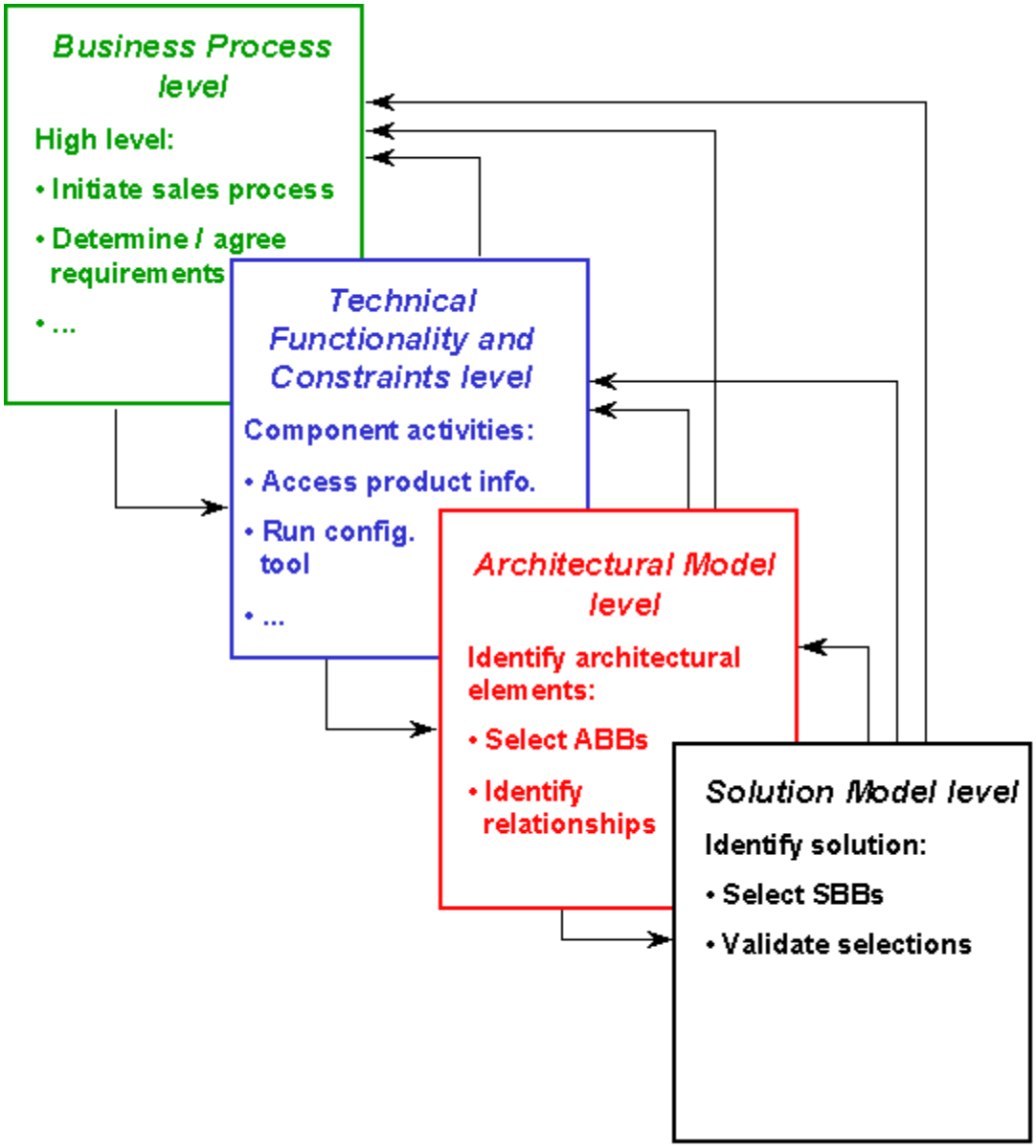


Figure 2: Iteration between the Four Levels of Modeling

Mapping the Modelling Levels to the ADM

The **business process** level of definition takes place in Phases A and B of the ADM.

The **technical functionality and constraints** level work happens early in Phases B, C, and D, once the characteristics of the current system have been established. At that stage it is possible to identify the constraints imposed on new architecture work by the legacy of the old system (the baseline).

The **architectural model** and **solution model** levels consist of work done later in Phases B, C, and D, the Target Architecture steps of each phase, and Phase E, Opportunity Identification. The architectural model work is mostly done when taking different views of the architecture, and the solution model work in Phase E, where selection of products and projects takes place.

To show how building block definition happens in practice, the remainder of this appendix consists of a fictional worked example. In this fictional example much detail has been left out in order to emphasise the process.

PART III: Enterprise Continuum

The Enterprise Continuum: Introduction

Overview Architecture Re-use ADM Constituents Structure of Part III

Overview

This section introduces the concept of the *Enterprise Continuum*, which sets the broader context for TOGAF, by explaining the different types and scopes of the architecture artifacts and assets that can be derived from it, and leveraged during its use.

The Enterprise Continuum is an important aid to communication and understanding, both within individual enterprises, and between customer enterprises and vendor organizations. Without an understanding of "where in the continuum you are", people discussing architecture can often talk at cross purposes because they are referencing different points in the continuum at the same time, without realizing it.

Any architecture is context specific; for example, there are architectures that are specific to individual customers, industries, subsystems, products, and services. Architects, on both the buy side and supply side, must have at their disposal a consistent language to effectively communicate the differences between architectures. Such a language will enable engineering efficiency and the effective leveraging of Commercial Off-The-Shelf (COTS) product functionality. The Enterprise Continuum provides that consistent language.

Not only does the Enterprise Continuum represent an aid to communication, it represents an aid to organizing re-usable architecture and solution assets. This is explained further below.

The Enterprise Continuum and Architecture Re-use

The simplest way of thinking of the Enterprise Continuum is as a "virtual repository" of all the architecture assets - models, patterns, architecture descriptions, and other artifacts - that exist both within the enterprise and in the IT industry at large, which the enterprise considers itself to have available for the development of architectures for the enterprise.

Examples of "assets within the enterprise" are the deliverables of previous architecture work, which are available for reuse. Examples of "assets in the IT industry at large" are the wide variety of industry reference models and architecture patterns that exist, are continually emerging, including those that are highly generic (such as TOGAF's own Technical Reference Model); those specific to certain aspects of information technology (such as a web services architecture, or a generic manageability architecture); those specific to certain types of information processing, such as e-Commerce, supply chain management, etc.; and those specific to certain vertical industries, such as the models generated by vertical consortia such as TMF (in the Telecommunicaitons sector), ARTS (retial), POSC (petrotechnical), etc.

The decision as to which architecture assets a specific enterprise considers part of its own Enterprise Continuum will normally form part of the overall Architecture Governance function within the enterprise concerned.

The Enterprise Continuum and the TOGAF Architecture Development Method

The TOGAF Architecture Development Method describes the process of moving from the TOGAF Foundation Architecture to an enterprise-specific architecture (or set of architectures). This process leverages the elements of the TOGAF Foundation Architecture and other relevant architectural assets, components, and building blocks along the way.

At relevant places throughout the TOGAF ADM, there are reminders to consider which architecture assets from the Enterprise Continuum the architect should use, if any. In some cases, from example in the development of a Technology Architecture, this may be TOGAF's own Foundation Architecture (see below). In other cases, for example in the development of a business architecture, it may be a reference model for e-Commerce taken from the industry at large.

TOGAF itself provides two reference models for consideration for inclusion in an organization's Enterprise Continuum:

- The **TOGAF Foundation Architecture**. This foundation architecture comprises a Technical Reference Model of generic services and functions that provides a firm foundation on which more specific architectures and architectural

components can be built; and a Standards Information Base - an information base of relevant specifications and standards.
- The **Integrated Information Infrastructure Reference Model**, which is based on the TOGAF Foundation Architecture, and is specifically designed to help the realization of architectures that enable and support the "Boundaryless Information Flow" vision.

However, in developing architectures in the various domains within an overall enterprise architecture, the architect will need to consider the use and re-use of a wide variety of different architecture assets, and the Enterprise Continuum provides a framework (a "framework-within-a framework", if you like) for categorizing and communicating these different assets.

Constituents of the Enterprise Continuum

The "Enterprise Continuum" is a phrase that denotes the combination of two complementary concepts: the "Architecture Continuum" and the "Solutions Continuum".

- The Architecture Continuum offers a consistent way to define and understand the generic rules, representations and relationships in an information system. The Architecture Continuum represents a structuring of re-usable architecture assets. The Architecture Continuum is directly supported by the Solutions Continuum (see below). The Architecture Continuum shows the relationships among foundational frameworks such as TOGAF, common systems architectures such as the Integrated Information Infrastructure Reference Model, industry architectures, and enterprise architectures. The Architecture Continuum is a useful tool to discover commonality and eliminate unnecessary redundancy.

- The Solutions Continuum provides a consistent way to describe and understand the implementation of the Architecture Continuum. The Solutions Continuum defines what is available in the organizational environment as re-usable solutions building blocks. The solutions are the results of agreements, between customers and business partners, that implement the rules and relationships defined in the architecture space. The Solutions Continuum addresses the commonalties and differences among the products, systems and services of implemented systems.

The immediately following section provides greater detail about the Enterprise Continuum. It first outlines the Architecture Continuum and Solutions Continuum separately, and then explains how they come together to form the Enterprise Continuum.

Structure of Part III

The remainder of Part III is structured as follows:

- Introduction (the present section)
- The Enterprise Continuum in Detail
  - Architecture Continuum
  - Solutions Continuum
  - Entrprise Continuum and Your Organization
- TOGAF Foundation Architecture: Technical Reference Model
  - TRM Concepts
  - High-level Breakdown
  - TRM in Detail
  - Detailed Platform Taxonomy
- TOGAF Foundation Architecture: Standards Information Base
  - Introduction
  - Open Group Standards
  - Using the SIB and Linked Resources
- TOGAF Integrated Information Infrastructure Reference Model
  - Basic Concepts
  - High-Level View
  - Detailed Taxonomy

The Architecture Continuum

Introduction Foundation Architecture Common System Architectures Industry Architectures Enterprise Architectures

Introduction

There is a continuum of architectures, architectural building blocks, and architectural models, that are relevant to the task of constructing an enterprise-specific architecture.

The Architecture Continuum, and the relative positioning of different types of architectures within it, is illustrated in Figure 1.

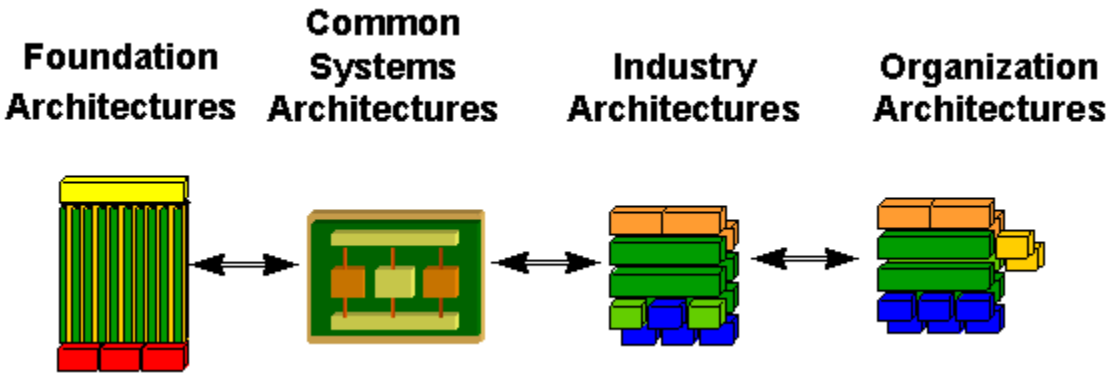


Figure 1: The Architecture Continuum

Figure 1 illustrates how architectures are developed across a continuum ranging from foundational architectures such as TOGAF's, through common systems architectures, and industry-specific architectures, to an enterprise's own individual architectures.

The arrows in Figure 1 represents the bi-directional relationship that exists between the different architectures in the Architecture Continuum. The leftwards direction focuses on meeting enterprise needs and business requirements, while the rightwards direction focuses on leveraging architectural components and building blocks.

The enterprise needs and business requirements are addressed in increasing detail from left to right. The architect will typically look to find re-usable architectural elements toward the left of the continuum. When elements are not found, the requirements for the missing elements are passed to the left of the continuum for incorporation. Those implementing architectures within their own organizations can use the same continuum models specialized for their business.

The four particular architectures illustrated in Figure 1 are intended to indicate the range of different types of architecture that may be developed at different points in the continuum: they are not fixed stages in a process. Many different types of architecture may occur at points in-between those illustrated in Figure 1.

Although the continuum illustrated in Figure 1 does not represent a formal process, it does represent a progression, which occurs at several levels:

- Logical to physical
- Horizontal (IT technology focused) to vertical (business focused)
- Generalization to specialization
- Taxonomy to complete and specific architecture specification

At each point in the continuum, an architecture is designed in terms of the design concepts and building blocks available and

relevant to that point.

The four architectures illustrated in Figure 1 represent main classifications of potential architectures, and will be relevant and familiar to many architects. They are analyzed in detail in the following subsections.

Foundation Architecture

A Foundation Architecture is an architecture of building blocks and corresponding standards that supports all the common systems architectures, and, therefore, the complete computing environment.

For The Open Group, this Foundation Architecture is the TOGAF Technical Reference Model and Standards Information Base. The TOGAF ADM explains how to get from that foundation architecture to an enterprise specific one.

The TOGAF TRM and SIB describe a fundamental architecture upon which other, more specific, architectures can be based. The As a TOGAF Foundation Architecture, it contains many alternatives in each of the architectural building blocks. Other characteristics of the TOGAF foundation architecture include the following:

- reflects general computing requirements
- reflects general building blocks
- defines technology standards for implementing these building blocks
- provides direction for products and services
- reflects the function of a complete, robust computing environment that can be used as a foundation
- provides open system standards, directions, and recommendations
- reflects directions and strategies

Common Systems Architectures

Common Systems Architectures guide the selection and integration of specific services from the Foundation Architecture to create an architecture useful for building common (i.e., reusable) solutions across a wide number of relevant domains.

Examples of Common Systems Architectures include: a Security Architecture, a Management Architecture, a Network Architecture, etc. Each is incomplete in terms of overall information system functionality, but is complete in terms of a particular problem domain (security, manageability, networking, etc.), so that solutions implementing the architecture constitute reusable building blocks for the creation of functionally complete information systems.

Other characteristics of Common Systems Architectures include:

- reflects requirements specific to a generic problem domain
- defines building blocks specific to a generic problem domain
- defines technology standards for implementing these building blocks
- provides building blocks for easy reuse and lower costs

The TOGAF Integrated Information Infrastructure Reference Model is a Common Systems Architecture that focuses on the requirements, building blocks, and standards relating to the vision of Boundaryless Information Flow.

Industry Architectures

Industry Architectures guide the integration of common systems components with industry-specific components, and guide the creation of industry solutions for targeted customer problems within a particular industry.

A typical example of an industry-specific component is a Data Model representing the business functions and processes specific to a particular vertical industry, such as the Retail industry's "Active Store" architecture, or an industry architecture that incorporates the Petrotechnical Open Software Corporation (POSC) Data Model.

Other characteristics of Industry Architectures include the following:

- reflects requirements and standards specific to a vertical industry
- defines building blocks specific to a generic problem domain
- contains industry-specific logical data and process models
- contains industry-specific applications and process models, as well as industry-specific business rules

- provides guidelines for testing collections of systems
- encourages levels of interoperability throughout the industry

Enterprise Architectures

Enterprise Architectures are the most relevant to the IT customer community, since they describe and guide the final deployment of user-written or third-party components that constitute effective solutions for a particular enterprise or enterprises that have a need to share information.

IEEE Std 1003.23, *Guide for Developing User Organization Open System Environment (OSE) Profiles*, provides a method for identifying and documenting an organization's operational requirements, the Information Systems services and Information Technology services needed to support those requirements, and the standards, standards options, interim solutions and products that will provide the needed services.

There may be a continuum of Enterprise Architectures that are needed to effectively cover the organization's requirements by defining the Enterprise Architecture in increasing levels of detail. Alternatively, this might result in several more detailed Enterprise Architectures for specific entities within the global enterprise.

The Enterprise Architecture guides the final customization of the solution, and has the following characteristics:

- provides a means to communicate and manage the information technology environment
- reflects requirements specific to a particular enterprise
- defines building blocks specific to a particularenterprise
- contains organization-specific physical data, applications, and process models, as well as business rules
- provides a means to encourage implementation of appropriate information technology to meet business needs
- provides the criteria to measure and select appropriate products, solutions, and services
- provides an evolutionary path to support growth and new business needs

The Solutions Continuum

Introduction Products and Services System Solutions Industry Solutions Enterprise Solutions

Introduction

The Solutions Continuum represents the implementations of the architectures at the corresponding levels of the Architecture Continuum. At each level, the Solutions Continuum is a population of the architecture with reference building blocks, either purchased products or built components, that represent a solution to the enterprise's business need expressed at that level. A populated Solutions Continuum can be regarded as a solutions inventory or re-use library, which can add significant value to the task of managing and implementing improvements to the IT environment.

The Solutions Continuum is illustrated in Figure 3.

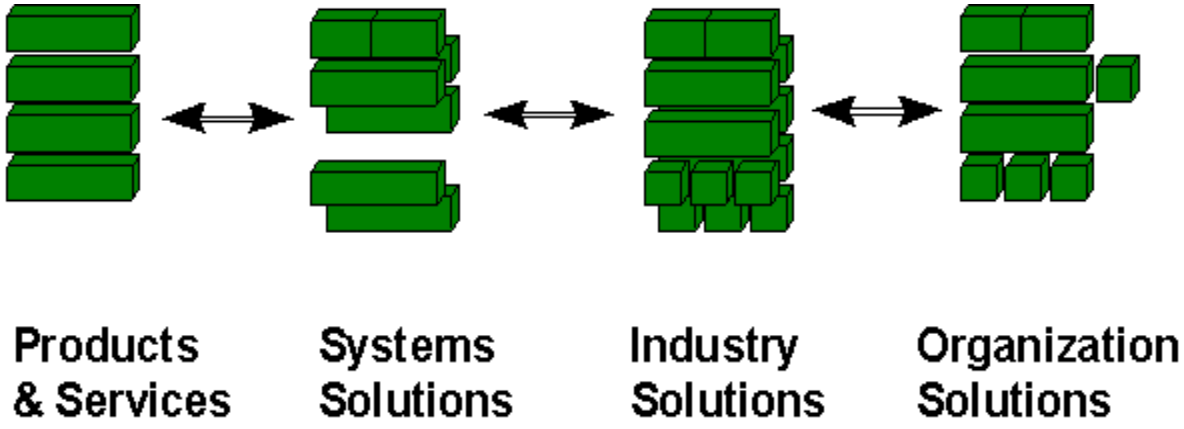


Figure 3: The Solutions Continuum

In the Architecture Continuum in Figure 1, the arrows represent bi-directional relationships that exist between the different architectures in the Architecture Continuum (the leftwards direction focused on meeting customer needs and business requirements, the rightwards direction on leveraging architectural components and building blocks).

A similar concept applies to the bi-directional arrows underlying the Solutions Continuum in Figure 3. The rightwards direction of the arrows is focused on providing solutions value (i.e., Products and Services provides value in creating Systems Solutions; Systems Solutions value is used to create Industry Solutions; and Industry Solutions are used to create Customer Solutions). The leftwards direction is focused on addressing enterprise needs.

These two viewpoints are significant for a company attempting to focus on its needs while maximizing the use of its internal resources through leverage.

The following subsections describe each of the solution types within the Solutions Continuum.

Products and Services

Products are separately procurable hardware, software, or service entities. Products are the fundamental providers of capabilities.

Services include professional services, such as training and consulting services, that ensure the maximum investment value from solutions in the shortest possible time; and support services, such as Help Desk, that ensure the maximum possible value from solutions (services that ensure timely updates and upgrades to the products and systems).

Systems Solutions

A System Solution is an implementation of a Common System Architecture comprised of a set of products and services, which may be certified or branded. It represents the highest common denominator for one or more solutions in the industry segments that the System Solution supports.

System Solutions represent collections of common requirements and capabilities, rather than those specific to a particular customer or industry. System Solutions provide organizations with operating environments specific to operational and informational needs, such as High Availability Transaction Processing and Scaleable Data Warehousing systems. Examples of Systems Solutions include: an Enterprise Management System product, or a Security System product.

Computer systems vendors are the primary provider of Systems.

Industry Solutions

An Industry Solution is an implementation of an Industry Architecture, which provides reusable packages of common components and services specific to an industry.

Fundamental components are provided by System Solutions and/or Products and Services, and are augmented with industry specific components. Examples include: a physical database schema or an industry specific point-of-service device.

Industry Solutions are industry-specific, aggregate procurements that are ready to be tailored to an individual organization's requirements.

In some cases an Industry Solution may include not only an implementation of the Industry Architecture, but also other solution elements, such as specific products, services and systems solutions that are appropriate to that industry.

Enterprise Solutions

An Enterprise Solution is an implementation of the Enterprise Architecture that provides the required business functions. Because solutions are designed for specific business operations, they contain the highest amount of unique content in order to accommodate the varying people and processes of specific organizations.

Building Enterprise Solutions on Industry Solutions, Systems Solutions, and Products and Services, is the primary purpose of connecting the Architecture Continuum to the Solutions Continuum, as guided by the Architects within an enterprise.

The Enterprise Continuum and Your Enterprise

Introduction Relationships Your Enterprise

Introduction

The preceding sections have described both the Architecture Continuum and the Solutions Continuum. As explained in the Introduction, the combination of these two complementary concepts constitutes the "Enterprise Continuum", illustrated in Figure 4.

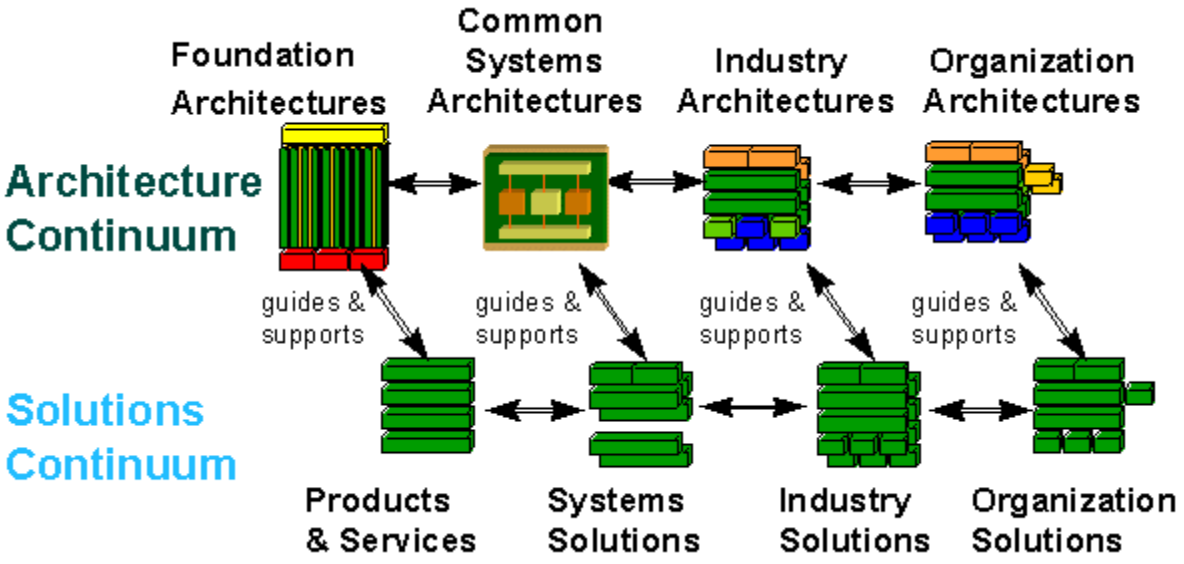


Figure 4: The Enterprise Continuum

Relationships

The relationship between the Architecture Continuum and the Solutions Continuum is one of guidance, direction, and support.

For example, the Foundation Architecture guides the creation or selection of products and services. The Products and Services support the Foundation Architecture by helping to realize the architecture defined in the Architecture Continuum. The Foundation Architecture also guides development of Products and Services, by providing descriptions of required functions to build open computing systems, and the rationale for those functions. A similar relationship exists between the other elements of the Enterprise Continuum.

The Enterprise Continuum presents mechanisms to help improve productivity through leverage. The Solutions Continuum offers a consistent way to understand the different products, systems, services, and solutions required. The Architecture Continuum offers a consistent way to understand the different architectures and their components.

The Enterprise Continuum should not be interpreted as representing strictly chained relationships. Enterprise Architectures could have components from a Common Systems Architecture, and Enterprise Solutions could contain a product or service. The relationships depicted in Figure 4 are a best case for the ideal leveraging of architecture and solution components.

Finally, it is worth emphasizing that the beginning and the end of the Enterprise Continuum lies in a Foundation Architecture, which serves as a tool chest or repository of reusable guidelines and standards. For The Open Group, this Foundation Architecture is The Open Group's own TOGAF Technical Reference Model and Standards Information Base.

Your Enterprise

TOGAF provides a method for you to "architect" the IT systems in your enterprise. Your architecture organization will have to deal with each type of architecture described above. For example it is recommended that you have you own foundational architecture that governs all your IT systems. You should also have your own common systems architectures that govern major shared infrastructure systems - such as the networking system or management system. You may have your own industry-specific architectures that govern the way your IT systems must behave within your industry. Finally any given department or organization within your business may need its own individual enterprise architecture to govern the IT Systems within that department. All in all this implies that your architecture organization is responsible for maintaining your business' Architecture Continuum.

Your architecture organization will either adopt or adapt existing architectures, or will develop your own architectures from ground up. In either case TOGAF represents a tool to help. It provides a method to assist you in generating/maintaining any type of architecture within the architecture continuum while leveraging architecture assets already defined, internal or external to your organization. The TOGAF architecture development method helps you re-use architecture assets, making your architecture organization more efficient and effective.

Technical Reference Model Concepts

Role Components Other TRMs

Role of the TRM in the Foundation Architecture

The *TOGAF Foundation Architecture* is an architecture of generic services and functions that provides a foundation on which more specific architectures and architectural components can be built. This Foundation Architecture has two main elements:

- the TOGAF **Technical Reference Model (TRM)**, which provides a model and taxonomy of generic platform services; and
- the TOGAF **Standards Information Base (SIB)**, which provides a database of standards that can be used to define the particular services and other components of an organization specific architecture that is derived from the TOGAF Foundation Architecture.

The Technical Reference Model is universally applicable, and therefore can be used to build any system architecture. The list of standards and specifications in the Standards Information Base is designed to facilitate choice by concentrating on open standards, so that architectures derived from the framework will have good characteristics of interoperability and software portability, but the TOGAF Architecture Development Method specifically includes extending the list of specifications to allow for coexistence with or migration from other architectures.

TRM Components

Any technical reference model has two main components:

- a *Taxonomy*, which defines terminology, and provides a coherent description of the components and conceptual structure of an information system
- an associated *Technical Reference Model graphic*, which provides a visual representation of the taxonomy, as an aid to understanding.

The objective of the TOGAF Technical Reference Model is to provide a widely accepted core taxonomy, and an appropriate visual representation of that taxonomy. The Technical Reference Model graphic is illustrated under The Technical Reference Model in Detail, and the taxonomy is explained under Application Platform - Taxonomy.

Other TRMs

One of the great difficulties in developing an architectural framework is in choosing a technical reference model that works for everyone.

The TOGAF TRM was originally derived from the TAFIM TRM (which in turn was derived from the IEEE POSIX 1003.0 model - see Part IV, ISO/IEC 14252 (IEEE Std 1003.0), for details). This TRM is 'platform-centric': it focuses on the services and structure of the underlying platform necessary to support the use and reuse of applications (i.e., on application portability). In particular, it centers on the interfaces between that platform and the supported applications, and between the platform and the external environment.

The current TOGAF TRM is an amended version of the TAFIM TRM, which aims to emphasise the aspect of interoperability as well as that of portability.

The objective of the TRM is to enable structured definition of the standardized application platform and its associated interfaces. The other entities, which are needed in any specific architecture, are only addressed in the Technical Reference Model insofar as they influence the application platform. The underlying aim in this approach is to ensure that the higher level building blocks which make up business solutions have a complete, robust platform on which to run.

Other architectural models - taxonomies and/or graphics - not only are possible, but may be preferable for some enterprises. For example, such an enterprise-specific model could be derived by extension or adaptation of the TOGAF TRM. Alternatively,

a different taxonomy may be embodied in the legacy of previous architectural work by an enterprise, and the enterprise may prefer to perpetuate use of that taxonomy. Similarly, an enterprise may prefer to represent the TOGAF taxonomy (or its own taxonomy) using a different form of graphic, which better captures legacy concepts and proves easier for internal communication purposes.

Apart from the need to recognize that the structure embodied in the taxonomy is reflected in the structure of the Standards Information Base (so any enterprise adopting a different taxonomy will need to reflect its taxonomy in its own SIB), there is no problem with using other architectural taxonomies and/or graphics with TOGAF. The core of TOGAF is its Architecture Development Method: the TRM and the SIB are tools used in applying the ADM in the development of specific architectures. Provided consistency between TRM and SIB are maintained, the TOGAF ADM is valid whatever the choice of specific taxonomy, TRM graphic, or SIB toolset.

Technical Reference Model - High Level Breakdown

Overview Portability and Interoperability

Overview

The coarsest breakdown of the Technical Reference Model is shown in Figure 1 below, which shows three major entities (Applications, Application Platform, and Communications Infrastructure) connected by two interfaces (Application Platform Interface and Communications Infrastructure Interface).

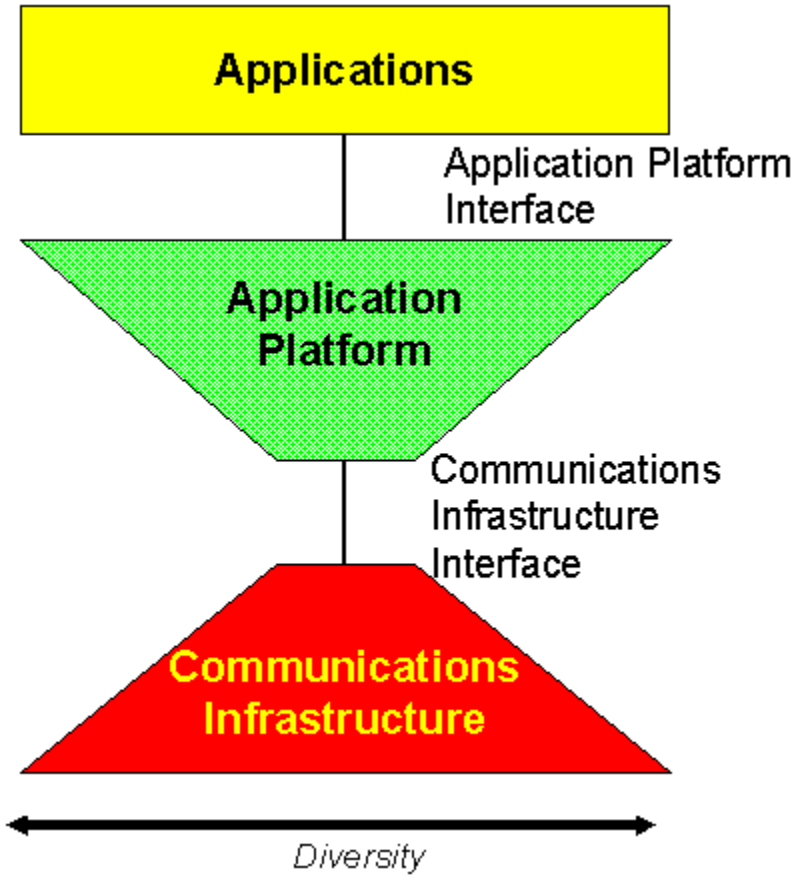


Figure 1: Technical Reference Model - High-Level View

The diagram says nothing about the detailed relationships between the entities, only that they exist.

Each of the elements in this diagram is discussed in detail under The Technical Reference Model in Detail.

Portability and Interoperability

The high-level Technical Reference Model seeks to emphasise two major common architectural objectives:

- *application portability*, via the application platform interface - identifying the set of services that are to be made available in a standard way to applications via the platform; and
- *interoperability*, via the communications infrastructure interface - identifying the set of communications infrastructure services that are to be leveraged in a standard way by the platform.

Both of these goals are essential to enable integration within the enterprise and trusted interoperability on a global scale between enterprises.

In particular, the high-level model seeks to reflect the increasingly important role of the Internet as the basis for inter- and intra-enterprise interoperability.

The horizontal dimension of the model in Figure 1 represents *diversity*, and the shape of the model is intended to emphasise the importance of minimum diversity at the interface between the application platform and the communications infrastructure.

This in turn means focusing on the core set of services that can be guaranteed to be supported by every IP-based network, as the foundation on which to build today's interoperable enterprise computing environments.

The Technical Reference Model in Detail

Introduction

The Detailed Technical Reference Model diagram expands on High Level TRM diagram to present the service categories of the Application Platform and the two categories of Application Software.

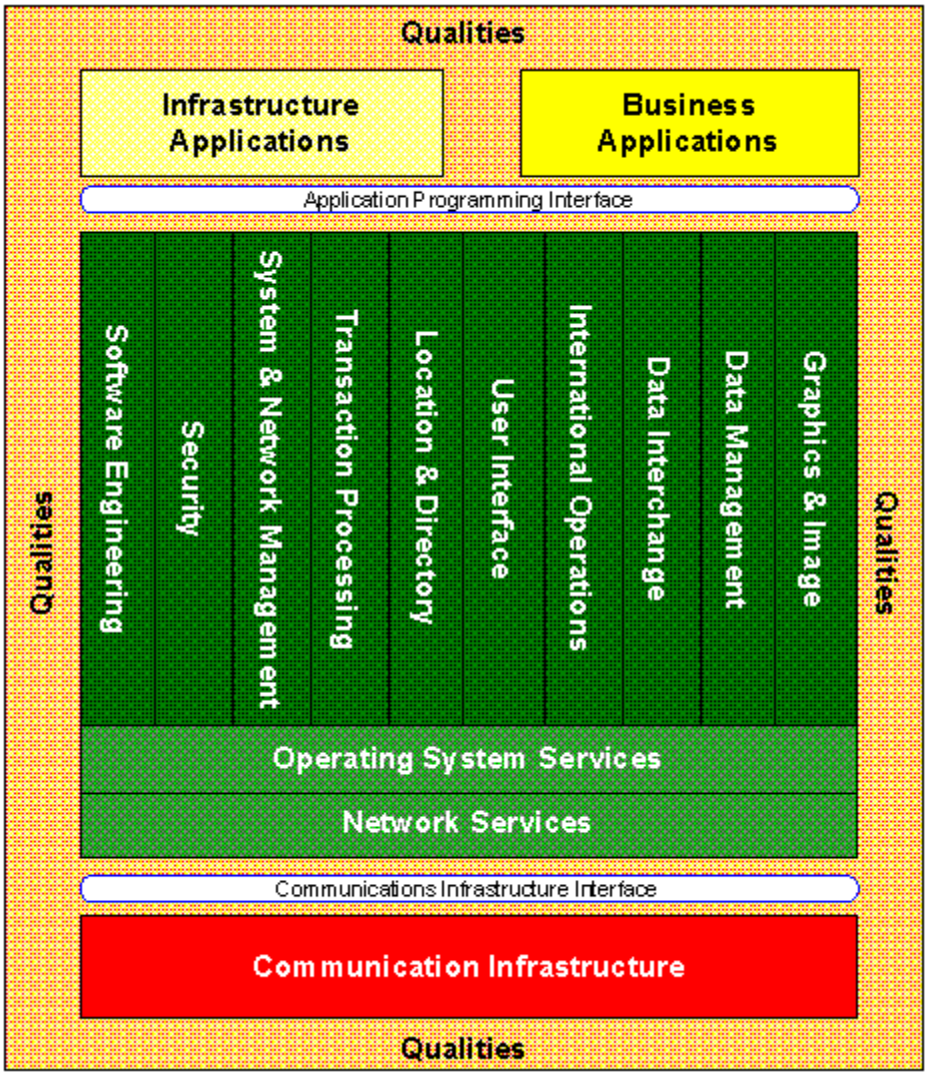


Figure 1: Detailed Technical Reference Model (Showing Service Categories)

The Detailed TRM diagram is only a depiction of the TRM entities: it neither implies nor inhibits interrelationships among them.

IT architectures derived from TOGAF may differ greatly depending on the requirements of the information system. In practice, many architectures will not include all of the services discussed here, and many will include additional services to support application software that is specific to the organization or to its vertical industry.

In building an architecture, users of TOGAF should assess their own requirements and select the services, interfaces, and standards that satisfy their own business needs.

TRM Entities and Interfaces

The following sections discuss in detail each element of the Technical Reference Model illustrated in the Detailed Technical Reference Model diagram. They are dealt with in the following order:

- the three entities:
  - Application Software
  - Application Platform
  - Communications Infrastructure
- and the two interfaces:
  - Application Platform Interface
  - Communications Infrastructure Interface

Application Software

The detailed TRM recognises two categories of application software:

- **Business Applications**, which implement business processes for a particular enterprise or vertical industry. The internal structure of Business Applications relates closely to the specific application software configuration selected by an organization.
- **Infrastructure Applications**, which provide general-purpose business functionality, based on Infrastructure services.

During development of the Technology Architecture, Business Applications and Infrastructure Applications are important sources of requirements for Technology Architecture services, and the selection of standards for the application platform will be influenced strongly by the application software configuration to be supported.

Business Applications

Business Applications are applications that are specific to a particular enterprise or vertical industry. Such applications typically model elements of an enterprise's domain of activity or business processes. Examples of Business Applications might include:

- patient record management services used in the medical industry;
- inventory management services used in the retail industry; or
- geological data modeling services used in the petroleum industry.

Over time, particular Business Applications may become Infrastructure Applications, if they become sufficiently ubiquitous, interoperable, and general purpose to be potentially useful to a broad range of enterprise IT users.

Infrastructure Applications

Infrastructure Applications are applications that have all, or nearly all, of the following characteristics:

- Widespread availability as commercial off-the-shelf software means that it is uneconomic to consider custom implementation;
- User interaction is an important part of the application's function;
- Implementations are based on infrastructure services;
- Implementations may include significant extensions beyond that needed to use the underlying infrastructure services; and
- Interoperability is a strong requirement.

Examples of Applications in this category include:

- Electronic payment and funds transfer services
- Electronic mail client services
- Publish and subscribe

- Intelligent agents
- Calendaring and scheduling services
- Groupware services
- Workflow services
- Spreadsheets
- Presentation software
- Document editing and presentation
- Management applications, performing general-purpose system and network management functions for the System Administrator;
- Software Engineering tools, providing software development functions for systems development staff.

Infrastructure applications have strong dependencies on lower-level services in the architecture. For example, a workflow application may use platform services such as messaging or transaction processing to implement the flow of work among tasks. Similarly, a groupware application is likely to make extensive use of both Data and Communication services for the structure of documents, as well as the mechanics of storing and accessing them.

The Infrastructure Applications by definition are applications that are considered sufficiently ubiquitous, interoperable, and general purpose within the enterprise to be effectively considered as part of the IT infrastructure. Just as Business Applications may over time come to be regarded as Infrastructure Applications, so Infrastructure Applications are normally candidates for inclusion as Infrastructure Services in future versions of an IT architecture.

Application Platform

The "Platform" Concept

The term "platform" is used in many different ways within the IT industry today. Because of the different usages, one often see the term qualified: for example, "Application Platform", "Standardized" and "Proprietary Platforms", "Client" and "Server Platforms", "Distributed Computing Platform", "Portability Platform". Common to all these usages is the idea that someone needs a set of services provided by a particular kind of "platform", and will implement a "higher level" function that makes use of those services.

The TOGAF Technical Reference Model focuses on the Application Platform, and the "higher level function" is the set of Application Software, running on top of the Application Platform, that is needed to address the enterprise's business requirements.

It is important to recognize that the Application Platform in the TOGAF Technical Reference Model is a single, generic, conceptual entity. From the viewpoint of the TOGAF TRM, the Application Platform contains all possible services. In a specific target architecture, the Application Platform will contain only those services needed to support the required functions.

Moreover, the Application Platform for a specific target architecture will typically not be a single entity, but rather a combination of different entities for different, commonly required functions, such as Desk Top Client, File Server, Print Server, Application Server, Internet Server, Data Base Server, etc., each of which will comprise a specific, defined set of services necessary to support the specific function concerned.

It is also important to recognize that many of the real-world IT systems that are procured and used today to implement a Technology Architecture come fully equipped with many advanced services, which are often taken for granted by the purchaser. For example, a typical desktop computer system today comes with software that implements services from most if not all of the service categories of the TOGAF Technical Reference Model. Since the purchaser of such a system often does not consider anything "smaller" than the total bundle of services that comes with the system, that service bundle can very easily become the "platform". Indeed, in the absence of aTechnology Architecture to guide the procurement process, this is invariably what happens. As this process is repeated across an enterprise, different systems purchased for similar functions such as Desk Top Client, Print Server, etc., can contain markedly different bundles of services.

Service bundles are represented in a Technology Architecture in the form of "Building Blocks". One of the key tasks of the IT architect in going from the conceptual Application Platform of the TRM to an enterprise-specific Technology Architecture, is to look beyond the set of real-world "platforms" already in existence in the enterprise. The IT architect must analyse the services actually needed in order to implement an IT infrastructure that meets the enterprise's business requirements in the optimal manner, and define the set of optimal solution building blocks - real-world "platforms" - to implement that architecture.

Extending the TRM

The TOGAF TRM identifies a generic set of platform services, and provides a taxonomy in which these platform services are divided into categories of like functionality. A particular organization may need to augment this set with additional services or service categories which are considered to be generic in its own vertical market segment.

The set of services identified and defined for the Application Platform will change over time. New services will be required as new technology appears and as application needs change.

Interfaces Between Services

In addition to supporting application software through the API, services in the Application Platform may support each other, either by openly specified interfaces which may or may not be the same as the API, or by private, unexposed interfaces. A key goal of architecture development is for service modules to be capable of replacement by other modules providing the same service functionality via the same service API. Use of private, unexposed interfaces among service modules may compromise this substitutability. Private interfaces represent a risk that should be highlighted to facilitate future transition.

Future Developments

The Technical Reference Model deals with future developments in the application platform in two ways. Firstly, as interfaces to services become standardized, functionality which previously formed part of the application software entity migrates to become part of the application platform. Secondly, the TRM may be extended with new service categories as new technology appears.

Examples of functional areas which may fall into Application Platform service categories in the future include:

- **Spreadsheet** functions, including the capability to create, manipulate, and present information in tables or charts. This capability should include fourth-generation-language-like capabilities that enable the use of programming logic within spreadsheets.
- **Decision support** functions, including tools that support the planning, administration, and management of projects.
- **Calculation** functions, including the capability to perform routine and complex arithmetic calculations.
- **Calendar** functions, including the capability to manage projects and co-ordinate schedules via an automated calendar.

A detailed taxonomy of the Application Platform is given later.

Communications Infrastructure

The Communications Infrastructure provides the basic services to interconnect systems and provide the basic mechanisms for opaque transfer of data. It contains the hardware and software elements which make up the networking and physical communications links used by a system, and of course all the other systems connected to the network. It deals with the complex world of networks and the physical communications infrastructure, including switches, service providers and the physical transmission media.

A primary driver in enterprise-wide Technology Architecture in recent years has been the growing awareness of the utility and cost effectiveness of the Internet as the basis of an communications infrastructure for enterprise integration. This is causing a rapid increase in Internet usage and a steady increase in the range of applications linking to the network for distributed operation.

This is considered further under Communications Infrastructure Interface below.

Application Platform Interface

The Application Platform Interface specifies a complete interface between the Application Software and the underlying Application Platform across which all services are provided. A rigorous definition of the interface results in application portability, provided that both platform and application conform to it. For this to work, the API definition must include the syntax and semantics of not just the programmatic interface but also all necessary protocol and data structure definitions.

Portability depends on the symmetry of conformance of both applications and the platform to the architected API. That is, the

platform must support the API as specified, and the application must use no more than the specified API.

The API specifies a complete interface between an application and one or more services offered by the underlying application platform. An application may use several APIs, and may even use different APIs for different implementations of the same service.

Communications Infrastructure Interface

The Communications Infrastructure Interface is the interface between the application platform and the communications infrastructure.

The High-Level Model seeks to reflect the increasingly important role of the Internet as the basis for inter- and intra-enterprise interoperability. The horizontal dimension of the model in the High Level TRM diagram represents *diversity*, and the shape of the model is specifically intended to emphasise minimum diversity at the interface between the application platform and the communications infrastructure.

In particular, the model emphasises the importance of focusing on the core set of services that can be guaranteed to be supported by every IP-based network, as the foundation on which to build today's interoperable enterprise computing environments.

Qualities

Besides the set of components making up the TRM, there is a set of attributes or **qualities** that are applicable across the components. For example, for the management service to be effective, *manageability* must be a pervasive quality of all platform services, applications and communications infrastructure services.

The Detailed TRM diagram captures this concept by depicting the TRM components sitting on a backplane of Qualities.

Another example of a service quality is Security. The proper system-wide implementation of Security requires not only a set of Security services, corresponding to the Security services category shown in the Platform, but also the support (i.e., the 'security awareness') of software in other parts of the TRM. Thus an application might use a security service to mark a file as read-only, but it is the correct implementation of the security quality in the Operating System services which prevents write operations on the file. Security and Operating System services must co-operate in making the file secure.

Qualities are specified in detail during the development of a target architecture. Some qualities are easier than others to describe in terms of standards. For instance, support of a set of locales can be defined to be part of the specification for the *international operation* quality. Other qualities can better be specified in terms of measures rather than standards. An example would be performance, for which standard APIs or protocols are of limited use.

Application Platform - Taxonomy

Principles Service Categories Service Qualities

Basic Principles

The TOGAF Technical Reference Model has two main components:

- a *taxonomy*, which defines terminology, and provides a coherent description of the components and conceptual structure of an information system,
- an associated *Technical Reference Model graphic*, which provides a visual representation of the taxonomy, as an aid to understanding.

This section describes in detail the taxonomy of the TOGAF Technical Reference Model. The aim is to provide a core taxonomy that provides a useful, consistent, structured definition of the application platform entity and is widely acceptable.

No claims are made that the chosen categorization is the only one possible, or that it represents the optimal choice.

Indeed, it is important to emphasise that the use of TOGAF, and in particular the TOGAF Architecture Development Method, is in no way dependent on use of the TOGAF TRM taxonomy. Other taxonomies are perfectly possible, and may be preferable for some organizations.

For example, a different taxonomy may be embodied in the legacy of previous architectural work by an organization, and the organization may prefer to perpetuate use of that taxonomy. Alternatively, an organization may decide that it can derive a more suitable, organization-specific taxonomy by extending or adapting the TOGAF TRM taxonomy.

In the same way, an organization may prefer to depict the TOGAF taxonomy (or its own taxonomy) using a different form of TRM graphic, which better captures legacy concepts and proves easier for internal communication purposes.

However, a consideration to bear in mind in deciding which taxonomy to use, is that the taxonomy of the TOGAF TRM is used in structuring the TOGAF Standards Information Base (SIB), the database of all Open Group endorsed industry standards which is available for populating an architecture. Any differences from the TOGAF TRM taxonomy would need to be catered for when using the TOGAF SIB. (This typically represents an additional overhead, but not a major obstacle.)

Application Platform Service Categories

The major categories of services defined for the application platform are listed below.

Note that *Object services* does not appear as a category in the Technical Reference Model taxonomy. This is because all the individual object services are incorporated into the relevant main service categories. However, the various descriptions are also collected into a single subsection Object-Oriented Provision of Services, in order to provide a single point of reference which shows how object services relate to the main service categories.

To go to the detailed description of a particular service category, follow the hyperlink from the relevant service name.

Alternatively, to browse a number of service category descriptions, you may find it more convenient to load the (Index to) Detailed Platform Taxonomy into the Contents frame opposite.

- Data Interchange Services
  - Document generic data typing and conversion services
  - Document generic data typing and conversion services

    - Graphics data interchange services
    - Specialized data interchange services
    - Electronic data interchange services
    - Fax services
    - Raw graphics interface functions
    - Text processing functions
    - Document processing functions
    - Publishing functions
    - Video processing functions
    - Audio processing functions
    - Multimedia processing functions
    - Media synchronization functions
    - Information presentation and distribution functions
    - Hypertext functions
- Data Management Services
    - Data dictionary/repository services
    - Database management system (DBMS) services
    - Object Oriented Database Management System services
    - File management services
    - Query processing functions
    - Screen generation functions
    - Report generation functions
    - Networking/concurrent access functions
    - Warehousing functions
- Graphics and Imaging Services
    - Graphical object management services
    - Drawing services
    - Imaging functions
- International Operation Services
    - Character sets and data representation services
    - Cultural convention services
    - Local language support services
- Location and Directory Services
    - Directory Services
    - Special Purpose Naming Services
    - Service Location Services
    - Registration Services
    - Filtering Services
    - Accounting Services
- Network Services
    - Data communications services
    - Electronic Mail services
    - Enhanced telephony functions
    - Shared screen functions
    - Video conferencing functions
    - Broadcast functions
    - Mailing list functions
    - Distributed time services
    - Distributed data services
    - Distributed file services
    - Distributed name services
    - Remote process (access) services
    - Remote print spooling and output distribution services
- Operating System Services
    - Kernel operations services
    - Command interpreter and utility services
    - Batch processing services
    - File and directory synchronization services
- Software Engineering Services
    - Programming language services
    - Object code linking services
    - Computer Aided Software Engineering (CASE) environment and tools services
    - Graphical User Interface (GUI) building services
    - Scripting language services
    - Language binding services
    - Run Time Environment services
    - Application Binary Interface services

- Transaction Processing Services
  - Transaction manager services
- User Interface Services
  - Graphical client-server services
  - Display objects services
  - Window management services
  - Dialogue support services
  - Printing services
  - Computer-based training and on-line help services
  - Character-based services
- Security Services
  - Identification and authentication services
  - System entry control services
  - Audit services
  - Access control services
  - Non-repudiation services
  - Security management services
  - Trusted recovery services
  - Encryption services
  - Trusted communication services
- System and Network Management Services
  - User management services
  - Configuration management (CM) services
  - Performance management services
  - Availability and fault management services
  - Accounting management services
  - Security management services
  - Print management services
  - Network management services
  - Backup and Restore services
  - On-line Disk Management services
  - License Management services
  - Capacity Management services
  - Software Installation services
  - Trouble Ticketing functions

Object-Oriented Provision of Services

A detailed description of each of these service categories is given under Object-Oriented Services.

- Object request broker (ORB) services
  - Implementation repository services
  - Installation and activation services
  - Interface repository services
  - Replication services
- Common object services
  - Change management services
  - Collections services
  - Concurrency control services
  - Data interchange services
  - Event management services
  - Externalization services
  - Licensing services
  - Life cycle services
  - Naming services
  - Persistent object services
  - Properties services
  - Query services
  - Relationship services
  - Security services
  - Start-up services
  - Time services
  - Trading services
  - Transaction services

Application Platform Service Qualities

Besides the platform service categories delineated by functional category, service *qualities* affect information system architectures.

A service quality describes a behavior such as adaptability or manageability. Service qualities have a pervasive effect on the operation of most or all of the functional service categories.

The detailed discussion of service qualities is consolidated in a single section Service Qualities in order to provide a coherent perspective.

The service qualities presently identified in the TRM taxonomy are:

- *Availability*, including:
  - *manageability*
  - *serviceability*
  - *performance*
  - *reliability*
  - *recoverability*
  - *locatability*

- *Assurance*, including:
  - *security*
  - *integrity*
  - *credibility*

- *Usability*, including:
  - *International operation*

- *Adaptability*, including:
  - *interoperability*
  - *scalability*
  - *portability*
  - *extensibility*
  - the ability to offer access to services in new paradigms such as object orientation.

Data Interchange Services

Data interchange services provide specialized support for the interchange of information between applications and the external environment. These services are designed to handle data interchange between applications on the same platform and applications on different (heterogeneous) platforms. An analogous set of services exists for object-oriented data interchange, which can be found under data interchange services and externalization services in the paragraph on Object Services.

- **Document generic data typing and conversion** services are supported by specifications for encoding the data (e.g., text, pictures, numerics, special characters) and both the logical and visual structures of electronic documents, including compound documents.
- **Graphics data interchange** services are supported by device-independent descriptions of picture elements for vector-based graphics and descriptions for raster-based graphics.
- **Specialized data interchange** services are supported by specifications that describe data used by specific vertical markets. Markets where such specifications exist include the medical, library, dental, assurance and oil industries.
- **Electronic data interchange** services are used to create an electronic (paperless) environment for conducting commerce and achieving significant gains in quality, responsiveness, and savings afforded by such an environment. Examples of applications that use electronic commerce services include vendor search and selection; contract award; product data; shipping, forwarding, and receiving; customs; payment information; inventory control; maintenance; tax-related data; and insurance-related data.
- **Fax** services are used to create, examine, transmit and/or receive fax images.

The following functional areas are currently supported mainly by application software, but are progressing towards migration into the Application Platform:

- **Raw graphics interface** functions support graphics data file formats such as TIFF, JPEG, GIF and CGM.
- **Text processing** functions, including the capability to create, edit, merge, and format text.
- **Document processing** functions, including the capability to create, edit, merge, and format documents. These functions enable the composition of documents that incorporate graphics, images, and even voice annotation, along with stylized text. Included are advanced formatting and editing functions such as style guides, spell checking, use of multiple columns, table of contents generation, headers and footers, outlining tools, and support for scanning images into bit-mapped formats. Other capabilities include compression and decompression of images or whole documents.
- **Publishing** functions, including incorporation of photographic quality images and color graphics, and advanced formatting and style features such as wrapping text around graphic objects or pictures and kerning (i.e., changing the spacing between text characters). These functions also interface with sophisticated printing and production equipment. Other capabilities include color rendering and compression and decompression of images or whole documents.
- **Video processing** functions, including the capability to capture, compose, edit, compress and decompress video information using formats such as MPEG. Still graphics and title generation functions are also provided.
- **Audio processing** functions, including the capability to capture, compose, edit compress and decompress audio information.
- **Multimedia processing** functions, including the capability to store, retrieve, modify, sort, search, and print all or any combination of the above-mentioned media. This includes support for microfilm media, optical storage technology that allows for storage of scanned or computer produced documents using digital storage techniques, a scanning capability, and data compression and decompression.
- **Media synchronization** functions allow the synchronization of streams of data such as audio and video for presentation purposes.
- **Information presentation and distribution** functions are used to manage the distribution and presentation of information from batch and interactive applications. These functions are used to shield business area applications from how information is used. They allow business area applications to create generic pools of information without embedding controls that dictate the use of that information. Information distribution and presentation functions include the selection of the appropriate formatting functions required to accomplish the distribution and presentation of information to a variety of business area applications and users. Information presentation and distribution functions also include the capability to store, archive, prioritize, restrict, and recreate information.
- **Hypertext** functions support the generation, distribution, location, search and display of text and images either locally or globally. These functions include searching and browsing, hypertext linking and the presentation of multimedia information.

Data Management Services

Central to most systems is the management of data that can be defined independently of the processes that create or use it, maintained indefinitely, and shared among many processes. Data management services include:

- **Data dictionary / repository** services allow data administrators and information engineers to access and modify data about data (i.e., metadata). Such data may include internal and external formats, integrity and security rules, and location within a distributed system. Data dictionary and repository services also allow end users and applications to define and obtain data that are available in the database. Data administration defines the standardization and registration of individual data element types to meet the requirements for data sharing and interoperability among information systems throughout the enterprise. Data administration functions include procedures, guidelines, and methods for effective data planning, analysis, standards, modeling, configuration management, storage, retrieval, protection, validation and documentation. Data dictionaries are sometimes tied to a single database management system, but heterogeneous data dictionaries will support access to different DBMSs. Repositories can contain a wide variety of information including Management Information Bases or CASE-related information. Object oriented systems may provide repositories for objects and interfaces, described under implementation repository services and interface repository services in the paragraph on Object Services.
- **Database management system (DBMS)** services provide controlled access to structured data. To manage the data, the DBMS provides concurrency control and facilities to combine data from different schemas. Different types of DBMS support different data models, including relational, hierarchical, network, object-oriented and flat-file models. Some DBMSs are designed for special functions such as the storage of large objects or multimedia data. DBMS services are accessible through a programming language interface, an interactive data manipulation language interface such as SQL, or an interactive / fourth-generation language interface. Look-up and retrieval services for objects are described separately under query services in the paragraph on Object Services. For efficiency, database management systems often provide specific services to create, populate, move, backup, restore, recover and archive databases, although some of these services could be provided by the general file management capabilities described in the section on Operating System Services or a specific backup service. Some database management systems support distribution of the database, including facilities for remotely updating records, data replication, locating and caching data, and remote management.
- **Object Oriented Database Management System** services provide storage for objects and interfaces to those objects. These services may support the implementation repository, interface repository and persistent object services in the paragraph on Object Services.
- **File management** services provide data management through file access methods including indexed sequential (ISAM) and hashed random access. Flat file and directory services are described in the Operating System Services category.

The following functional areas are currently supported mainly by applications, but are progressing towards migration into the Application Platform:

- **Query processing** functions that provide for interactive selection, extraction, and formatting of stored information from files and databases. Query processing functions are invoked via user-oriented languages and tools (often referred to as fourth generation languages), which simplify the definition of searching criteria and aid in creating effective presentation of the retrieved information (including use of graphics).
- **Screen generation** functions that provide the capability to define and generate screens that support the retrieval, presentation, and update of data.
- **Report generation** functions that provide the capability to define and generate hard copy reports composed of data extracted from a database.
- **Networking / concurrent access** functions that manage concurrent user access to Database Management System (DBMS) functions.
- **Warehousing** functions provide the capability to store very large amounts of data, usually captured from other database systems and to perform on-line analytical processing on it in support of ad-hoc queries.

Graphics and Imaging Services

Graphics services provide functions required for creating, storing, retrieving and manipulating images. These services include:

- **Graphical object management** services, including defining multi-dimensional graphic objects in a form that is independent of output devices, and managing hierarchical structures containing graphics data. Graphical data formats include two- and three-dimensional geometric drawings as well as images.
- **Drawing** services support the creation and manipulation of images with software such as GKS, PEX, PHIGS or OpenGL.

The following functional areas are currently supported mainly by applications, but are progressing towards migration into the Application Platform:

- **Imaging** functions providing for the scan, creation, edit, compression and decompression of images in accordance with recognized image formatting standards. For example: PIKS/IPI, OpenXIL or XIE.

International Operation Services

As a practice, information system developers have generally designed and developed systems to meet the requirements of a specific geographic or linguistic market segment, which may be a nation or a particular cultural market. To make that information system viable, or marketable, to a different segment of the market, a full re-engineering process was usually required. Users or organizations that needed to operate in a multinational or multicultural environment typically did so with multiple, generally incompatible information processing systems.

International operation provides a set of services and interfaces that allow a user to define, select, and change between different culturally related application environments supported by the particular implementation. In general these services should be provided in such a way that internationalization issues are transparent to the application logic.

- **Character sets and data representation** services include the capability to input, store, manipulate, retrieve, communicate, and present data independently of the coding scheme used. This includes the capability to maintain and access a central character-set repository of all coded character sets used throughout the platform. Character sets will be uniquely identified so that the end user or application can select the coded character set to be used. This system-independent representation supports the transfer (or sharing) of the values and syntax, but not the semantics, of data records between communicating systems. The specifications are independent of the internal record and field representations of the communicating systems. Also included is the capability to recognize the coded character set of data entities and subsequently to input, communicate, and present that data.
- **Cultural convention** services provide the capability to store and access rules and conventions for cultural entities maintained in a cultural convention repository called a locale. Locales should be available to all applications. Locales typically include date and currency formats, collation sequences and number formats. Standardized locale formats and APIs allow software entities to use locale information developed by others.
- **Local language support** services provide the capability to support more than one language concurrently on a system. Messages, menus, forms, and on-line documentation can be displayed in the language selected by the user. Input from keyboards that have been modified locally to support the local character sets can be correctly interpreted.

The proper working of international operation services depends on all the software entities involved having the capability to:

- use locales
- switch between locales as required
- maintain multiple active locales
- access suitable fonts

This requires software entities to be written to a particular style and to be designed from the outset with internationalization in mind.

Location and Directory Services

Location and Directory services provide specialized support for locating required resources and for mediation between service consumers and service providers.

The World Wide Web, based on the Internet**,** has created a need for locating **information resources**, which currently is mainly satisfied through the use of search engines. Advancements in the global Internet, and in heterogeneous distributed systems, demand active mediation through broker services that include automatic and dynamic registration, directory access, directory communication, filtration, and accounting services for access to resources.

- **Directory Services** provide services for clients to establish where resources are, and by extension how they can be reached. "Clients" may be humans or computer programs, and "resources" may be a very wide variety of things, such as names, email addresses, security certificates, printers, web pages, etc.
- **Special Purpose Naming Services** provide services that refer names (ordered strings of printable characters) to objects within a given context (namespaces). Objects are typically hierarchically organized within namespaces. Examples are:
  - File systems
  - Security databases
  - Process queues
- **Service Location Services** provide access to "Yellow Pages" services in response to queries based on constraints
- **Registration Services** provide services to register identity, descriptions of the services a resource is providing, and descriptions of the means to access them.
- **Filtering Services** provide services to select useful information from data using defined criteria
- **Accounting Services** provide services such as account open, account update, account balance, account detail, account close, account discounts, account bill/usage tally, account payment settlement based on message traffic, and/or connection time, and/or resource utilization, and/or broker specific (e.g., value based)

Network Services

Network services are provided to support distributed applications requiring data access and applications interoperability in heterogeneous or homogeneous networked environments.

A network service consists of both an interface and an underlying protocol.

- **Data communications**, which include interfaces and protocols for reliable, transparent, end-to-end data transmission across communications networks. Data communications services include both high level functions like file transfer, remote login, remote process execution or PC integration services and low level functions (like a sockets API) giving direct access to communications protocols.
- **Electronic Mail** services including the capability to send, receive, forward, store, display, retrieve, prioritize, authenticate and manage messages. This includes the capability to append files and documents to messages. Messages may include any combination of data, text, audio, graphics, and images and should be capable of being formatted into standard data interchange formats. This service includes the use of directories and distribution lists for routing information, the ability to assign priorities, the use of pre-formatted electronic forms, and the capability to trace the status of messages. Associated services include a summarized listing of incoming messages, a log of messages received and read, the ability to file or print messages, and the ability to reply to or forward messages.
- **Distributed data** services provide access to, and modification of, data/metadata in remote or local databases. In a distributed environment, data not available on the local database are fetched from a remote data server at the request of the local client.
- **Distributed file** services provide for transparent remote file access. Applications have equivalent access to data regardless of the data's physical location. Ancillary services for this function can include: transparent addressing, cached data, data replication, file locking and file logging.
- **Distributed name** services provide a means for unique identification of resources within a distributed computing system. This service is available to applications within the network and provides information that can include: resource name, associated attributes, physical location, and resource functionality. Note that all system resources should be identifiable, in all information systems, by the distributed name. This permits physical location to change, not only to accommodate movement, but also load balancing, system utilization, scaling (adding processors and moving resources to accommodate the increased resources), distributed processing, and all aspects of open systems. Distributed name services include directory services such as X.500 and network navigation services. Distributed name services include ways to locate data objects both by name and by function. The paragraph on Object Services describes equivalent services under naming services and trading services respectively.
- **Distributed time** services provide synchronized time co-ordination as required among distributed processes in different time zones. An equivalent service is described under time services in the paragraph on Object Services.
- **Remote process** (access) services provide the means for dispersed applications to communicate across a computer network. These services facilitate program-to-program communications regardless of their distributed nature or operation on heterogeneous platforms. Remote process services including remote procedure call (RPC) and asynchronous messaging mechanisms underpin client/server applications.
- **Remote print spooling and output distribution** services provide the means for printing output remotely. The services include management of remote printing including printer and media selection, use of forms, security and print queue management.

The following functional areas are currently supported mainly by application software, but are progressing towards migration into the Application Platform:

- **Enhanced telephony** functions, including call set-up, call co-ordination, call forwarding, call waiting, programmed directories, teleconferencing, automatic call distribution (useful for busy customer service categories), and call detail recording.
- **Shared screen** functions that provide audio teleconferencing with common workstation windows between two or more users. This includes the capability to refresh windows whenever someone displays new material or changes an existing display. Every user is provided with the capability to graphically annotate or modify the shared conference window.
- **Video conferencing** functions that provide two-way video transmission between different sites. These functions include call set-up, call co-ordination, full motion display of events and participants in a bi-directional manner, support for the management of directing the cameras, ranging from fixed position, to sender directed, to receiver directed, to automated sound pickup.
- **Broadcast** functions that provide one-way audio or audio/video communications functions between a sending location and multiple receiving locations or between multiple sending and receiving locations.
- **Mailing list** functions that allow groups to participate in conferences. These conferences may or may not occur in real time. Conferees or invited guests can drop in or out of conferences or subconferences at will. The ability to trace the exchanges is provided. Functions include exchange of documents, conference management, recording facilities, and search and retrieval capabilities.

Operating System Services

Operating system services are responsible for the management of platform resources, including the processor, memory, files, and input and output. They generally shield applications from the implementation details of the machine. Operating system services include:

- **Kernel operations** provide low-level services necessary to:
  - create and manage processes and threads of execution
  - execute programs
  - define and communicate asynchronous events
  - define and process system clock operations
  - implement security features
  - manage files and directories, and
  - control input/output processing to and from peripheral devices.
- Some kernel services have analogues described in the paragraph on Object Services, such as concurrency control services.
- **Command interpreter and utility services** include mechanisms for services at the operator level, such as:
  - comparing, printing, and displaying file contents
  - editing files
  - searching patterns
  - evaluating expressions
  - logging messages
  - moving files between directories
  - sorting data
  - executing command scripts
  - local print spooling
  - scheduling signal execution processes, and
  - accessing environment information.
- **Batch processing** services support the capability to queue work (jobs) and manage the sequencing of processing based on job control commands and lists of data. These services also include support for the management of the output of batch processing, which frequently includes updated files or databases and information products such as printed reports or electronic documents. Batch processing is performed asynchronously from the user requesting the job.
- **File and directory synchronization** services allow local and remote copies of files and directories to be made identical. Synchronization services are usually used to update files after periods of off line working on a portable system.

Software Engineering Services

The functional aspect of an application is embodied in the programming languages used to code it. Additionally, professional system developers require tools appropriate to the development and maintenance of applications. These capabilities are provided by software engineering services, which include:

- **Programming language** services provide the basic syntax and semantic definition for use by a software developer to describe the desired application software function. Shell and executive script language services enable the use of operating system commands or utilities rather than a programming language. Shells and executive scripts are typically interpreted rather than compiled, but some operating systems support compilers for executive scripts. In contrast, some compilers produce code to be interpreted at runtime. Other tools in this group include source code formatters and compiler compilers.
- **Object code linking** services provide the ability for programs to access the underlying application and operating system platform through APIs that have been defined independently of the computer language. It is used by programmers to gain access to these services using methods consistent with the operating system and specific language used. Linking is operating system dependent, but language independent.
- **Computer Aided Software Engineering (CASE) environment and tools** services include systems and programs that assist in the automated development and maintenance of software. These include, but are not limited to, tools for requirements specification and analysis, for design work and analysis, for creating, editing, testing and debugging program code, for documenting, for prototyping, and for group communication. The interfaces among these tools include services for storing and retrieving information about systems and exchanging this information among the various components of the system development environment. An adjunct to these capabilities is the ability to manage and control the configuration of software components, test data, and libraries, to record changes to source code or to access CASE repositories. Other language tools include code generators and translators, artificial intelligence tools and tools like the UNIX command "make", which uses knowledge of the interdependencies between modules to recompile and link only those parts of a program which have changed.
- **Graphical User Interface (GUI) building** services assist in the development of the Human Computer Interface elements of applications. Tools include services for generating and capturing screen layouts, and for defining the appearance, function, behavior and position of graphical objects.
- **Scripting language** services provide interpreted languages which allow the user to carry out some complicated function in a simple way. Application areas served by special purpose scripting languages include calculation, graphical user interface development and development of prototype applications.
- **Language binding** services provide mappings from interfaces provided by programming languages onto the services provided by the Application Platform. In many cases the mapping is straightforward since the platform supplies analogous services to those expected by the application. In other cases the language binding service must use a combination of Application Platform services to provide a fully functional mapping.
- **Run Time Environment** services provide support for application software at run time. This support includes locating and connecting dynamically linked libraries, or even emulation of an operating environment other than the one which actually exists.
- **Application Binary Interface** services provide services that make the Application Platform comply with defined application binary interface standards.

Transaction Processing Services

Transaction Processing (TP) services provide support for the on-line processing of information in discrete units called transactions, with assurance of the state of the information at the end of the transaction. This typically involves predetermined sequences of data entry, validation, display, and update or inquiry against a file or database. It also includes services to prioritize and track transactions. Transaction processing services may include support for distribution of transactions to a combination of local and remote processors.

A transaction is a complete unit of work. It may comprise many computational tasks, which may include user interface, data retrieval and communications. A typical transaction modifies shared resources. Transactions must also be able to be rolled back (that is, undone) if necessary, at any stage. When a transaction is completed without failure, it is committed. Completion of a transaction means either commitment or rollback.

Typically a transaction processing service will contain a transaction manager, which links data entry and display software with processing, database and other resources to form the complete service.

The sum of all the work done anywhere in the system in the course of a single transaction is called a global transaction. Transactions are not limited to a single application platform.

- **Transaction manager** services, which allow an application to demarcate transactions, and direct their completion. Transaction manager services include:
  - starting a transaction
  - co-ordination of recoverable resources involved in a transaction
  - committing or rolling back transactions
  - controlling timeouts on transactions
  - chaining transactions together
  - monitoring transaction status
- Some transaction manager services have equivalents described in the paragraph on Object Services, under the heading transaction services.

User Interface Services

User interface services define how users may interact with an application. Depending on the capabilities required by users and the applications, these interfaces may include the following:

- **Graphical client-server** services that define the relationships between client and server processes operating graphical user interface displays, usually within a network. In this case, the program that controls each display unit is a server process, while independent user programs are client processes that request display services from the server.
- **Display objects** services that define characteristics of display elements such as color, shape, size, movement, graphics context, user preferences, font management and interactions among display elements.
- **Window management** services that define how windows are created, moved, stored, retrieved, removed and related to each other.
- **Dialogue support** services translate the data entered for display to that which is actually displayed on the screen (e.g., cursor movements, keyboard data entry, external data entry devices).
- **Printing** services support output of text and/or graphical data, including any filtering or format conversion necessary. Printing services may include the ability to print all or part of a document, to print and collate more than one copy, to select the size and orientation of output, to choose print resolution, colors and graphical behavior, to specify fonts and other characteristics.
- **Computer-based training and on-line help** services provide an integrated training environment on user workstations. Training is available on an as-needed basis for any application available in the environment. Electronic messages are provided at the stroke of a key from anywhere within the application. This includes tutorial training on the application in use and the availability of off-line, on-site interactive training.
- **Character-based** services, which deal with support for non-graphical terminals. Character-based services include support for terminal type-independent control of display attributes, cursor motions, programmable keys, audible signals and other functions.

The services associated with a window system include the visual display of information on a screen that contains one or more windows or panels, support for pointing to an object on the screen using a pointing device such as a mouse or touch-screen, and the manipulation of a set of objects on the screen through the pointing device or through keyboard entry. Other user interfaces included are industrial controls and virtual reality devices.

Security Services

Security services are necessary to protect sensitive information in the information system. The appropriate level of protection is determined based upon the value of the information to the business area end users and the perception of threats to it.

To be effective, security needs to be made strong, must never be taken for granted, and must be designed into an architecture and not bolted on afterwards. Whether a system is standalone or distributed, security must be applied to the whole system. It must not be forgotten that the requirement for security extends not only across the range of entities in a system but also through time.

In establishing a security architecture, the best approach is to consider what is being defended, what value it has, and what the threats to it are. The principal threats to be countered are:

- loss of confidentiality of data
- unavailability of data or services
- loss of integrity of data
- unauthorized use of resources

Counters to these threats are provided by the following services:

- **Identification and authentication** services provide:
  - identification, accountability and audit of users and their actions
  - authentication and account data
  - protection of authentication data
  - active user status information
  - password authentication mechanisms
- **System entry control services** provide:
  - warning to unauthorized users that the system is security-aware
  - authentication of users
  - information, displayed on entry , about previous successful and unsuccessful login attempts
  - user initiated locking of a session preventing further access until the user has been re-authenticated
- **Audit** services provide authorized control and protection of the audit trail, recording of detailed information security-relevant events and audit trail control, management and inspection.
- **Access control** services provide:
  - access control attributes for subjects (such as processes) and objects like files
  - enforcement of rules for assignment and modification of access control attributes
  - enforcement of access controls
  - control of object creation and deletion, including ensuring that reuse of objects does not allow subjects to accidentally gain access to information previously held in the object
- Access control services also appear under the security services heading in the paragraph on Object Services.
- **Non-repudiation** services provide proof that a user carried out an action, or sent or received some information, at a particular time. Non-repudiation services also appear under the security services heading in the paragraph on Object Services.
- **Security management** services provide secure system set-up and initialization, control of security policy parameters, management of user registration data and system resources and restrictions on the use of administrative functions.
- **Trusted recovery** services provide recovery facilities such as restoring from backups in ways that do not compromise security protection.
- **Encryption** services provide ways of encoding data such that it can only be read by someone who possesses an appropriate key, or some other piece of secret information. As well as providing data confidentiality for trusted communication, encryption services are used to underpin many other services including identification and authentication, system entry control, and access control services.
- **Trusted communication** services provide
  - a secure way for communicating parties to authenticate themselves to each other without the risk of an eavesdropper subsequently masquerading as one of the parties.
  - a secure way of generating and verifying check values for data integrity.
  - data encipherment and decipherment for confidentiality and other purposes.
  - a way to produce an irreversible hash of data for support of digital signature and non-repudiation functions.
  - generation, derivation, distribution, storage, retrieval and deletion of cryptographic keys.

Security services require other software entities to co-operate in:

- access control for resources managed by the entity
- accounting and audit of security relevant events
- the import and export of data

- and potentially all other security services depending on the particular implementation approach

Security services are one category where a wide view is particularly important, as a chain is only as strong as its weakest link. This is one category of services where the external environment has critical implications on the application platform. For instance, the presence of a firewall may provide a single point of access onto a network from the outside world, making it possible to concentrate access control in one place and relax requirements behind the firewall.

System And Network Management Services

Information systems are composed of a wide variety of diverse resources that must be managed effectively to achieve the goals of an open system environment. While the individual resources (such as printers, software, users, processors) may differ widely, the abstraction of these resources as managed objects allows for treatment in a uniform manner. The basic concepts of management, including operation, administration, and maintenance may then be applied to the full suite of information system components along with their attendant services.

System and network management functionality may be divided in several different ways; one way is to make a division according to the management elements that generically apply to all functional resources. This division reduces to:

- **User management** services provide the ability to maintain a user's preferences and privileges.
- **Configuration management (CM)** services address four basic functions:
  - identification and specification of all component resources
  - control, or the ability to freeze configuration items, changing them only through agreed processes
  - status accounting of each configuration item
  - verification through a series of reviews to ensure conformity between the actual configuration item and the information recorded about it
- These CM services include: Processor CM, Network CM, Distributed System CM, Topology CM and Application CM. Processor CM takes a platform-centric approach. Network CM and Distributed System CM services allow remote systems to be managed and monitored including the interchange of network status. Topology CM is used to control the topology of physical or logical entities that are distributed. Application CM focuses on applications. Configuration management also appears as change management services in the paragraph on Object Services.
- **Performance management** services monitor performance aspects of hardware, platform and application software, and network components and provide ways to tune the system to meet performance targets.
- **Availability and fault management** services allow a system to react to the loss or incorrect operation of system components including hardware, platform software, and application software.
- **Accounting management** services provide the ability to cost services for charging and reimbursement.
- **Security management** services control the security services in accordance with applicable security policies.
- **Print management services** provide the ability to manage both local and remote print spooling services.
- **Network management** services comprise elements of all the services described above, but are often treated as a separate service.
- **Backup and Restore** services provide a multi-level storage facility to ensure continued data security in case of component or sub-system failure.
- **On-line Disk Management** services manage the utilization of disk storage against threshold values and invoke corrective action.
- **License Management** services support the effective enforcement of software license agreements. Licensing services for objects are described under the licensing services heading in the paragraph on Object Services.
- **Capacity Management** services address three basic functions:
  - capacity management analyzing current and historic performance and capacity
  - workload management to identify and understand applications that use the system
  - capacity planning to plan required hardware resources for the future
- **Software Installation** services support distribution, installation, removal, relocation, activation and automatic update of software or data packages from transportable media or over networks. Similar services for objects are described under installation and activation services in the paragraph dealing with Object Services.

The following functional areas are currently supported mainly by application software, but are progressing towards migration into the Application Platform:

- **Trouble Ticketing** services support the generation, processing and tracking of problem reports. Trouble ticketing is a term originating in the telecommunications world, referring to the ability to pass fault reports both within and between telecommunications service providers. In this environment, faults are often found by a customer of one provider, while the cause of the problem lies within the administrative domain of another provider. Trouble ticketing is a common service that may be useful to an increasing range of applications if the necessary work is done to extend it from telecommunications into wider areas of distributed applications such as electronic mail.

This breakout of system and network management services parallels the breakout of emerging OSI network management, thereby presenting an overall coherent framework that applies equally to whole networks and the individual nodes of the networks.

One important consideration of the standards supporting the services in this category is that they should not enforce specific management policies but rather enable a wide variety of different management policies to be implemented, selected according to the particular needs of the end-user installations.

System and network management services require the co-operation of other software entities in:

- providing status information
- notifying events
- responding to management instructions

Object-Oriented Provision of Services

This subsection shows how services are provided in an object-oriented manner. 'Object Services' does not appear as a category in the Technical Reference Model since all the individual object services are incorporated as appropriate in the given service categories.

An object is an identifiable, encapsulated entity that provides one or more services that can be requested by a client. Clients request a service by invoking the appropriate method associated with the object, and the object carries out the service on the client's behalf. Objects provide a programming paradigm that can lead to important benefits, including:

- increased modularity
- a reduction in errors, and
- ease of debugging

Object management services provide ways of creating, locating and naming objects, and allowing them to communicate in a distributed environment. The complete set of object services identified so far is listed below for the sake of completeness. Where a particular object service is part of a more generally applicable service category, a pointer to the other service category is given. Object services include:

- **Object request broker (ORB)** services, which enable objects to transparently make and receive requests and responses in a distributed environment. ORB services include:
  - **Implementation repository** services support the location and management of object implementations. The services resemble those provided by the data dictionary/repository services in the Data Management service category.
  - **Installation and activation** services provide ways to distribute, install, activate and relocate objects. This corresponds to the software installation services in the System and Network Management service.
  - **Interface repository** services support the storage and management of information about interfaces to objects. The services resemble those provided by the data dictionary/repository services in the Data Management service category.
  - **Replication** services support replication of objects in distributed systems, including management of consistency between the copies.
- **Common object services**, which provide basic functions for using and implementing objects. These are the services necessary to construct any distributed application. Common object services include:
  - **Change management** services provide for version identification and configuration management of object interfaces, implementations and instances. This corresponds to the configuration management services described in the System and Network Management service category.
  - **Collections** services provide operations on collections of objects, such as lists, trees, stacks or queues. Services include establishing, adding objects to or removing them from collections, testing set membership, forming unions and intersections of sets and so on.
  - **Concurrency control** services enable multiple clients to co-ordinate their access to shared resources. Synchronization like this is normally provided using kernel services provided in the Operating System service category.
  - **Data interchange** services support the exchange of visible state information between objects. Depending on the kind of object involved, this corresponds to one or more of the services provided in the Data Interchange service category.
  - **Event management** services provide basic capabilities for the management of events, including asynchronous events, event "fan-in",notification "fan-out" and reliable event delivery.
  - **Externalization** services define protocols and conventions for externalizing and internalizing objects. Externalizing means recording the object state in a stream of data, and internalizing means recreating an object state from a data stream. This is one example of the information presentation and distribution functions in the Data Interchange service category.
  - **Licensing** services support policies for object licensing, and measurement and charging for object use. This corresponds to the license management services of the System and Network Management service category.
  - **Life cycle** services define conventions for creating, deleting, copying and moving objects. The creation of objects is defined in terms of factory objects, which are objects that create other objects.
  - **Naming** services provide the ability to bind a name to an object, and to locate an object by its name. This is analogous to the distributed name service described in the Distributed Computing service category.
  - **Persistent object** services provide common interfaces for retaining and managing the persistent state of objects. Objects are often stored in an object oriented database management system, described as one of the Data Management services.
  - **Properties** services support the creation, deletion, assignment and protection of dynamic properties associated with objects.
  - **Query** services support indexing and query operations on collections of objects that return a subset of the collection. This is similar to database lookup, a part of the database management system functions of the Data Management service category.

- **Relationship** services allow relationships between objects, such as ownership or containment, to be explicitly represented as objects.
- **Security** services support access control on objects and non-repudiation of operations on objects. Access control is defined as a Security service. Non-repudiation, which is also a Security service, provides proof that an action was carried out by a particular user at a particular time.
- **Start-up** services support automatic start-up and termination of object services at ORB start-up or termination.
- **Time** services support synchronization of clocks in a distributed system. This is the same as the distributed time service in the Distributed Computing service category.
- **Trading** services allow clients to locate objects by the services the objects provide, rather than by name. This is similar to the distributed name service in the Network service category.
- **Transaction** services provide facilities for grouping operations into atomic units, called transactions, with the certainty that a transaction will be carried out in its entirety or not at all. This corresponds to some of the transaction manager services in the Transaction Processing service category.

Application Platform Service Qualities

Principles Taxonomy

Principles

Besides the platform service categories delineated by functional category, service *qualities* affect information system architectures. A service quality describes a behavior such as adaptability or manageability. Service qualities have a pervasive effect on the operation of most or all of the functional service categories.

The discussion of service qualities is consolidated here in a single location in order to provide a coherent perspective.

In general a requirement for a given level of a particular service quality requires one or more functional service categories to co-operate in achieving the objective. Usually this means that the software building blocks that implement the functional services contain software which contributes to the implementation of the quality.

For the quality to be provided properly, all relevant functional services must have been designed to support it. Service qualities may also require support from software in the Application Software entity and the External Environment as well as the Application Platform.

In some cases, a service quality affects each of the service categories in a similar fashion, while in other cases, the service quality has a unique influence on one particular service category. For instance, international operation depends on most of the service categories in the same way, both providing facilities and needing their co-operation for localization of messages, fonts and other features of a locale, but it may have a more profound effect on the software engineering services, where facilities for producing internationalized software may be required.

During the process of architecture development, the architect must be aware of the existence of qualities and the extent of their influence on the choice of software building blocks used in implementing the architecture. The best way of making sure that qualities are not forgotten is to create a quality matrix, describing the relationships between each functional service and the qualities that influence it.

Taxonomy of Service Qualities

The service qualities presently identified are:

- *Availability* (the degree to which something is available for use), including:
  - *manageability*, the ability to gather information about the state of something and to control it
  - *serviceability*, the ability to identify problems and take corrective action such as to repair or upgrade a component in a running system
  - *performance*, the ability of a component to perform its tasks in an appropriate time
  - *reliability*, or resistance to failure
  - *recoverability*, or the ability to restore a system to a working state after an interruption
  - *locatability*, the ability of a system to be found when needed

- *Assurance*, including:
  - *security*, or the protection of information from unauthorized access
  - *integrity*, or the assurance that data has not been corrupted
  - *credibility*, or the level of trust in the integrity of the system and its data

- *Usability*, or ease of operation by users, including:
  - *International operation*, including multilingual and multicultural abilities

- *Adaptability*, including:
  - *interoperability*, whether within or outside the organization (for instance interoperability of calendaring or scheduling functions may be key to the usefulness of a system)
  - *scalability*, the ability of a component to grow or shrink its performance or capacity appropriately to the demands of the environment in which it operates

- *portability*, of data, people, applications, and components
- *extensibility*, or the ability to accept new functionality
- the ability to offer access to services in new paradigms such as object orientation.

Introduction to the Standards Information Base (SIB)

Role Access

Role of the Standards Information Base

Previous sections of Part III have set the TOGAF Foundation Architecture in context to the Architecture Continuum, and described in detail one part of it, the TOGAF Technical Reference Model. This section describes the other part of the TOGAF Foundation Architecture, the Standards Information Base.

What is the Standards Information Base?

The Open Group's Standards Information Base is a database of facts and guidance about information systems standards. The standards to which it refers come from many sources: from formal standards bodies such as ISO or IEEE; from authoritative standards makers such as the Internet Society; and from other consortia, like the World Wide Web Consortium and the Object Management Group.

What is it for?

The Standards Information Base has three main uses:

- Architecture development: For an organization that is creating an architecture for its information systems, the Standards Information Base provides a valuable source of information about standards that may be used to populate the architecture.
- Acquisition / Procurement: An organization that is planning a procurement (whether or not based on an architecture) will find that the Standards Information Base can help ensure that the procurement gives a clear statement of technical requirements, with an assurance of conformance.
- General information : Finally, it can simply be a source of information about relevant IT standards, for use by anyone at any time.

The standards listed in the various tables are all Open Group standards - that is, standards endorsed by The Open Group as fit for purpose in architecture specification and procurement. They have been approved by the members of The Open Group as appropriate for use in Architecture and Procurement.

How is it Used in Architecture Development?

The entries in the Standards Information Base are linked either to other Open Group databases and resources, in particular those relating to Product Standards and Registered Products, or, where relevant, to the web sites of other organizations.

In this way, the SIB provides the architect with a gateway to a uniquely powerful set of tools for defining the standards that an architecture is to mandate, and for checking the availability in the market place of products guaranteed to conform to those standards.

In the context of TOGAF, the Standards Information Base can be used to dynamically generate lists, structured according to the TOGAF Technical Reference Model taxonomy, of the standards endorsed by The Open Group for use in open systems architectures.

For a detailed explanation of how the standards generated in this way are used, refer to *Part II Architecture Development Method*, which describes how to use the complete TOGAF Foundation Architecture as a basis for defining (by service) all the standards that make up the target Technology Architecture, and all the software building blocks that will be used to implement it.

Accessing the Standards Information Base

Originally held as part of the TOGAF document set, the Standards Information Base is now held in a database with web-enabled user access.

- The Standards Information Base home page is the usual starting point. You may want to bookmark the home page after following the hyperlink. It contains the two direct links shown below, plus a link to the detailed explanation on Using the SIB.
- You can access the SIB selectively through a search interface, according to your own defined criteria.
- Alternatively, you can view the entire SIB.

Open Group Standards

Overview Criteria for Inclusion in the SIB Technical Processes Product Standards Open Brand

Overview

Besides the fact that they are structured and made accessible in the Standards Information Base, using Open Group standards to populate an architecture has a number of distinct advantages for the architect, and for the architect's organization:

- Individual Open Group standards are developed by proven technical processes that establish strong industry consensus behind a standard.
- The Open Group adds value to individual standards by integrating them into sets, known as Product Standards, which are designed to be used together.
- Open Group Product Standards are supported by a unique brand - the Open Brand - which in turn is supported by an extensive set of conformance tests, and which guarantees the conformance to a Product Standard of real products available in the market.

Information on the different types of Open Group standards and publications is available from the Open Group Publications web site.

The Open Group makes all of its standards published since January 1997 freely available in HTML form. It also maintains a current list of the Open Group standards and other Open Group publications that are accessible in this way.

Criteria for Inclusion in the Standards Information Base

The content of the Standards Information Base is determined by a consensus technical process. In order to be included in the SIB, a standard needs to be recommended by a Program Group that has responsibility in the relevant technical area. At the same time, confirmation is sought that each new entry acceptably meets the criteria listed below.

Criterion	Explanation
Non-discriminatory Implementation	If the specification is taken from a existing product source licensable from a single vendor only, then implementations should be available to all companies on an non-discriminatory basis. This includes pricing and licensing conditions.
Availability of Dependencies	If an implementation requires other products or services to be usable (e.g... protocols), the complementary products or services must be publicly available, specified by The Open Group or obtainable from multiple sources.
Availability of Implementations	Commercial availability of implementations.
Completeness of specification	The interfaces to be adopted must be specified sufficiently that a conformant product may be implemented (and usable) using only: • The specification itself • Products or services (e.g. protocols) that are publicly available or obtainable from multiple sources on a non-discriminatory basis • Formal standards from accredited standards development organisations • Other Open Group originated or adopted specifications • Other freely available information
Freedom to Develop	Freedom for anyone to develop a practical product which either supports or utilises the same specification, subject to the need to license any predisclosed patents
Future Access	The contributor to give The Open Group access to all future versions of the material with no obligation on The Open Group to adopt them.

Immunity from Liability	An assurance that a person developing a product in accordance with the specification is immune from any liability to the contributor of the material in respect of the use by him or his customers of such material, other than through failure to properly license predisclosed patents.
Market Need	Evidence that there is a market need for the interface. For example: • Customer requirement • Requirement derived from The Open Group's product management activities • Vendor submitted evidence
No proprietary lock in	The interfaces to be adopted are complete, in that it is not necessary to use any additional interfaces, retained as proprietary, in order to create commercially usable products.
Non-Discriminatory Patents	If the interfaces to be adopted are covered by patents, such patents must be licensed by their owners on a reasonable and non-discriminatory basis.
Other Activities	Understanding of activities in this area in other consortia and official standards bodies
Specification Availability	The availability of a high quality specification upon which Open Group activities can be based
Test Suite Availability	The availability of a test suite which could be used as the basis for conformance testing

The Open Group Technical Processes

The technical processes by which The Open Group produces its standards are long established and widely accepted throughout the industry.

They are also central to The Open Group's mission of delivering greater business efficiency by making it easier to integrate information technology across the enterprise.

By means of these processes, The Open Group both:

- defines the industry consensus standards necessary to support the deployment of interoperable infrastructures and business applications, and
- promotes the availability in the market place of products that conform to those standards.

As a result, enterprise architects can design, and customer organizations procure, multi-vendor information technology solutions that both meet the business needs, and integrate within and between enterprises, with reduced time, cost, and risk.

More about the Open Group technical processes.

Product Standards

All individual standards adopted through The Open Group's technical processes are known as *Open Group Standards*, and are documented in its Standards Information Base.

The Open Group adds value to the standards in its Standards Information Base by integrating related standards into sets, known as *Product Standards* (analogous to "solution building blocks" in TOGAF terms, or "technical profiles" in the formal standards world[1]), which are designed to be used together.

This is a recursive process, the goal being the definition of "procurement ready" Product Standards, whose functionality is strongly related to the needs of customers, and whose scope and structure is strongly related to real products that can be procured in the open market.

More about Open Group Product Standards.

The Open Brand

The Open Brand is signified by the green "X" Device. It can be associated with, and used in relation to, IT systems that have been registered with The Open Group as being fully conformant with one or more Product Standards.

Anyone wishing to register a product, or products, and use the "X" Device, must first sign the Open Brand Trademark License Agreement and thereby "warrant and represent" that any products they register will fully conform to the identified Product Standard(s), and continue to do so.

Thus the Open Brand, when associated with a vendor's product, communicates clearly and unambiguously to a procurer that the software bearing the brand correctly implements the corresponding Open Group Product Standard.

Customers specifying the Open Brand in their procurements can therefore be certain that the branded products they buy will conform to the Product Standard at the time of purchase, and will continue to do so for as long as the product remains registered.

More about the Open Brand

_____

[1]For example, as defined in ISO/IEC TR 14252 (IEEE Std 1003.0) and IEEE Std 1003.23

Using the Standards Information Base and Linked Resources

Introduction

The entries in the Standards Information Base are linked either to other Open Group databases and resources, in particular those relating to Product Standards and Registered Products, or, where relevant, to the web sites of other de facto and de jure standards organizations.

In this way, the SIB provides the architect with a gateway to a uniquely powerful set of tools for defining the standards that an architecture is to mandate, and for checking the availability in the market place of products guaranteed to conform to those standards.

Examples

Getting Started

First, go to the Standards Information Base home page. (It opens in a new window, to enable you to keep these instructions visible.) You may want to bookmark the home page after following the hyperlink.

The home page provides three hyperlinks:

- **Search it** - generates a form to guide the search for specific standards or sets of standards
- **View it** - generates a listing of the entire Standards Information Base, structured according to the TOGAF Technical Reference Model taxonomy
- **Learn more about it** - links to this page in the TOGAF document

The following examples are intended to provide an initial guide through the different resources available, and to provide readers with an understanding of the wide range of information available, starting from the SIB home page.

The examples are far from exhaustive, and readers are encouraged to investigate further for themselves after following the examples.

Example 1 - The Entire SIB

From the Standards Information Base home page, click the **View it** hyperlink. The system may take some time to respond - it is compiling the entire contents of the Standards Information Base.

The result is a series of tables, one for each of the major service categories in the TOGAF Technical Reference Model taxonomy. The hyperlinks at the head of the page provide links to the start of each service category table.

- If you want to, save this page for off-line viewing later. (Size is ~200K.)

As you can see, the SIB contains hyperlinks to the web sites of many different standards organizations, both de jure and de facto.

The standards listed in the various tables are all Open Group standards - that is, standards endorsed by The Open Group as fit for purpose in architecture specification and procurement.

Example 2 - Referenced Standards

The majority of the standards listed in the SIB have been developed and published by The Open Group itself. However, there are also many *Referenced Standards* - standards developed and published by other organizations, and referenced from the SIB.

Data Interchange in particular is an area where The Open Group has elected not to duplicate the excellent work done in other organizations, and instead has adopted from those organizations the relevant standards with demonstrated industry consensus.

1. From the Standards Information Base home page, click the Search it hyperlink. The search form appears, which allows you to specifiy several criteria to help you find what you want.
2. From the Service Category drop-down box, select the Data Interchange Services category (but don't click the Search button just yet).
3. Click the Service drop-down box. You will see that it now lists all the individual services within the Data Interchange category.
4. Select Hypertext, and then click the Search button.
5. When the search results appear, look under the Reference and Status column and locate the HTML 4.0 entry. Click on Details in the Other Views column. This displays the full SIB entry for HTML 4.0.
6. In the full SIB entry, click the hyperlink shown against URL (you could also have clicked the HTML 4.0 hyperlink in the previous page). This takes you to the HTML 4.0 Specification on the W3C public web site.
7. Go back to the Open Group web site, and go back again to the search results for Hypertext.
8. Locate the entry for IETF RFC 2068 and click the Details link. This displays the full SIB entry for IETF RFC 2068. The full entry explains the relationship of HTTP/1.1 to HTTP/1.0, and (under See Also) gives a link to the SIB entry for the corresponding IETF RFC (1945).
9. Again, click the hyperlink shown against URL. This takes you to the text of the HTTP/1.1 specification as approved by the Internet Engineering Task Force.
10. Go back to the Open Group web site, and go back again to the search results for Hypertext. Browse the remaining links at your leisure. Note that all organizations whose standards are referenced from the SIB make the full text freely available.

Example 3 - Open Group Technical Standards

Now we will look at the facilities available in the SIB for the Technical Standards developed and published by The Open Group itself.

1. From the Standards Information Base home page, click the Search it hyperlink.
2. When the search form appears, under Service Category select System and Network Management Services (but don't click the Search button just yet).
3. Click the Service drop-down box. You will see that it now lists all the individual services within the System and Network Management category.
4. Leave the Any entry in place under Service, and press the Search button. The table displayed as a result shows all the standards in the SIB under the System and Network Management Services category.
5. Search down the table under the Reference and Status column to locate the entry for C701(XDSA). This is the 'Systems Management: Distributed Software Administration (XDSA)' Technical Standard. This time, click on the C701 hyperlink.
6. This links to the Open Group Publications database (a separate database from the SIB). The next table displayed shows full publication details of the Technical Standard, and links to a range of further information. Options at this point:
    - Press the SYSTEMS MANAGEMENT hyperlink. This provides a page of structured hyperlinks to all Open Group Systems Management publications (not just standards).
    - Go back to the previous page. The two hyperlinks under Availability offer two different ways of obtaining a copy of the standard, including free access to an HTML version; and an order form for a hard copy version. Investigate these links at your leisure.

Example 4 - Open Group Product Standards

This time we will look at the facilities for the Product Standards developed and published by The Open Group.

1. From the Standards Information Base home page, click the Search it hyperlink.
2. When the search form appears, under Service Category select Operating System Services, and press the Search button.
3. The table displayed as a result shows all the standards in the SIB under the Operating System Services category. As you can see, there are a lot!
4. Search down the table under the Reference and Status column to locate the entry X/Open XX. This is the UNIX 98 Product Standard. Click on the X/Open XX hyperlink.
5. The next table displayed shows all the products registered as conformant to the UNIX 98 Product Standard, organized by vendor. Select a vendor's registered product and click on the hyperlink.

6. The next table shows details of the product registration, and links to a range of further information. Options at this point:
    i. Click the vendor's company name to go the vendor's own web site, either for general information on the vendor, or for information on the specific registered product.
    ii. Click the Brand Certificate link to display a copy of the actual Brand Certificate (in PDF)
    iii. Click the Go to the completed Conformance Statement hyperlink. (The Conformance Statement is a document, compiled from the answers to a Conformance Statement Questionnaire (CSQ), that the vendor supplies as part of the registration process, giving full details of the conformance of the product to the relevant Product Standard.)
        - The resultant table offers three levels of detail, available from the three icons in the left-hand column. (If required, click on the Help button for an explanation of the icons, in addition to other help information; then return to this table.) Click one of the icons.
        - The next table shows the Conformance Statement for the overall Product Standard, most of which is effectively a compilation of the Conformance Statements for each of the individual Open Group standards that the Product Standard comprises. Select a particular question, and click on an icon.
        - The final table shows the full details of the Conformance Statement for that Open Group standard.
    iv. Click the manual search of the CSQ System hyperlink.
        - The resultant search form provides access to the complete Conformance Statement Library.
        - Clicking the Search the completed Conformance Statements hyperlink enables you to search the complete library by individual Product Standard and/or by vendor, and to select how you want the results organized, and the level of detail displayed. Options now:
            - Select a particular Product Standard, and leave the vendor column as Any, to show all the vendor products registered as conforming to that Product Standard.
            - Select a particular vendor, and leave the Product Standard column as Any, to show all the Product Standards for which that vendor has registered a conforming product.
            - Select a particular Product Standard, and a particular vendor, to show whether that vendor has a product registered as conforming to that Product Standard.
            - Click the select here for an extended selection hyperlink for an even more detailed search form, allowing selection of individual vendor products as well as Product Standards and vendors.
        - Clicking the View the Conformance Statements Questionnaires hyperlink provides access to a complete list of (blank) Conformance Statements Questionnaires, so you can see the questions that vendors have to answer as part of the product registration process.
    v. Click the More information about UNIX 98 hyperlink to display details of the UNIX 98 Product Standard, including:
        - the full text of the Product Standard definition,
        - a list of all the component standards,
        - links to each of the individual specifications,
        - links to the corresponding Conformance Statement Questionnaires
        - links to information on the test suites used as the indicator of compliance

As you can see, there is a wealth of information underpinning the entries for Open Group developed standards, particularly the Product Standards.

For this reason, Open Group Product Standards should be the first point of departure when considering open industry standards for architecture specifications and procurements.

Where Open Group Product Standards do not exist, individual Open Group standards will often be the next best thing.

Summary of Open Group Databases and Resources

A summary of URLs to key Open Group resources relevant to the architect is given below.

http://www.opengroup.org/sib/	*Starting Point: the Standards Information Base home page.*
http://www.opengroup.org/sib.htm	*Direct link to the full listing of the Standards Information Base.*
http://www.opengroup.org/sib2/search_sib.tpl	*Direct link to the Standards Information Base search facility.*
http://www.opengroup.org/public/arch	*The TOGAF document (included here for completeness).*
http://www.opengroup.org/togaf	*The TOGAF public information home page.*
http://www.opengroup.org/registration	*The Open Brand and Procurement.*
http://www.opengroup.org/regproducts	*Directory of Registered Products.*

http://www.opengroup.org/prodstandards	*Current set of Product Standards.*
http://www.opengroup.org/publications	*Catalog of Specifications and Other Publications.*
http://www.opengroup.org/testing	*Testing Technology information.*
http://www.opengroup.org/testing/checklist	*Product Registration Checklists.*
http://www.opengroup.org/csqs	*Conformance Statements and Conformance Statement Questionnaires (CSQs).*
http://www.opengroup.org/corrigenda	*Publications Corrigenda.*
http://www.opengroup.org/interpretations	*Interpretations/Temporary Waiver process.*
http://www.opengroup.org/interpretations/database	*Interpretations Database.*

Integrated Information Infrastructure Reference Model (III-RM) - Basic Concepts

Background

With the emergence of internet based technologies in recent years, for many organizations the main focus of attention, and the main return on investment in architecture effort, has shifted from the Application Platform space to the Applications space. (Indeed, this has been one of drivers behind the migration of TOGAF itself from a framework and method for Technology Architecture to one for overall Enterprise Architecture.)

The TOGAF Technical Reference Model (TRM) described in the previous section focuses on the Application Platform space, and it is what the Enterprise Continuum terms a "Foundation Architecture".

This section describes another reference model, one that focuses on the Applications space, and one that is a "Common Systems Architecture" in Enterprise Continuum terms. This is the **Integrated Information Infrastructure Reference Model** (III-RM).

The Integrated Information Infrastructure Reference Model is a subset of the TOGAF Technical Reference Model in terms of its overall scope, but it also expands certain parts of the Technical Reference Model - in particular, the Business Applications and Infrastructure Applications parts - in order to provide help in addressing one of the key challenges facing the enterprise architect today: the need to design an integrated information infrastructure to enable "boundaryless information flow". These concepts are explained in detail below.

This introductory subsection examines the concept of "boundaryless information flow"; why an integrated information infrastructure is necessary to enable it; and how the Integrated Information Infrastructure Reference Model can help the architect in designing an integrated information infrastructure for his or her enterprise.

Components of the Model

Like the TOGAF Technical Reference Model, the Integrated Information Infrastructure Reference Model has two main components:

- a *taxonomy*, which defines terminology, and provides a coherent description of the components and conceptual structure of an integrated information infrastructure,
- an associated *Integrated Information Infrastructure Reference Model graphic*, which provides a visual representation of the taxonomy, and the inter-relationship of the components, as an aid to understanding.

The model assumes the underlying existence of a computing and network platform, as described in the TRM; these are not depicted in the model.

Relationship to Other parts of TOGAF

The relationship of the Integrated Information Infrastructure Reference Model to the Technical Reference Model is explained above.

Although the Integrated Information Infrastructure Reference Model is intended as a useful tool in the execution of the TOGAF

Architecture Development Method, it is important to emphasise that the ADM is in no way dependent on use of the Integrated Information Infrastructure Reference Model (any more than it is dependent on use of the TRM). Other taxonomies and reference models exist in this space that can be used in conjunction with the ADM, and indeed may be preferable for some organizations.

Key Business and Technical Drivers

Problem Space: The Need for "Boundaryless Information Flow"

The Boundaryless Information Flow problem space is one that is shared by many customer members of The Open Group, and by many similar organizations worldwide. It is essentially the problem of getting information to the right people at the right time in a secure, reliable manner, in order to support the operations that are core to the extended enterprise.

In General Electric, Jack Welch invented the term "the Boundaryless Organization", not to imply that there are no boundaries, but that they should be made permeable.

Creating organizational structures that enabled each individual department to operate at maximum efficiency was for a long time accepted as the best approach to managing a large enterprise. Among other benefits, this approach fostered the development of specialist skills in staff, who could apply those skills to specific aspects of an overall activity (such as a manufacturing process), in order to accomplish the tasks involved better, faster, and cheaper.

As each overall activity progressed through the organization, passing from department to department (for example, from Design to Production to Sales), each department would take inputs from the previous department in the process, apply its own business processes to the activity, and send its output to the next department in line.

In today's world where speed, flexibility and responsiveness to changing markets make all the difference between success and failure, this method of working is no longer appropriate. Organizations have been trying for some time to overcome the limitations imposed by traditional organization structures. Many business process re-engineering efforts have been undertaken and abandoned because they were too ambitious, while others cost far more in both time and money than originally intended.

However, organizations today recognize that they need not abandon functional or departmental organization altogether. They can enable the right people to come together in cross-functional teams so that all the skills, knowledge and expertise can be brought to bear on any specific problem or business opportunity.

But this in turn poses it own challenges. CIOs are under enormous pressure to provide access to information to each cross-functional team on an as-required basis, and yet the sources of this data can be numerous and the volumes huge.

Even worse, the IT systems, which have been built over a period of 20 or 30 years at a cost of many billions of dollars, and are not about to be thrown out or replaced wholesale, were built for each functional department. So although it may be possible to get people to work together effectively (no minor achievement in itself), the IT systems they use are designed to support the old-style thinking. The IT systems in place today do not allow for information to flow in support of the boundaryless organization. When they do, then we will have Boundaryless Information Flow.

Solution Space: The Need for "Integrated Information Infrastructure"

The Open Group's Interoperable Enterprise Business Scenario, originally published in 2001, crystallizes this need for Boundaryless Information Flow and describes the way in which this need drives IT customers' deployment of their information infrastructure.

In this scenario, the customer's problem statement says that I (as the customer enterprise) could gain significant operational efficiencies and improve the many different business processes of the enterprise - both nternal processes, and those spanning the key interactions with suppliers, customers, and partners - if only I could provide my staff with:

- **integrated information** - so that different and potentially conflicting pieces of information are not distributed throughout different systems; and
- **integrated access to that information** - so staff can access all the information they need and have a right to, through one convenient interface.

The infrastructure that enables this vision is termed the **Integrated Information Infrastructure**.

As an example, one current approach to Integrated Information Infrastructure is to provide "enterprise portals" that allow integrated access to information from different applications systems enterprise-wide, via a convenient, web-enabled interface (one of the colored segments in the ends of the cylinder in the figure below):

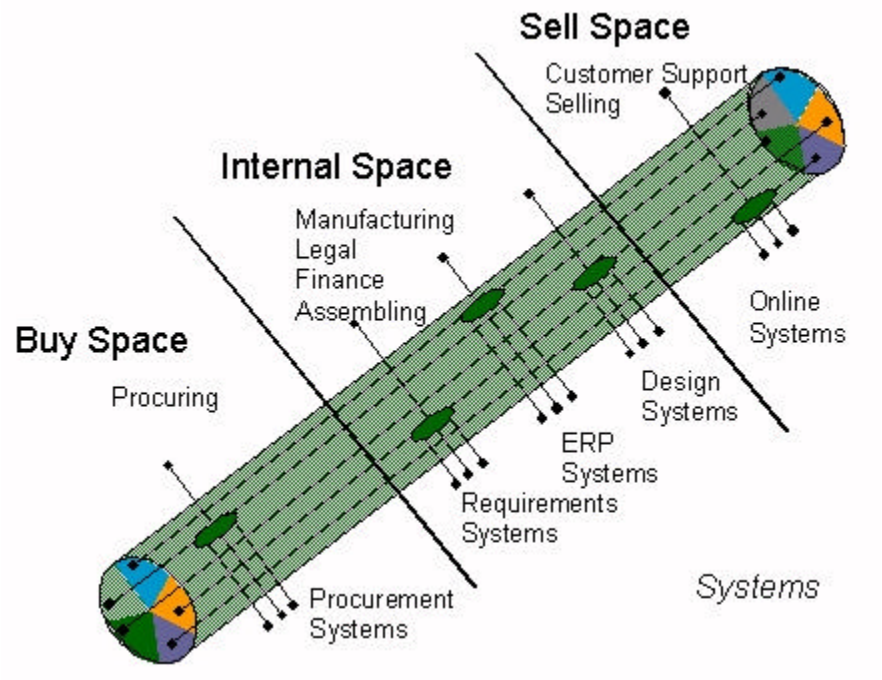


Figure 1: An approach to Boundaryless Information Flow ("Enterprise Portals")

One of the key challenges for the architect in today's enterprise is to work out, and then communicate to senior management, how far technologies such as web services, application integration services, etc., can go toward achieving an integrated information infrastructure, and realizing the vision of Boundaryless Information Flow, in the enterprise concerned.

The Open Group's follow-up analysis of the Interoperable Enterprise Business Scenario has resulted in the development of an Integrated Information Infrastructure model, which depicts the major components required to address the Boundaryless Information Flow problem space, and can help the architect in this task.

The Integrated Information Infrastructure Reference Model thus provides insights related to customer needs for Boundaryless Information Flow in enterprise environments. The model also points to rules and standards to assist in leveraging solutions and products within the value chain.

The following subsections discuss the model in detail.

"Health Warning"

The Integrated Information Infrastructure Reference Model is documented as it stands today, and is by no means considered a finished article. However, it is a model that has been developed and approved by the members of The Open Group as a whole, in response to an Interoperable Enterprise Business Scenario, which itself was developed in response to an urgent need articulated by the customer members of The Open Group for assistance in this field.

The business scenario and the reference model thus represent a problem and a solution approach that The Open Group membership as a whole fully endorses.

It is hoped that publication of the model as part of TOGAF will encourage its widespread adoption and use, and provide a channel of communication whereby experience with use of the model can be fed back, improvement points assimilated, and the model refined and republished as necessary.

Integrated Information Infrastructure Reference Model (III-RM) - High-Level View

Derivation High-Level Graphic Components

Derivation of the III-RM from the TRM

The Integrated Information Infrastructure Reference Model is a model of the major component categories for developing, managing, and operating an integrated information infrastructure. It is a model of a set of applications that sits on top of an application platform. This model is a subset of the TOGAF TRM, and it uses a slightly different orientation.

Consider Figure 2a below where two views of the TOGAF TRM are presented. The left side is the familiar view of the TOGAF TRM; it is a side view, where we look at the model as if looking at a house from the side, revealing the contents of the "floors". The top down view on the right hand side depicts what one might see if looking at a house from the "roof" down.

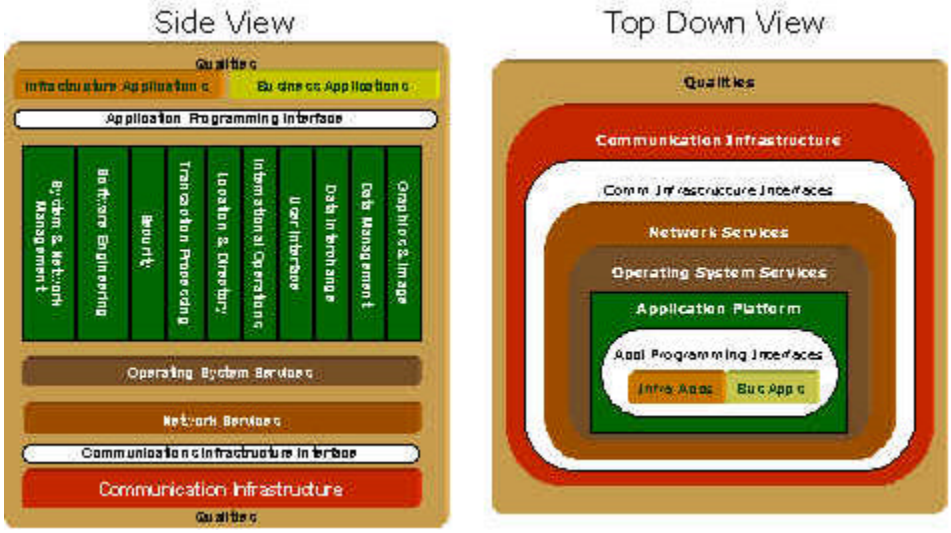


Figure 2a: TOGAF TRM Orientation Views

The subset of the TRM that comprises the Integrated Information Infrastructure Reference Model is depicted in Figure 2 below, in which those parts of the TRM not relevant to the Integrated Information Infrastructure Reference Model are "greyed out".

Figure 2b illustrates that the focus is on the application, application platform, and qualities subset of the TOGAF TRM.

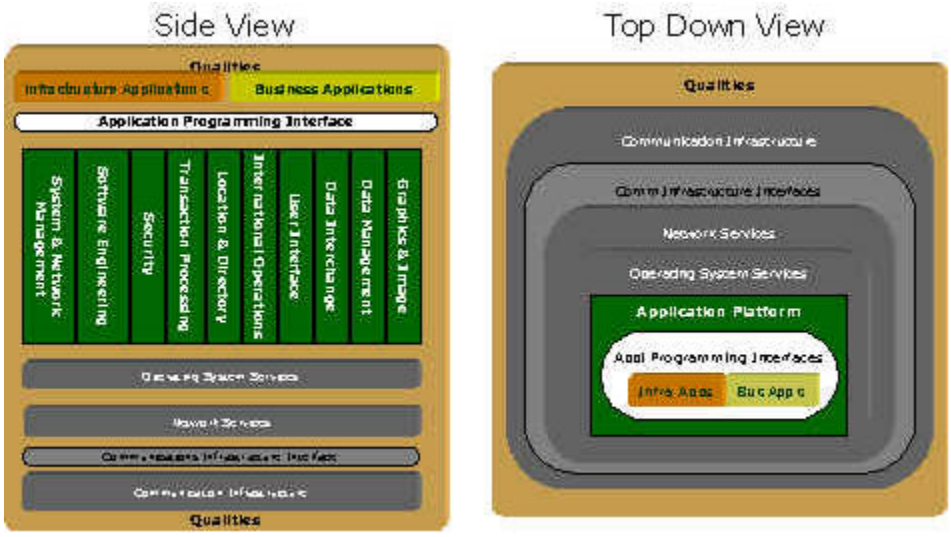


Figure 2b: Focus of the Integrated Information Infrastructure Model

The High-Level III-RM Graphic

The resulting Integrated Information Infrastructure Reference Model itself is depicted in Figure 3. It is fundamentally an Application Architecture reference model - a model of the application components and application services software essential for an integrated information infrastructure. (There are more Business Applications and Infrastructure Applications than these in the environment, of course, but these are the subsets relevant to the Boundaryless Information Flow problem space.)

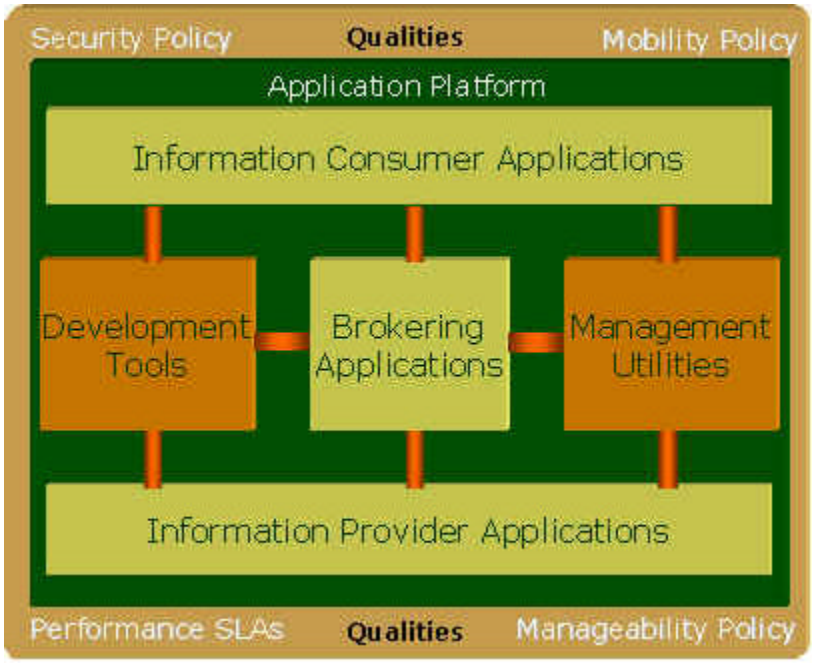


Figure 3 - Integrated Information Infrastructure Reference Model - High-Level

As explained above, the model assumes the underlying existence of a computing and network platform, and does not depict them explicitly.

Although the computing and network platform are not depicted, there may be requirements on them that must be met, in addition to requirements on the components of the Integrated Information Infrastructure Reference Model, in order to fully address the Boundaryless Information Flow problem space.

Components of the High-Level III-RM

The Integrated Information Infrastructure Reference Model has the following core components:

- **Business Applications**, denoted by the light-brown boxes in the high-level model (corresponding to the light-brown "Business Applications" box in the TRM graphic). There are three types of Business Application in the model:
  - **Information consumer applications**, which deliver content to the user of the system, and provide services to request access to information in the system on the user's behalf
  - **Brokering applications**, which manage the requests from any number of clients to and across any number of Information provider applications
  - **Information provider applications**, which provide responses to client requests and rudimentary access to data managed by a particular server
- **Infrastructure Applications**, denoted by the dark-brown boxes in the high-level model (corresponding to the dark-brown "Infrastructure Applications" box in the TRM graphic). There are two types of Infrastructure Application in the model:
  - **Development tools**, which provide all the necessary modeling, design, and construction capabilities to develop and deploy applications that require access to the integrated information infrastructure, in a manner consistent with the standards of the environment
  - **Management utilities**, which provide all the necessary utilities to understand, operate, tune, and manage the run-time system in order to meet the demands of an ever changing business, in a manner consistent with the standards of the environment
- An **Application Platform**, which provides supporting services to all the above applications, in areas such as location, directory, work flow, data management, data interchange, etc., and thereby provides the ability to locate, access, and move information within the environment. This set of services constitutes a subset of the total set of services of the TRM Application Platform, and is denoted by the dark green underlay in the high-level model (corresponding to the dark green of the application platform in the TRM graphic).
- The **Interfaces** used between the components. Interfaces include formats and protocols, application programming interfaces, switches, data values, etc. Interfaces among components at the application level are colored brown. Interfaces between any application-level components and their supporting services in the applicaiton platform are colored white (corresponding to the white of the Application Programming Interface box in the TRM graphic).
- The **Qualities** backplane, denoted by the beige underlay in the high-level model (corresponding to the beige of the Qualities backplane in the TRM graphic). The applications and application platform must adhere to the policies and requirements depicted by the qualities backplane.

Integrated Information Infrastructure Reference Model (III-RM) - Detailed Taxonomy

Detailed Graphic Business Applications Infrastructure Applications Application Platform Qualities

The Detailed III-RM Graphic

The detailed Integrated Information Infrastructure Reference Model is depicted in Figure 4.

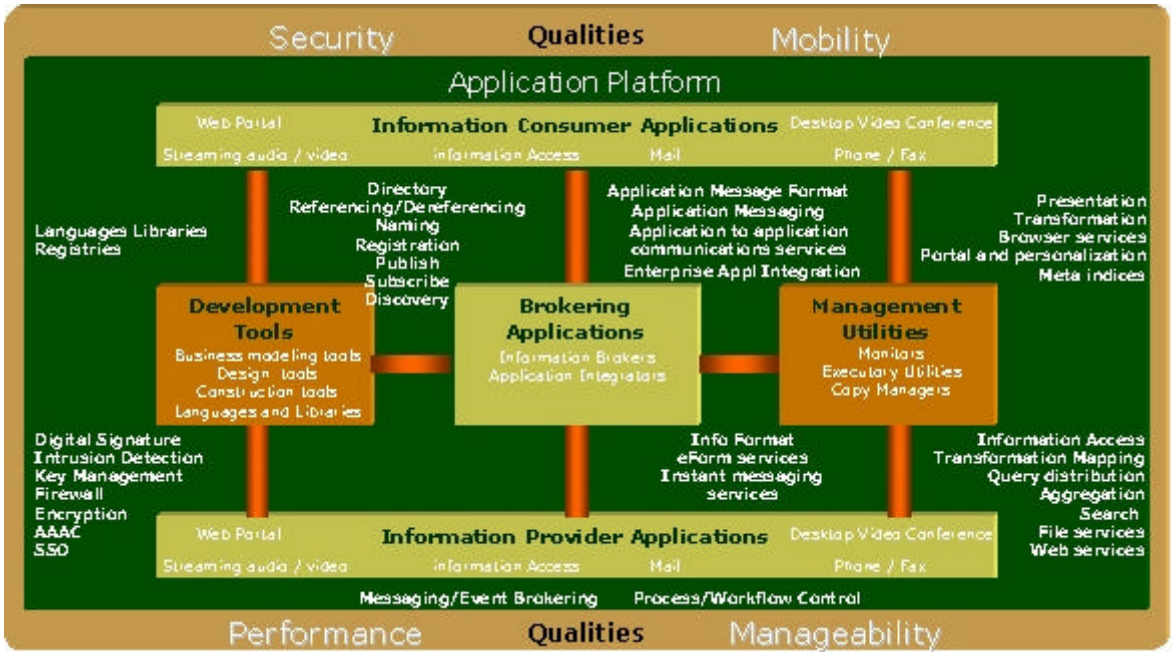


Figure 4 - Integrated Information Infrastructure Reference Model - Detailed

The remaining subsections expand on the taxonomy / component detail shown in Figure 4.

Business Applications

There are three types of Business Application in the model:

- **Information provider applications**, which provide responses to client requests and rudimentary access to data managed by a particular server
- **Brokering applications**, which manage the requests from any number of clients to and across any number of service providers
- **Information consumer applications**, which deliver content to the user of the system, and provide services to request access to information in the system on the user's behalf

The overall set of Information Provider, Information Consumer, and Brokerage applications collectively creates an environment that provides a rich set of end-user services for transparently accessing heterogeneous systems, databases, and file systems.

Information Provider Applications

To the extent that information today can be regarded as being "held hostage", as depicted in Figure 5, Information Provider applications are those applications that "liberate" data from their silos.

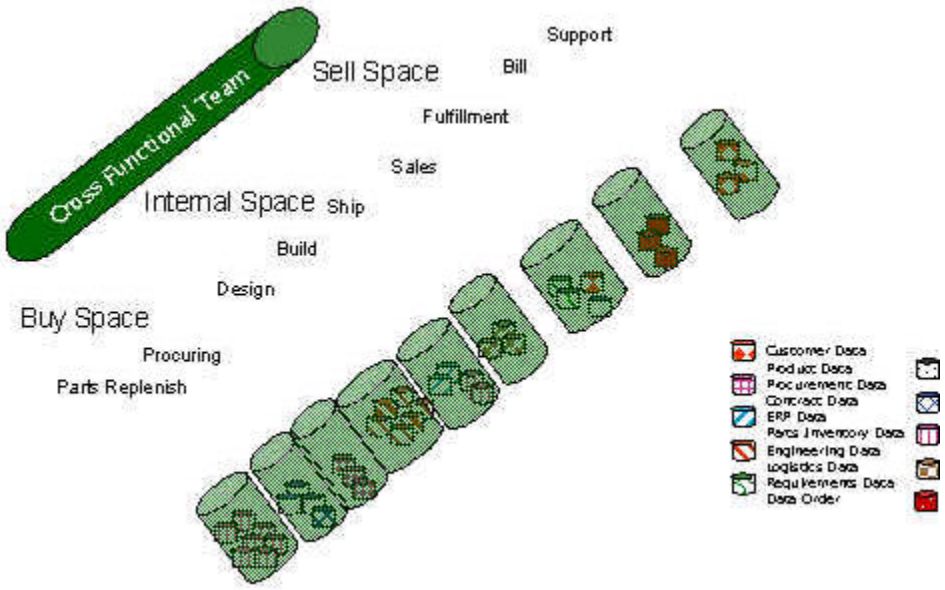


Figure 5: The Need to "Liberate" Data from their Silos to Meet the Information Needs of Cross-Functional Enterprise Teams

Information Provider applications achieve this by providing an open interface to a potentially proprietary silo interface, as illustrated in Figure 6, where the interfaces on the left of the Information Provider applications are open interfaces and the interfaces between the Information Provider applications and silo data are proprietary interfaces.

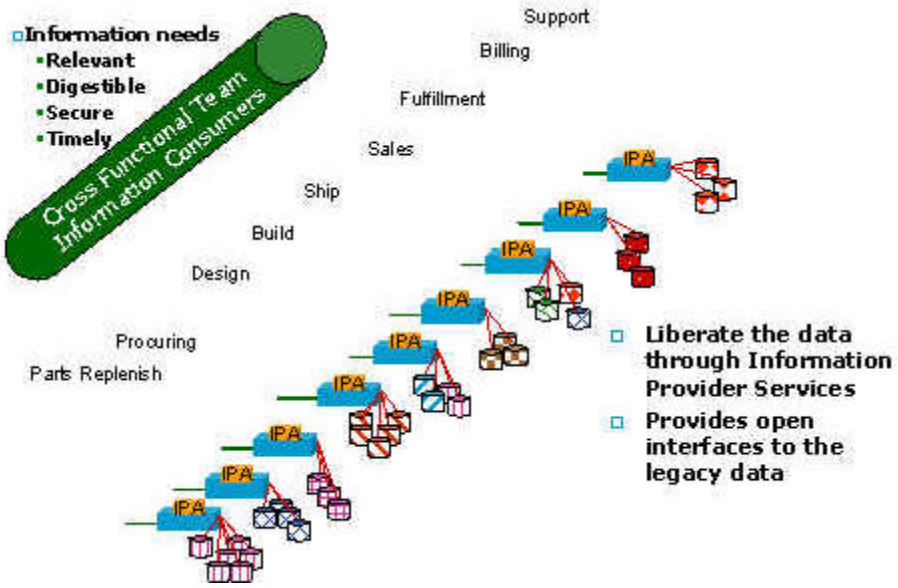


Figure 6: Information Provider Applications"Liberate" Data by Providing Open Interfaces to the Data Silos

Brokerage Applications

Brokerage applications serve up single requests that require access to multiple information sources. A brokerage application break downs such a request, distributes the request to multiple information sources, collects the responses, and sends a single response back to the requesting client.

Brokerage applications access Information Provider applications using the open interfaces provide by the Information Provider applications (as described above); they integrate information from multiple Information Provider applications and pass the integrated information to information consumer applciations using open interfaces.

Brokerage applications also enable access to information within the enterprise by strategic partners.

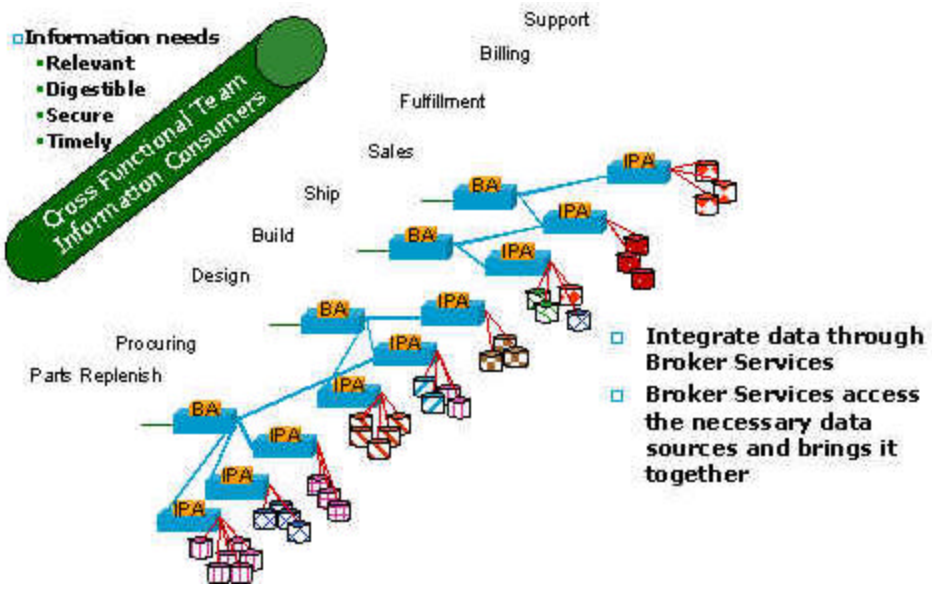


Figure 7: Brokerage Applications Integrate the Information from Multiple Information Provider Applications

Information Consumer Applications

Information Consumer Applications provide information to end-users in the form in which they need it, when they need it, and in a secured manner. This includes providing the information in text, video, audio, English, German, … etc.

Information Consumer Applications communicate with Brokerage applications or Information Provider applications using the open interfaces that the Brokerage and Information Provider applications provide. Security is provided through the firewalls and / or security services.

Figure 8 below depicts the Information Consumer Applications with the security services depicted as the brick pattern.

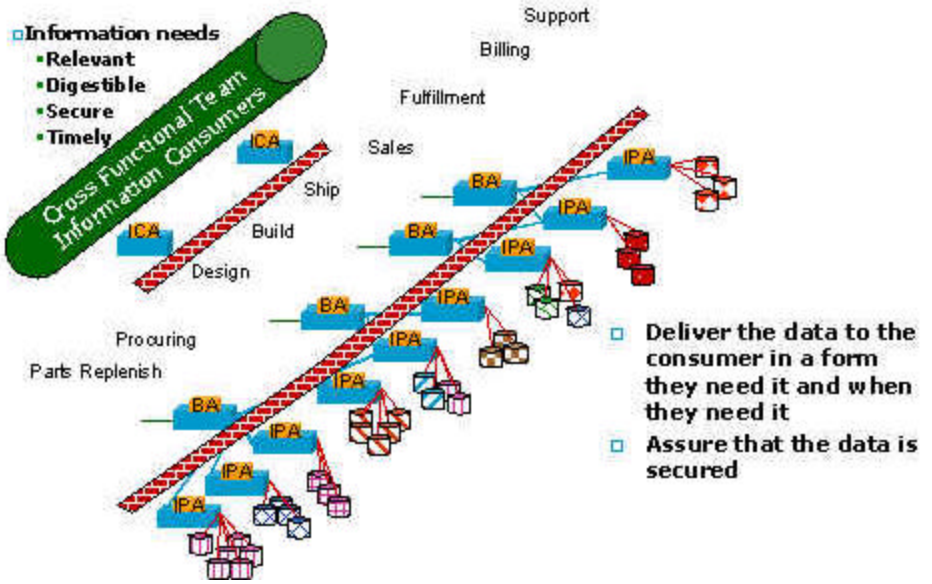


Figure 8: Information Consumer Applications communicate with Brokerage or Information Provider applications using the open interfaces that they provide

Infrastructure Applications

There are two types of Infrastructure Application in the model:

- **Development tools**, which provide all the necessary modeling, design, and construction capabilities to develop and deploy applications that require access to the integrated information infrastructure, in a manner consistent with the standards of the environment
- **Management utilities**, which provide all the necessary utilities to understand, operate, tune, and manage the run-time system in order to meet the demands of an ever changing business, in a manner consistent with the standards of the environment

Development Tools

The Development Tools component of the model comprises applications that take the form of tools for modeling, designing and constructing the integrated information infrastructure. Specifically, it includes tools for business, process, and data modeling, as well as the traditional application construction tools that transform the business model into software that automates the business processes revolving around information.

Note that each set of tools will be logically connected through a Directory, allowing one tool to be driven by data from another. The following sections describe the requirements for components of Development Tools. The tool set also inlcudes a repository.

Business Modeling Tools

This category covers tools for the modeling of business rules and business process rules.

Business modeling describes and documents the business in a comprehensive knowledge base. It establishes a consensus among general management of the business direction, organization, processes, information requirements, and the current environment of the business. Perhaps most importantly, this understanding is documented in a common, business-oriented format to be utilized for subsequent enhancement.

Design Modeling Tools

This category covers tools for designing, defining, and documenting the most pertinent information technology elements of the business based upon the business and business process rules. Examples of elements to be designed include: connections between people, organizations, work flows and computers; data and object models; physical data translation and translation rules; and constraints.

Implementation and Construction Tools

Implementation tools enable timely development of reusable processes, applications and application services. Such tools include intelligent browsers, data manipulation language compilers and optimizers, distributed application compilers and debuggers, heterogeneous client and server development tools, policy definition tools, and work flow script generation tools.

Data Modeling Tools

[add stuff here]

Deployment Tools

Deployment tools are necessary to move implemented software from the development environment into the operational environment.

Libraries

The Development Tools component includes re-usable libraries of software that use the standards of the operational environment.

Management Utilities

This category covers applications that take the form of utilities for operations, administration, and systems management, and for the management of data based on availability and cost requirements. Such utilities may execute in an attended or an unattended environment.

Operations, Administration and Management (OA&M) Utilities

The OA&M component covers traditional systems management and administration utilities that manage business rules and information objects. Examples include: utilities for installation, copyright and license management; and miscellaneous administration, configuration and registration functions. Additionally there are utilities the control of service billing, service triggering and account management.

Quality of Service Manager Utilities

These include Health Monitoring and Management Utilities.

Copy Management Utilities

Copy Management utilities are those that manage data movement from any given operational system to necessary distribution points in the enterprise, in order to ensure the maximum leverage of operational systems data. They also include tools that detect and flag poor quality data.

Storage Management Utilities

These are utilities that provide least-cost data storage management. Storage management utilities support the wide variety of storage mechanisms and are connected to file, object and database systems.

Application Platform

All the different types of application described above are built on top of the services provided by the Application Platform.

The Application Platform component of the Integrated Information Infrastructure Reference Model comprises a subset of all the services defined in the TOGAF Techical Reference Model, the subset that pertains to Integrated Information Infrastructure. Specifically, it comprises all those services in the TRM application platform that allow applications to focus on understanding and processing the information required, rather than understanding the form, format, and/or location of the information.

The services of the Application Platform component can be used to support conventional applications as well as Brokerage, Information Consumer, and Information Provider applications. When used as part of an overall Applications Architecture in this way, such an approach enables maximum leverage of a single operational environment that is designed to ensure effective and consistent transfer of data between processes, and to support fast and efficient development, deployment and management of applications.

The Application Platform component comprises the following categories of service.

Software Engineering Services

- Languages
- Libraries and
- Registries

Security Services

- Authentication, Authorization, and Access Control
- Single Sign-On
- Digital Signature
- Firewall
- Encryption
- Intrusion detection
- Identity Management
- Key Management

Location and Directory Services

Location and Directory services provide access facilities for name, location, description, and relationship data that describes the integrated information infrastructure.

Directory services support the deployment and enterprise-wide availability of an integrated information infrastructure directory. The data in the directory is made available to all other components in the architecture model.

Figure 9 depicts the juxtaposition of location and directory services to the other components.

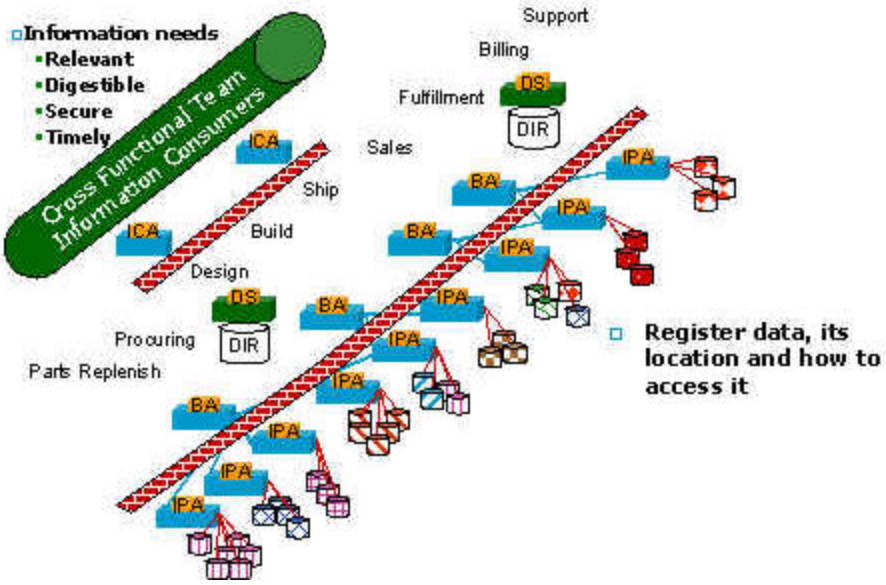


Figure 9: Juxtaposition of Location and Directory Services to Other Components

Specific services include:

- Directory
- Registration
- Publish/Subscribe
- Discovery
- Naming
- Referencing/Dereferencing

Human Interaction Services

Human Interaction Services provide the means to consistently present data to the end user in the appropriate format. They comprise services that assist in the formulation of customer data requests and enable visualization and presentation of the data accessed.

Specific services include:

- Presentation
- Transformation
- Browser services
- Meta indices
- Portal and personalization services

Data Interchange Services

Specific services include:

- Information Format
- eForm services
- Instant messaging services
- Application messaging

- Application to application communications services
- Enterprise application integration

Data Management Services

Specific services include:

- Information and data access
- Transformation Mapping
- Query distribution
- Aggregation
- Search
- File services

Information access services provide the ability for an application to access an integrated view of data, regardless of whether the data exists in a mainframe system or in a distributed system. The information access services ensure that data integrity is maintained among multiple databases, and also provide on-line data cleansing (whereby data is checked against data rules for each access.)

Data Access Services provide open interfaces to legacy data, provide new applications standard database access services to vast amounts of existing data, and provide standard access services to new data types.

Additional Operating System Services

Specific services include:

- Event Brokering Services
- Workflow Services

These additional services enable the flow of information, as depicted in Figure 10.

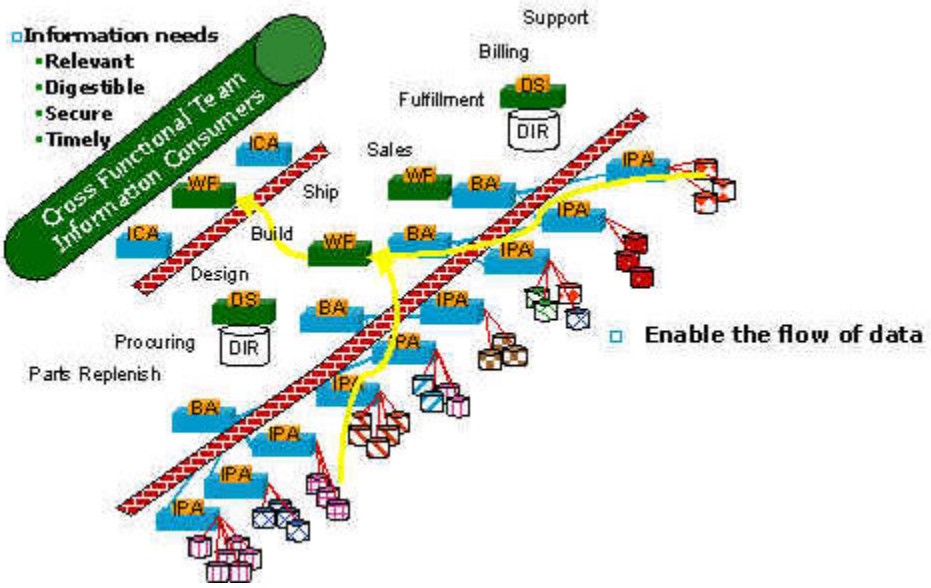


Figure 10: Workflow Services Enable Information Flow

Workflow denotes the concept of automating processes by facilitating user interactions and executing applications according to a process map. Workflow services enable integration of enterprise applications, resulting in applications of extended value.

Workflow services also address the needs of managing an environment where legacy systems are prevalent.

Workflow services also provide a means to encapsulate existing applications, thereby supporting customer needs for leverage of existing assets.

Qualities

The Qualities component of the model is supported by Quality of Service services, including the various services required to maintain the quality of the system as specified in service level agreements.

Included in this are the services to post conditions to, and react to requests from, the Quality of Service Manager.

PART IV: Resource Base

Architecture Board

Role Responsibilities Setting Up Operation

Role

A key element in a successful Architecture Governance strategy is a cross-organization Architecture Board to oversee the implementation of the strategy. This body should be representative of all the key stakeholders in the architecture, and will typically comprise a group of executives responsible for the review and maintenance of the overall architecture.

The cost of establishing and operating an Architecture Board are more than offset by the savings that accrue as a result of preventing one-off solutions and unconstrained developments across the enterprise, which invariably lead to:

- High costs of development
- High costs of operation and support:
- Numerous run-time environments
- Numerous implementation languages
- Numerous interfaces and protocols ...
- Lower quality
- Higher risk
- Difficulty in replicating and re-using solutions

Architecture Boards may have global, regional or business line scope. Particularly in larger enterprises, Architecture Boards typically comprise representatives from the organization at a minimum of two levels:

- Local (domain experts, line responsibility)
- Global (organisation-wide responsibility)

In such cases, each board will be established with identifiable and articulated:

- Responsibilities and decision-making capabilities
- Remit and authority limits

Responsibilities

The Architecture Board is typically made responsible, and accountable, for achieving some or all of the following goals:

- Consistency between sub-architectures
- Identifying reusable components
- Flexibility of enterprise architecture
  - to meet changing business needs
  - to leverage new technologies
- Enforcement of architecture compliance
- Improving the maturity level of architecture discipline within the organization
- Ensuring that the discipline of architecture-based development is adopted
- Providing the basis for all decision-making with regard to changes to the architectures
- Supporting a visible escalation capability for out-of-bounds decisions

Further responsibilities from an operational perspective should include:

- All aspects of monitoring and control of the architectures contract
- Meeting on a regular basis
- Ensuring the effective and consistent management and implementation of the architectures

- Resolving ambiguities, issues or conflicts that have been escalated
- Providing advice, guidance, and information
- Ensuring compliance with the architectures, and granting dispensations that are in keeping with the technology strategy and objectives
- Considering policy (schedule, SLA etc.) changes where similar dispensations are requested and granted: e.g., new form of service requirement
- Ensuring that all information relevant to the implementation of the architecture contract is published under controlled conditions and made available to authorised parties
- Validation of reported service levels, cost savings etc.

From a Governance perspective, the Architecture Board is also responsible for:

- The production of usable governance material and activities
- Providing a mechanism for the formal acceptance and approval of Architecture through consensus and authorised publication
- Providing a fundamental control mechanism for ensuring the effective implementation of the Architecture
- Establishing and maintaining the link between the implementation of the Architecture, the architectural strategy and objectives embodied in the Enterprise Architecture, and the strategic objectives of the business
- Identifying divergence from the architecture and planning activities for realignment through dispensations or policy updates

---

Setting Up the Architecture Board

Triggers

One or more of the following occurrences typically triggers the establishment of an Architecture Board:

- New CIO
- Merger or acquisition
- Consideration of a move to newer forms of computing
- Recognition that IT is poorly aligned to business
- Desire to achieve competitive advantage via technology
- Creation of an enterprise architecture program
- Significant business change or rapid growth
- Requirement for complex, cross-functional solutions

In many companies, the Executive Sponsor of the initial architecture effort is the CIO (or other senior executive). However, to gain broad corporate support, a sponsoring body has more influence. This sponsoring body is here called an *Architecture Board*, but the title is not important. Whatever the name, it is the executive-level group responsible for the review and maintenance of the strategic architecture and all of its sub-architectures.

The Architecture Board is the sponsor of the architecture within the enterprise, but the Architecture Board itself needs an Executive Sponsor from the highest level of the corporation. This commitment must span the planning process and continue into the maintenance phase of the architecture project. In many companies that fail in an architecture planning effort, there is a notable lack of executive participation and encouragement for the project.

A frequently overlooked source of Architecture Board members is the company's Board of Directors. These individuals invariably have diverse knowledge about the business and its competition. Because they have a significant impact on the business vision and objectives, they may be successful in validating the alignment of IT strategies to business objectives.

Size of the Board

The recommended size for an Architecture Board is four or five (and no more than ten) permanent members.

In order to keep the Architecture Board to a reasonable size, while ensuring enterprise-wide representation on it over time, membership of the Architecture Board may be rotated, giving decision-making privileges and responsibilities to various senior managers. This may be required in any case, due to some Architecture Board members finding that time constraints prevent long-term active participation.

However, some continuity must exist on the Architecture Board, to prevent the corporate architecture from varying from one

set of ideas to another. One technique for ensuring rotation with continuity is to have set terms for the members, and to have the terms expire at different times.

In the ongoing architecture process following the initial architecture effort, the Architecture Board may be re-chartered. The Executive Sponsor will normally review the work of the Architecture Board and evaluate its effectiveness; if necessary, the Architecture Compliance Review Process is updated or changed.

Board Structure

The TOGAF Architecture Governance Framework provides a generic organizational framework that positions the Architecture Board in the context of the broader Governance structures of the enterprise. This structure identifies the major organizational groups and responsibilities, as well as the relationship between each group. This is a best practice structure, and may be subject to change depending on the organization's form and existing structures.

Consideration must be taken into account regarding the size of the organization, its form, and how the IT functions are implemented. This will provide the basis for designing the architecture board structure within the context of the overall governance environment. In particular, consideration should be given to the concept of global ownership and local implementation, and the integration of new concepts and technologies from all areas implementing against architectures.

The structure of the Architecture Board should reflect the form of the organization. The architecture governance structure required may well go beyond the generic structures outlined in the TOGAF Architecture Governance Framework. The organization may need to define a combination of the IT governance process in place and the existing organizational structures and capabilities, which typically include the following types of body:

- Global governance board
- Local governance board
- Design authorities
- Working parties.

Operation of the Architecture Board

This section describes the operation of the Architecture Board particularly from the Governance perspective.

General

Architecture Board meetings should be conducted within clearly identified agendas with explicit objectives, content coverage, and defined actions. In general, board meetings will be aligned with best practice, such as given in the COBIT framework.

These meetings will provide key direction in:

- Supporting the production of quality governance material and activities
- Providing a mechanism for the formal acceptance through consensus and authorised publication
- Providing a fundamental control mechanism for ensuring the effective implementation of the architectures
- Establishing and maintaining the link between the implementation of the architectures and the stated strategy and objectives of the organisation (business and IT)
- Identifying divergence from the contract and planning activities to realign with the contract through dispensations or policy updates.

Preparation

Each participant will receive an agenda and any supporting documentation, e.g. dispensation requests, performance management reports, etc., and will be expected to be familiar with the contents of each.

Where actions have been allocated to an individual, it is that person's responsibility to report on progress against these.

Each participant must confirm his or her availability and attendance at the governance board meeting.

Agenda

This section outlines the contents of a governance board meeting agenda. Each agenda item is described in terms of its content only.

Minutes of Previous Meeting

Minutes contain the details of previous governance board meeting as per standard organisational protocol.

Requests for Change

Items under this heading are normally change requests for amendments to architectures, principles, etc., but may also include business control with regard to architecture contracts: e.g., ensure that voice traffic to premium numbers, such as Weather Reports, are barred and data traffic to certain web sites is controlled.

Any request for change is made within agreed authority levels and parameters defined by the architecture contract.

Dispensations

The dispensation is the mechanism used to request a change to the existing architectures, contracts, principles etc. outside of normal operating parameters, e.g. exclude provision of service to a subsidiary, request for unusual service levels for specific business reasons, deploy non-standard technology or products to support specific business initiatives.

Dispensations are granted for a given time period and set of identified service and operational criteria that must be enforced during the life span of the dispensation. Dispensations are not granted indefinitely, but are used as a mechanism to ensure that service levels and operational levels etc. are met while providing a level flexibility in their implementation and timing. The time-bound nature of dispensations ensures that they are a trigger to the compliance activity.

Compliance Assessments

Compliance is assessed against SLAs, OLAs, cost targets and required architecture refreshes. These assessments will be reviewed and either accepted or rejected depending on the criteria defined within the governance framework. The compliance assessment report will include details as described.

Dispute Resolution

Disputes that have not been resolved through the compliance and dispensation processes are identified here for further action and are documented through the compliance assessments and dispensation documentation.

Architecture Strategy and Direction Documentation

This describes the architecture strategies, direction and priorities and will only be formulated by the global governance board. It should take the form of standard architecture documentation.

Actions Assigned

This is a report on the actions assigned at previous governance board meetings. An action tracker is used to document and keep the status of all actions assigned during the governance board meetings and should consist of at least the following information:

- Reference
- Priority
- Action description
- Action owner
- Action details
- Date raised
- Due date
- Status
- Type
- Resolution date

Contract Documentation Management

This is a formal acceptance of updates and changes to architecture documentation for onward publication as versioned [Adobe Acrobat™ PDF format] files.

AOB

Description of issues not directly covered under any of the above. These may not be described in the agenda but should be raised at the beginning of the meeting. Any supporting documentation must be managed as per all architecture governance documentation.

Schedule of meetings

All meeting dates detail should be detailed and published.

Architecture Review Checklist - Hardware and Operating System

1. What is the project's life cycle approach?
2. At what stage is the project in its life cycle?
3. What key issues have been identified or analyzed that the project believes will drive evaluations of hardware and operating systems for networks, servers and end user devices?
4. What system capabilities will involve high-volume and/or high frequency data transfers?
5. How does the system design impact or involve end user devices?
6. What is the quantity and distribution (regional and global) of usage, data storage and processing?
7. What applications are affinitized with your project by similarities in data, application services, etc.?
8. To what degree does data?
9. What hardware and operating system choices have been made before functional design of key elements of the system?
10. If hardware and operating system decisions were made outside of the project's control:
    - What awareness does the project have of the rationale for those decisions?
    - How can the project influence those decisions as system design takes shape?
11. If some non-standards have been chosen:
    - What are the essential business and technical requirements for not using corporate standards?
    - Is this supported by a business case?
    - Have the assumptions in the business case been subject to scrutiny?
12. What is your process for evaluating full life-cycle costs of hardware & operating systems?
13. How has corporate financial management been engaged in evaluation of life-cycle costs?
14. Have you performed a financial analysis of the supplier?
15. Have you made commitments to any supplier?
16. Do you believe your requirements can be met by only one supplier?

Architecture Review Checklist - Software Services and Middleware

1. Describe how error conditions are defined, raised and propagated between application components.
2. Describe the general pattern of how methods are defined and arranged in various application modules.
3. Describe the general pattern for how method parameters are defined and organized in various application modules. (Are [in], [in/out], [out] parameters always specified in the same order?) Do Boolean values returned by modules have a consist outcome?
4. Describe the approach that is used to minimize the number of round trips between client and server calls particularly for out of process calls, and when complex data structures are involved.
5. Describe the major data structures that are passed between major system components.
6. Describe the major communication protocols that are used between major system components.
7. Describe the marshaling techniques that are used between various system components. Describe any specialized marshaling arrangements that are used.
8. Describe to what extent the system is designed with stateful and stateless components.
9. Describe how and when state is saved for both stateful and stateless components.
10. Describe the extent to which objects are created, used, and destroyed versus reused through object pooling.
11. Describe the extent to which the system relies on threading or critical section coding.
12. Describe the approach and the internal documentation that is used internally in the system to document the methods, methods arguments and method functionality.
13. Describe the code review process that was used to build the system.
14. Describe the unit testing that has been used to test the system components
15. Describe the pre and post condition testing that is included in various system modules.
16. Describe the assertion testing that is included with the system.
17. Do components support all the interface types they need to support or are certain assumptions made about what types of components will call other components either in terms of language bindings or other forms of marshaling.
18. Describe the extent to which big endian or little endian data format problems need to be handled across different platforms.
19. Describe if numbers or strings need to be handled differently across different platforms.
20. Describe whether the software needs to check for floating point round off errors.
21. Describe how time and data functions are Year 2000 compliant.
22. Describe what tools or process have been used to test the system for memory leaks, reachability or general robustness
23. Describe the layering of the systems services software. Describe the general number of links between major system components. Is the system composed of a lot of point to point interfaces or are major messaging backbones used instead?
24. Describe to what extent the system components are either loosely coupled or tightly coupled.
25. What requirements does the system need from the infrastructure in terms of shared libraries, support for communication protocols, load balancing, transaction processing, system monitoring, naming services or other infrastructure services?
26. Describe how the system and system components are designed for refactoring.
27. Describe how the system or system components rely on common messaging infrastructure versus a unique point to point communication structure.

Architecture Review Checklist - Applications

Infrastructure Applications Business Applications Application Integration Approach

Recommended Questions for Infrastructure ("Enterprise Productivity") Applications

1. Is there need for capabilities that are not provided through the enterprise's standard infrastructure application products? For example:

- Collaboration
  - Application sharing
  - Video conferencing
  - Calendaring
  - e-mail
- Workflow management
- Publishing / Word processing Applications
  - HTML
  - SGML & XML
  - Portable document format
  - Document processing - Proprietary format
  - Desktop publishing
- Spreadsheet Applications
- Presentation Applications
  - Business presentations
  - Image
  - Animation
  - Video
  - Sound
  - CBT
  - Web Browsers
- Data management applications
  - Database interface
  - Document Management
  - Product Data Management
  - Data Warehouses/Mart
- Program management Applications
  - Project Management
  - Program Visibility

2. Describe the business requirements for enterprise infrastructure application capabilities that are not met by the standard products.

Recommended Questions for Business Applications

1. Are any of the capabilities required provided by standard products supporting one or more Line-of-Business applications? For example:

- Business Acquisition applications
  - Sales & Marketing
- Engineering applications
  - Computer-aided design
  - Computer aided engineering
  - Mathematical & Statistics Analysis
- Supplier Management applications
  - Supply Chain Management
  - Customer Relationship Management
- Manufacturing applications
  - Enterprise Resource Planning Applications
  - Manufacturing Execution Systems

    - Manufacturing Quality
    - Manufacturing Process Engineering
    - Machine & Adaptive Control
- Customer Support applications
    - Airline Logistics Support
    - Maintenance Engineering
- Finance applications
- People applications
- Facilities applications
- Information Systems applications
    - Systems Engineering
    - Software Engineering
    - Web developer tools
    - Integrated Development Environments
    - Life Cycle Categories
    - Functional Categories
    - Specialty Categories
- Computer-aided manufacturing
- E-business enablement
- Business Process Engineering
    - Statistical Quality Control

2. Describe the process requirements for business application capabilities that are not met by the standard products.

Recommended questions for Application Integration Approach

1. What integration points (business process/activity, application, data, computing environment) are targeted by this architecture?
2. What application integration techniques will be applied (common business objects [ORBs], standard data definitions [STEP, XML, etc], common user interface presentation/desktop)?

Architecture Review Checklist - Information Management

Data Values Data Definition Security/Protection Hosting, Data Types, and Sharing Common Services Access Method

Data Values

1. What are the processes that standardize the management and use of the data?
2. What business process supports the entry and validation of the data? Use of the data?
3. What business actions correspond to the creation and modification of the data?
4. What business actions correspond to the deletion of the data and is it considered part of a business record?
5. What are the data quality requirements required by the business user?
6. What processes are in place to support data referential integrity and / or normalization?

Data Definition

1. What are the data model, data definitions, structure, and hosting options of purchased applications (COTS)?
2. What are the rules for defining and maintaining the data requirements and designs for all components of the information system?
3. What shareable repository is used to capture the model content and the supporting information for data?
4. What is the physical data model definition (derived from logical data models) used to design the database?
5. What software development and data management tools been selected?
6. What data owners have been identified to be responsible for common data definitions, eliminating unplanned redundancy, providing consistently reliable, timely, and accurate information, and protecting data from misuse and destruction?

Security/Protection

1. What are the data entity and attribute access rules, which protect the data from unintentional and unauthorized alterations, disclosure, and distribution?
2. What are the data protection mechanisms to protect data from unauthorized external access?
3. What are the data protection mechanisms to control access to data from external sources that temporarily have internal residence within Boeing?

Hosting, Data Types, and Sharing

1. What is the discipline for managing sole-authority data as one logical source with defined updating rules for physical data residing on different platforms?
2. What is the discipline for managing replicated data, which is derived from operational sole-authority data?
3. What tier data server has been identified for the storage of high- or medium-critical operational data?
4. What tier data server has been identified for the storage of type C operational data?
5. What tier data server has been identified for the storage of decision support data contained in a data warehouse?
6. What database management systems have been implemented?

Common Services

1. What are the standardized distributed data management services (e.g., validation, consistency checks, data edits, encryption, and transaction management) and where do they reside?

Access Method

1. What are the data access requirements for standard file, message, and data management?
2. What are the access requirements for decision support data?
3. What are the data storage and the application logic locations?
4. What query language is being used?

Architecture Review Checklist - Security

1. **Security Awareness**: Have you ensured that the corporate security policies and guidelines to which you are designing are the latest versions? Have you read them? Are you aware of all relevant Computing Security Compliance and Risk Acceptance processes? (Interviewer should list all relevant policies and guidelines.)

2. **Identification / Authentication**: Diagram the process flow of how a user is identified to the application and how the application authenticates that the user is who they claim to be. Provide supporting documentation to the diagram explaining the flow from the user interface to the application/database server(s) and back to the user. Are you compliant to corporate policies on Accounts, Passwords, etc?

3a. **Authorization**: Provide a process flow from beginning to end showing how a user requests access to the application, indicating the associated security controls and separation of duties. This should include how the request is approved by the appropriate data owner, how the user is placed into the appropriate access level classification profile, how the user id, password, and access is created and provided to the user. Also include how the user is informed of their responsibilities associated with using the application, given a copy of the access agreement, how to change password, who to call for help, etc.

3b. **Access controls**: Document how the user ids, passwords, and access profiles are added, changed, removed, and documented. The documentation should include who is responsible for these processes.

4. **Sensitive Information Protection**: Provide documentation that identifies sensitive data requiring additional protection. Identify the data owners responsible for this data and the process to be used to protect storage, transmission, printing, and distribution of this data. Include how the password file/field is protected. How will users be prevented from viewing someone else's sensitive information? Are there agreements with outside parties (partners, suppliers, contractors, etc.) concerning the safeguarding of information? If so, what are the obligations?

5. **Audit Trails and Audit Logs**: Identify and document group accounts required by the users or application support, include operating system group accounts. Identify and document individual accounts and/or roles that have super user type privileges, what these privileges are, who has access to these accounts, how access to these accounts are controlled, tracked, logged and how password change and distribution are handled, include operating system accounts. Also identify audit logs, who can read the audit logs, who can modify the audit logs, who can delete the audit logs, and how the audit logs are protected and stored. Is the user id obscured in the audit trails?

6. **External Access Considerations**: Will the application be used internally only? If not, are you compliant with corporate external access requirements?

Architecture Review Checklist - System Management

1. What is the frequency of software changes that must be distributed?
2. What tools are used for software distribution?
3. Are multiple software and/or data versions allowed in production?
4. What is the user data backup frequency and expected restore time?
5. How are user accounts created and managed?
6. What is the system license management strategy?
7. What general system administration tools are required?
8. What specific application administration tools are required?
9. What specific service administration tools are required?
10. How are service calls received and dispatched?
11. Describe how the system is uninstalled.
12. Describe the process or tools available for checking that the system is properly installed.
13. Describe tools or instrumentation that are available that monitor the health and performance of the system.
14. Describe the tools or process in place that can be used to determine where the system has been installed.
15. Describe what form of audit logs are in place to capture system history, particularly after a mishap.
16. Describe the capabilities of the system to dispatch its own error messages to service personnel.

Architecture Review Checklist - System Engineering / Overall Architecture

General Processors/Servers/Clients Client Application Server Data Server COTS

General

1. What other applications and/or systems require integration with yours?
2. Describe the integration level and strategy with each.
3. How geographically distributed is the user base?
4. What is the strategic importance of this system to other user communities inside or outside the enterprise?
5. What computing resources are needed to provide system service to users inside the enterprise? Outside the enterprise and using enterprise computing assets? Outside the enterprise and using their own assets?
6. How can users outside the native delivery environment access your applications and data?
7. What is the life expectancy of this application?
8. Describe the design that accommodates changes in the user base, stored data, and delivery system technology.
9. What is the size of the user base and their expected performance level?
10. What performance and stress test techniques do you use?
11. What is the overall organization of the software and data components?
12. What is the overall service and system configuration?
13. How are software and data configured mapped to the service and system configuration?
14. What proprietary technology (hardware and software) is needed for this system?
15. Describe how each and every version of the software can be reproduced and re-deployed over time.
16. Describe the current user base and how that base is expected to change over the next 3 to 5 years.
17. Describe the current geographic distribution of the user base and how that base is expected to change over the next 3 to 5 years.
18. Describe the how many current or future users need to use the application in a mobile capacity or who need to work off-line.
19. Describe what the application generally does, the major components of the application and the major data flows.
20. Describe the instrumentation included in the application that allows for the health and performance of the application to be monitored.
21. Describe the business justification for the system.
22. Describe the rationale for picking the system development language over other options in terms of initial development cost versus long term maintenance cost.
23. Describe the systems analysis process that was used to come up with the system architecture and product selection phase of the system architecture.
24. Who besides the original customer might have a use for or benefit from using this system?
25. What percentage of the users use the system in browse mode versus update mode?
26. What is the typical length of requests that are transactional?
27. Do you need guaranteed data delivery or update, or the system tolerate failure?
28. What are the up-time requirements of the system?
29. Describe where the system architecture adheres or does not adhere to standards.
30. Describe the project planning and analysis approach used on the project.

Processors/Servers/Clients

1. Describe the Client/Server application architecture.
2. Annotate the pictorial to illustrate where application functionality is executed.

Client

1. Are functions other than presentation performed on the user device?
2. Describe the data and process help facility being provided.
3. Describe the screen to screen navigation technique.
4. Describe how the user navigates between this and other applications.
5. How is this and other applications launched from the user device?
6. Are there any inter-application data and process sharing capabilities? If so, describe what is being shared and by what

technique / technology.
7. Describe data volumes being transferred to the client.
8. What are the additional requirements for local data storage to support the application?
9. What are the additional requirements for local software storage/memory to support the application?
10. Are there any known hardware / software conflicts or capacity limitations caused by other application requirements or situations, which would affect the application users?
11. Describe how the look and feel of your presentation layer compares to the look and feel of the other existing applications.
12. Describe to what extent the client needs to support asynchronous and / or synchronous communication.
13. Describe how the presentation layer of the system is separated from other computational or data transfer layers of the system.

Application Server

1. Can/does the presentation layer and application layers run on separate processors?
2. Can/does the application layer and data access layer run on separate processors?
3. Can this application be placed on an application server independent of all other applications? If not, explain the dependencies.
4. Can additional parallel application servers be easily added? If so, what is the load balancing mechanism?
5. Has the resource demand generated by the application been measured and what is the value? If so, has the capacity of the planned server been confirmed at the application and aggregate levels?

Data Server

1. Are there other applications, which must share the data server? If so, please identify them and describe the data and data access requirements.
2. Has the resource demand generated by the application been measured and what is the value? If so, has the capacity of the planned server been confirmed at the application and aggregate levels?

COTS (where applicable)

1. Is the vendor substantial and stable?
2. Will the enterprise receive source code upon demise of the vendor?
3. Is this software configured for the enterprise's usage?
4. Is there any peculiar A&D data or processes that would impede the use of this software?
    - Is this software currently available?
5. Has it been used/demonstrated for volume/availability/service level requirements similar to those of the enterprise?
    - Describe the past financial and market share history of the vendor.

Architecture Review Checklist - System Engineering / Methods & Tools

1. Do metrics exist for the current way of doing business?
2. Has the system owner created evaluation criteria that will be used to guide the project? Describe how the evaluation criteria will be used.
3. Has research of existing architectures been done to leverage existing work? Describe the method used to discover and understand. Will the architectures be integrated? If so, explain the method that will be used.
4. Describe the methods that will be used on the project:
    - For defining business strategies.
    - For defining areas in need of improvement.
    - For defining as-is and to-be business processes.
    - For defining transition processes.
    - For managing the project.
    - For team communication.
    - For knowledge management, change management & configuration management.
    - For software development.
    - For referencing standards & statements of direction.
    - For quality assurance of deliverables.
    - For design reviews and deliverable acceptance.
    - For capturing metrics.
5. Are the methods documented and distributed to each team member?
6. To what extent are team members familiar with these methods?
7. What processes are in place to ensure compliance with the methods?
8. Describe the infrastructure that is in place to support the use of the methods through the end of the project and anticipated releases?
    - How is consultation and trouble shooting provided?
    - How is training coordinated?
    - How are changes and enhancements incorporated and cascaded?
    - How are lessons learned captured and communicated?
9. What tools are being used on the project? (Please specify versions and platforms). To what extent are team members familiar with these tools?
10. Describe the infrastructure that is in place to support the use of the tools through the end of the project and anticipated releases?
    - How is consultation and trouble shooting provided?
    - How is training coordinated?
    - How are changes and enhancements incorporated and cascaded?
    - How are lessons learned captured and communicated?
11. Describe how the project will promote the reuse of its deliverables and deliverable content.
12. Will the architecture designs “live” after the project has been implemented? Describe the method that will be used to incorporate changes back into the architecture designs.
13. Were the current processes defined?
14. Were issues documented, rated, and associated to current processes? If not, how do you know you are fixing something that is broken?
15. Were existing/planned process improvement activities identified and associated to current processes? If not, how do you know this activity is not in conflict with or redundant to other statements of work?
16. Do you have current metrics? Do you have forecasted metrics? If not, how do you know you are improving something?
17. What processes will you put in place to gather, evaluate and report metrics?
18. What impacts will the new design have on existing business processes, organizations, and information systems? Have they been documented and shared with the owners?

Architecture Contracts

Role Guidelines Contents Relationship to Architecture Governance

Role

Architecture Contracts are the joint agreements between development partners and sponsors on the deliverables, quality, and fitness for purpose of an Architecture. Successful implementation of these agreements will delivered through effective Architecture Governance. By implementing a governed approach to the management of contracts, the following will be ensured:

- A system of continuous monitoring checking integrity, changes, decision-making and audit of all architecture-related activities within the organisation
- Adherence to the principles, standards and requirements of the existing or developing architectures
- Identification of risks in all aspects of the development and implementation of the architecture(s) covering the internal development against accepted standards, policies, technologies and products as well as the operational aspects of the architectures such that the organisation can continue its business within a resilient environment
- A set of processes and practices that ensure accountability, responsibility and discipline with regard to the development and usage of all architectural artefacts.

The traditional Architecture contract is an agreement between the sponsor and the Architecture function or Information Systems department. However, increasingly more services are being provided by Systems Integrators, Applications Providers, and Service Providers, co-ordinated through the Architecture function or Information Systems department. There is therefore a need for an Architecture Contract to establish joint agreements between all parties involved in the Architecture development and delivery.

"Architecture Contracts" may occur at various stages of the ADM; for example:

- The **Statement of Work** created in Phase A is effectively an architecture contract between the architecting organization and the sponsor of the Enterprise Architecture (or the IT Governance function).
- The **development of one or more architecture domains** (Business, Data, Applications, Technology Architecture), and in some cases the oversight of the overall Enterprise Architecture, may be contracted out to Systems Integrators, Applications Providers, and/or Service Providers. Each of these arrangements will normally be governed by an "architecture contract" that defines the deliverables, quality, and fitness for purpose of the developed Architecture, and the processes by which the partners in the architecture development will work together.
- **At the beginning of the Implementation Governance Phase** (at the beginning of Phase G), between the Architecture function and the function responsible for implementing the Enterprise Architecture defined in the preceding ADM phases. Typically, this will be either the in-house Systems Development function, or a major contractor to whom the work is outsourced.
    - What is being "implemented" in the Implementation Governance Phase of the ADM is the overall Enterprise Architecture. This will typically include the technology infrastructure (from Phase D), and also those enterprise applications and data management capabilities that have been defined in the Applications Architecture and Data Architecture (from Phase C), either because they are enterprise-wide in scope, or because they are strategic in business terms, and therefore of enterprise-wide importance and visibility. However, it will typically not include non-strategic business applications, which business units will subsequently deploy on top of the technology infrastructure that is implemented as part of the Enterprise Architecture.
    - In larger-scale implementations, there may well be one Architecture Contract per implementation team in a program of implementation projects.
- **When the Enterprise Architecture has been implemented** (at the end of Phase G), the ADM defines an "Architecture Contract" between the Architecting function (or the IT Governance function, subsuming the Architecting function) and the business users who will subsequently build and deploy business-unit-specific application systems in conformance with the architected environment.

It is important to bear in mind in all these cases that the ultimate goal is not just an enterprise architecture, but a **dynamic** enterprise architecture - i.e., one that allows for flexible evolution in response to changing technology and business drivers, without unnecessary constraints. The Architecture Contract is crucial to enabling a dynamic enterprise architecture.

Typical contents of these three kinds of Architecture Contract are explained below.

Contents

Statement of Architecture Work

The Statement of Work is created as a deliverable of Phase A, and is effectively an architecture contract between the architecting organization and the sponsor of the Enterprise Architecture (or the IT Governance function, on behalf of the enterprise).

Typical contents of a Statement of Architecture Work are:

- Statement of work title
- Project request and background
- Project description and scope
- Architecture vision
- Managerial approach
- Change of scope procedures
- Responsibilities and deliverables
- Acceptance criteria and procedures
- Project plan and schedule
- Support of the Enterprise Continuum
- Signature approvals

Architecture Contract between Architecture Design and Development Partners

This is a signed statement of intent on designing and developing the enterprise architecture, or significant parts of it, from partner organizations, including Systems Integrators, Applications Providers, and Service Providers.

Increasingly the development of one or more architecture domains (Business, Data, Applications, or Technology Architecture) may be contracted out, with the enterprise's Architecture function providing oversight of the overall Enterprise Architecture, and co-ordination and control of the overall effort. In some cases even this oversight role may be contracted out, although most enterprises prefer to retain that core responsibility in-house.

Whatever the specifics of the contracting-out arrangements, the arrangements themselves will normally be governed by an Architecture Contract that defines the deliverables, quality, and fitness for purpose of the developed Architecture, and the processes by which the partners in the architecture development will work together.

Typical contents of an Architecture Design and Development Contract are:

- Introduction & Background
- The nature of the agreement
- Scope of the Architecture
- Architecture and Strategic principles and requirements
- Conformance requirements
- Architecture development and management process and roles
- Target architecture measures
- Defined Phases of deliverables
- Prioritised Joint Workplan
- Time window(s)
- Architecture delivery and business metrics

The template for this contract will normally be defined as part of the Preliminary Phase of the ADM, if not existing already, and the specific contract will be defined at the appropriate stage of the ADM, depending on the particular work that is being contracted out.

Architecture Contract between Architecting Function and Business Users

This is a signed statement of intent to conform with the Enterprise Architecture, issued by enterprise business users. When the Enterprise Architecture has been implemented (at the end of Phase G, Implementation Governance), an Architecture Contract will normally be drawn up between the Architecting function (or the IT Governance function, subsuming the

Architecting function) and the business users who will subsequently be building and deploying application systems in the architected environment.

Typical contents of a Business Users' Architecture Contract are:

Introduction & Background
The nature of the agreement
Scope
Strategic requirements
Architecture deliverables that meet the business requirements
Conformance requirements
Architecture adopters
Time window
Architecture business metrics
Service Architecture (includes Service Level Agreement)

This contract is also used to manage changes to the Enterprise Architecture in Phase H, Architecture Change Management.

Relationship to Architecture Governance

The Architecture Contract document produced in the Implementation Governance Phase of the ADM figures prominently in the area of Architecture Governance, as explained elsewhere in Part IV.

In the context of Architecture Governance, the Architecture Contract is often used as a means of driving architecture change.

In order to ensure that the Architecture Contract is effective and efficient, the following aspects of the governance framework may need to be introduced into the Implementation Governance phase:

- Simple processes
- People-centred authority
- Strong communication
- Timely responses and an effective escalation process
- Supporting organizational structures

Introduction to Architecture Governance

Levels of Governance | The Nature of Governance | Technology Governance - Overview | IT Governance | Architecture Governance

Levels of Governance within the Enterprise

Architecture Governance is the practice and orientation by which enterprise architectures and other architectures are managed and controlled at an enterprise-wide level.

Architecture Governance typically does not operate in isolation, but within a hierarchy of governance structures, which, particularly in the larger enterprise, can include all of the following, as distinct domains with their own disciplines and processes:

- Corporate Governance
- Technology Governance
- Information Technology (IT) Governance
- Architecture Governance

Each of these domains of Governance may exist at multiple geographic levels - global, regional, and local - within the overall enterprise.

Corporate Governance is thus a broad topic, beyond the scope of an enterprise architecture framework such as TOGAF.

This and related subsections are focused on Architecture Governance; but they describe it in the context of enterprise-wide governance, because of the hierarchy of governance structures within which it typically operates, as explained above.

In particular, this and following sections aim to:

- provide an overview of the nature of governance as a discipline in its own right;
- describe the governance context in which Architecture Governance typically functions within the enterprise; and
- describe an Architecture Governance Framework that can be adapted and applied in practice, both for Enterprise Architecture and for other forms of IT Architecture.

The Nature of Governance

Governance - a Generic Perspective

Governance is essentially about ensuring that business is conducted properly. It is less about overt control and strict adherence to rules, and more about guidance and effective and equitable usage of resources to ensure sustainability of an organisation's strategic objectives.

The following outlines the basic principles of corporate governance, as identified by the OECD[1]:

- Focuses on the rights, roles and equitable treatment of shareholders
- Disclosure and transparency and the responsibilities of the board
- Ensures
  - Sound strategic guidance of the organisation
  - Effective monitoring of management by the board
  - Board accountability for the company and to the shareholders
- Board's responsibilities
  - Reviewing and guiding corporate strategy
  - Setting and monitoring achievement of management's performance objectives

Supporting this, the OECD considers a traditional view of governance as the "... system by which business corporations are directed and controlled. The corporate governance structure specifies the distribution of rights and responsibilities among different participants in the corporation, such as, the board, managers, shareholders and other stakeholders, and spells out the rules and procedures for making decisions on corporate affairs. By doing this, it also provides the structure through which the company objectives are set, and the means of attaining those objectives and monitoring performance", OECD (1999).

The Characteristics of Governance

The following characteristics have been adapted from Naidoo (2002) and are positioned here to highlight both the value and necessity for governance as an approach to be adopted within organizations and their dealings with all involved parties:

- **Discipline**
  - All involved parties will have a commitment to adhere to procedures, processes and authority structures established by the organisation
- **Transparency**
  - All actions implemented and their decision support will be available for inspection by authorised organisation and provider parties
- **Independence**
  - All processes, decision-making, and mechanisms used will be established so as to minimize or avoid potential conflicts of interest
- **Accountability**
  - Identifiable groups with the organization, e.g. Governance Boards, who take actions or make decisions are authorised and accountable for their actions
- **Responsibility**
  - Each contracted party is required to act responsibly to the organisation and its stakeholders
- **Fairness**
  - All decisions taken, processes used and their implementation will not be allowed to create unfair advantage to any one particular party.

1 OECD Principles of Corporate Governance, (1999), Organisation for Economic Co-operation and Development [Online], available at: http://www.oecd.org/EN/document/0,,EN-document-76-3-no-15-8293-0,00.html [2001, December].

Technology Governance

Technology Governance is a key capability, requirement, and resource for most organizations because of the pervasiveness of technology across the organizational spectrum.

Recent studies have shown that many organizations have a balance in favour of intangibles rather than tangibles that require management. Given that most of these intangibles are informational and digital assets, it is evident that businesses are becoming more reliant on information technology: and the governance of information technology - IT Governance - is therefore becoming an even more important part of Technology Governance.

These trends also highlight the dependencies of businesses on not only the information itself but also the processes, systems, and structures that create, deliver, and consume it. As the shift to increasing value through intangibles increases in many industry sectors, so risk management must be considered as key to understanding and moderating new challenges, threats and opportunities.

Not only are organizations increasingly dependent on information technology for their operations and profitability, but also their reputation, brand, and ultimately their value are also dependent on that same information and the supporting technology.

IT Governance

IT Governance provides the framework and structure that links IT resources and information to enterprise goals and strategies. Furthermore, IT Governance institutionalizes best practices for planning, acquiring, implementing, and monitoring IT performance, to ensure that the enterprise's information technology assets support its business objectives.

In recent years, IT Governance has become integral to the effective governance of the modern enterprise. Businesses are increasingly dependent on information technology to support critical business functions and processes; and to successfully gain competitive advantage, businesses need to manage effectively the complex technology that is pervasive throughout the organisation, in order to respond quickly and safely to business needs.

In addition, regulatory environments around the world are increasingly mandating stricter enterprise control over information, driven by increasing reports of information system disasters and electronic fraud. The management of IT related risk is now widely accepted as a key part of enterprise governance.

It follows that an IT Governance strategy, and an appropriate organization for implementing the strategy, must be established with the backing of top management, clarifying who owns the enterprise's IT resources, and, in particular, who has ultimate responsibility for their enterprise-wide integration.

An IT Governance Framework - COBIT

As with Corporate Governance, IT Governance is a broad topic, beyond the scope of an enterprise architecture framework such as TOGAF. A good source of detailed information on IT Governance is the COBIT Framework (**C**ontrol **OB**jectives for **I**nformation and related **T**echnology). This is an open standard for control over information technology, developed and promoted by the IT Governance Institute, and published by the Information Systems Audit and Control Foundation (ISACF).

COBIT also provides a generally accepted standard for good IT security and control practices to support the needs of enterprise management in determining and monitoring the appropriate level of IT security and control for their organisations.

The IT Governance Institute has also developed and built into the COBIT framework a set of Management Guidelines for COBIT, which consist of Maturity Models, Critical Success Factors (CFSs), Key Goal Indicators (KGIs), and Key Performance Indicators (KPIs). The framework responds to management's need for control and measurability of IT, by providing management with tools to assess and measure their organisation's IT environment against the IT processes that COBIT identifies.

Architecture Governance - Overview

Architecture Governance Characteristics

Architecture governance is the practice and orientation by which enterprise architectures and other architectures are managed and controlled at an enterprise-wide level. It includes the following:

- Implementing a system of controls over the creation and monitoring of all architectural components and activities, to ensure the effective introduction, implementation, and evolution of architectures within the organization.
- Implementing a system to ensure compliance with internal and external standards and regulatory obligations.
- Establishing processes that support effective management of the above processes within agreed parameters.
- Developing practices that ensure accountability to a clearly identified stakeholder community - both inside and outside the organization.

Architecture Governance as a Board Level Responsibility

As mentioned above, IT governance has recently become a board responsibility as part of overall business governance. The governance of an organization's architectures is a key factor in effective IT/Business linkage, and is therefore increasing becoming a key board level responsibility in its own right, .

This section aims to provide the impetus for opening up IT and Architecture Governance so that the business responsibilities associated with Architecture activities and artefacts can be elucidated and managed.

TOGAF and Architecture Governance

Phase G of the TOGAF Architecture Development Method is dedicated to Implementation Governance, which concerns itself with the realization of the architecture through change projects. Implementation Governance is just one aspect of Architecture Governance, which covers the management and control of all aspects of the development and evolution of enterprise architectures and other architectures within the enterprise.

Architecture Governance needs to be supported by an Architecture Governance Framework, described in detail in the following section, which assists in identifying effective processes so that the business responsibilities associated with architecture governance can be elucidated, communicated, and managed effectively.

Architecture Governance Framework

Introduction

As previously explained, Phase G of the TOGAF Architecture Development Method is dedicated to Implementation Governance, which concerns itself with the realization of the architecture through change projects.

Implementation Governance is just one aspect of Architecture Governance, which covers the management and control of all aspects of the development and evolution of enterprise architectures and other architectures within the enterprise.

Architecture Governance needs to be supported by an Architecture Governance Framework, described in detail below. The governance framework described in the following is a generic framework that can be adapted to the existing governance environment of an enterprise. It is intended to assist in identifying effective processes and organizational structures, so that the business responsibilities associated with Architecture Governance can be elucidated, communicated, and managed effectively.

Architecture Governance Framework - Conceptual Structure

Key Concepts

Conceptually, Architecture Governance is an approach, a series of processes, a cultural orientation, and set of owned responsibilities that ensure the integrity and effectiveness of the organization's architectures.

The key concepts are illustrated in Figure 1.

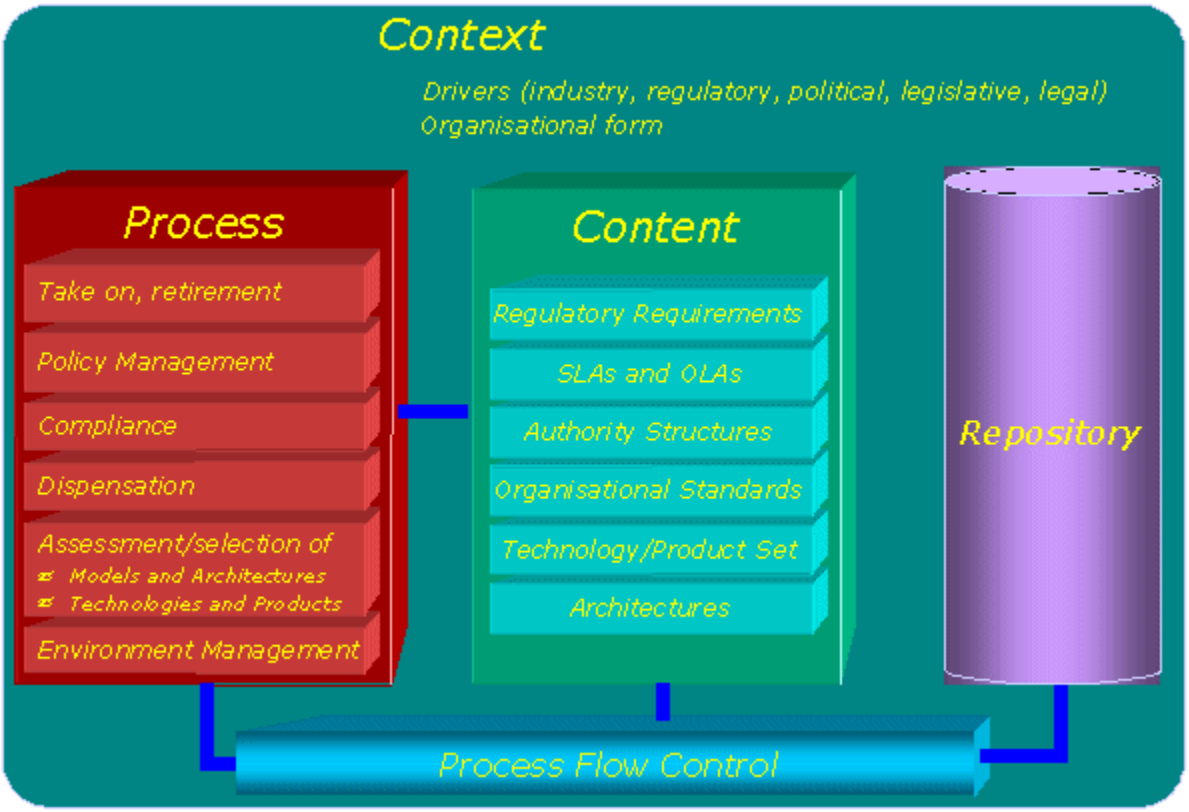


Figure 1: TOGAF Architecture Governance Framework- Conceptual

The split of **process**, **content** and **context** are key to the support of the architecture governance initiative, by allowing the introduction of new governance material (legal, regulatory, standards-based or legislative) without unduly impacting the processes. This *content-agnostic* approach ensures that the framework is flexible. The processes are typically independent of the content and implement a proven best-practice approach to active governance.

The Architecture Governance Framework is integral to the Enterprise Continuum, and manages all content relevant both to the architecture itself and to architecture governance processes.

Key Architecture Governance Processes

Governance processes are required to identify, manage, audit and disseminate all information related to architecture management, contracts and implementation. These governance processes identifies several processes that will be used to ensure that all architecture artefacts and contracts, principles and operational level agreements are monitored on an ongoing basis with clear auditability of all decisions made.

Policy management and take-on

All architecture amendments, contracts and supporting information must come under governance through a formal process in order to register, validate, ratify, manage and publish new or updated content. These processes will ensure the orderly integration with existing governance content such that all relevant parties, documents, contracts and supporting information are managed and audited.

Compliance

Compliance assessments against SLAs, OLAs, standards and regulatory requirements will be implemented on an ongoing basis to ensure stability, conformance and performance monitoring. These assessments will be reviewed and either accepted or rejected depending on the criteria defined within the governance framework.

Dispensation

A compliance assessment can be rejected where the subject area (design, operational, service level or technology) are not compliant. In this case the subject area can

a. be adjusted or realigned in order to meet the compliance requirements
b. request a dispensation.

Where a compliance assessment is rejected an alternate route to meeting interim conformance is provided through dispensations. These are granted for a given time period and set of identified service and operational criteria that must be enforced during the life-span of the dispensation. Dispensations are not granted indefinitely, but are used as a mechanism to ensure that service levels and operational levels are met while providing a level flexibility in their implementation and timing. The time-bound nature of dispensations ensures that they are a major trigger in the compliance cycle.

Monitoring and Reporting

Performance management is required to ensure that both the operational and service elements are managed against an agreed set of criteria. This will include monitoring against service and operational level agreements, feedback for adjustment and reporting.

Internal management information will be considered under Environment Management.

Business Control

Business Control relates to the processes invoked to ensure compliance with the organisation's business policies.

Environment Management

This identifies all the services required to ensure that the repository-based environment underpinning the governance framework is effective and efficient. This includes the physical and logical repository management, access, communication, training and accreditation of all users.

All architecture artefacts, service agreements, contracts and supporting information must come under governance through a formal process in order to register, validate, ratify, manage and publish new or updated content. These will processes will ensure the orderly integration with existing governance content such that all relevant parties, documents, contracts and supporting information are managed and audited.

The governance environment will have a number of administrative processes defined in order to effective a managed service and process environment. These processes will include user management, internal SLAs (defined in order to control its own processes) and management information reporting.

Architecture Governance Framework - Organizational Structure

Overview

Architecture Governance is the practice and orientation by which enterprise architectures and other architectures are managed and controlled. In order to ensure that this control is effective within the organisation, it is necessary to have the correct organizational structures established to support all governance activities.

An Architecture Governance structure for effectively implementing the approach described in this section will typically include the following levels, which may in practice involve a combination of existing IT governance processes, organizational structures, and capabilities. They will typically include the following:

- Global governance board
- Local governance board
- Design authorities
- Working parties.

The architecture organization illustrated in Figure 2 highlights the major structural elements required for an architecture governance initiative. While each enterprise will have differing requirements, it is expected that the basics of the organizational design shown in Figure 2 will be applicable and implementable in a wide variety of organizational types.

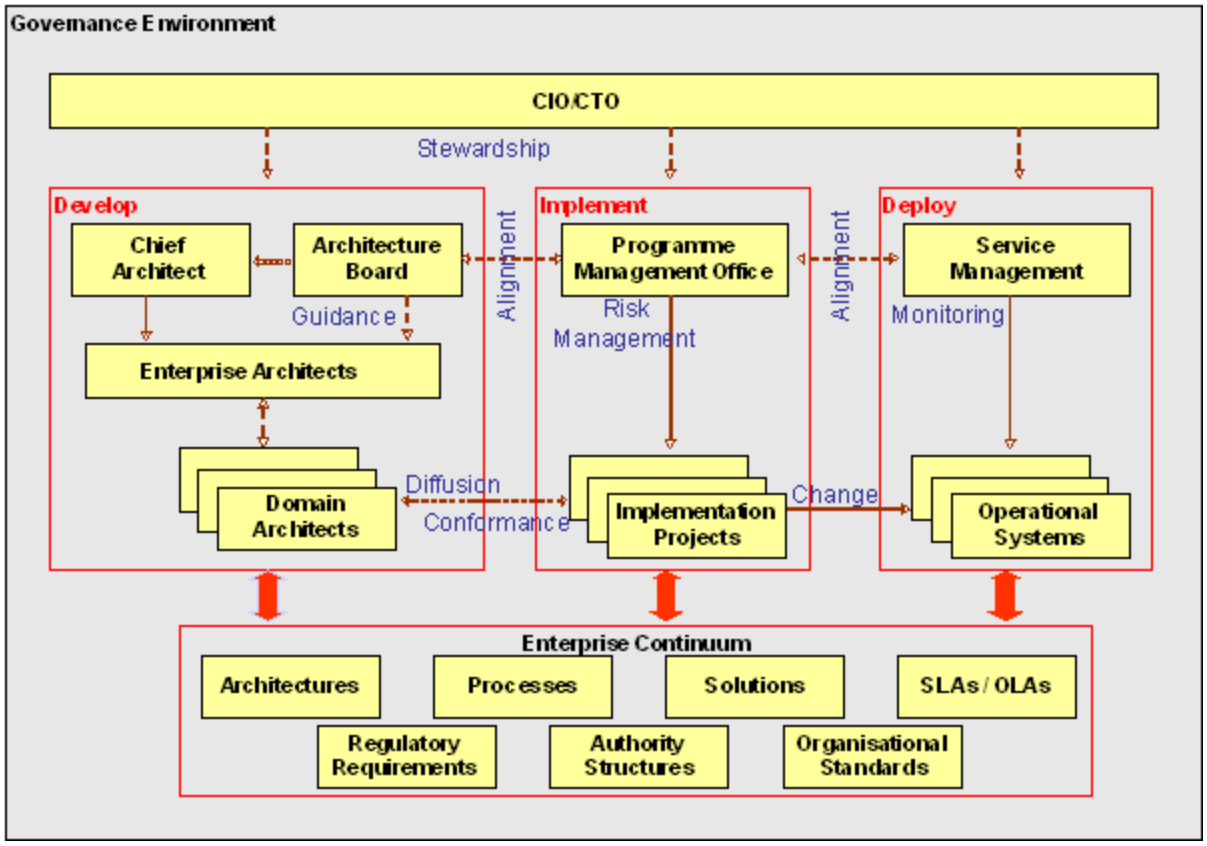


Figure 2: TOGAF Architecture Governance Framework- Organizational

Key Areas

Figure 2 identifies three key areas of the architecture management - **Develop**, **Implement**, and **Deploy**. Each of these is the responsibility of one or more groups within the organization, while the **Enterprise Continuum** is shown to support all activities and artefacts associated with the governance of the architectures throughout their lifecycle.

The **Develop** responsibilities, processes, and structures are usually linked to the TOGAF ADM and its usage, while the **Implement** responsibilities, processes, and structures are typically linked to TOGAF Implementation Governance Phase.

As mentioned above, the Architecture Governance Framework is integral to the Enterprise Continuum, and manages all content relevant both to the architectures themselves and to Architecture Governance processes.

Operational Benefits

As illustrated in Figure 2, the governance of the organization's architectures provides not only direct control and guidance of their development and implementation, but also extends into the operations of the implemented architectures.

The following benefits have been found to be derived through the continuing governance of architectures:

- Links IT processes, resources and information to organizational strategies and objectives
- Integrates and institutionalizes IT best practices
- Aligns with industry frameworks such as COBIT (planning and organising, acquiring and implementing, delivering and supporting, and monitoring IT performance)
- Enables the organization to take full advantage of its information, infrastructure, and hardware and software assets
- Protects the underlying digital assets of the organisation
- Supports regulatory and best practice requirements such as auditability, security, responsibility, and accountability
- Promotes visible risk management

These benefits position the TOGAF Architecture Governance Framework as an approach, a series of processes, a cultural orientation, and a set of owned responsibilities, that together ensures the integrity and effectiveness of the organization's architectures.

Architecture Governance in Practice

Key Success Factors Elements of an Effective Architecture Governance Strategy

Architecture Governance - Key Success Factors

It is important to consider the following to ensure a successful approach to architecture governance, and to the effective management of the Architecture Contract:

- Establishment and operation of best practice for the submission, adoption, reuse, reporting, and retirement of architecture policies, procedures, roles, skills, organisational structures, and support services.
- Establishment of the correct organizational responsibilities and structures to support the architecture governance processes and reporting requirements.
- Integration of tools and process to facilitate the take up of the processes, both procedurally and culturally.
- Management of criteria for the control of the architecture governance processes, dispensations, compliance assessments, Service Level Agreements and Operational Level Agreements.
- Meeting the internal and external requirements for the effectiveness, efficiency, confidentiality, integrity, availability, compliance, and reliability of all architecture governance related information, services and processes.

Elements of an Effective Architecture Governance Strategy

Architecture Governance and Corporate Politics

There is a similarity between enterprise architecture and architecture in the physical world, in that politics has an important role to play in the acceptance of both architectures. In the real world, it is the dual politics of the environment and commerce, while in the world of enterprise architecture a consideration of corporate politics is critical.

An enterprise architecture imposed without appropriate political backing is bound to fail. In order to succeed, the enterprise architecture must reflect the needs of the organization. Enterprise Architects, if they are not involved in the development of business strategy, must at least have a fundamental understanding of it and of the prevailing business issues facing the organisation. It may even be necessary for them to be involved in the system deployment process and to ultimately own the investment and product selection decisions arising from the implementation of the technical architecture.

Key Strategic Elements

There are three important elements of Architecture Governance strategy that relate particularly to the acceptance and success of architecture within the enterprise. While relevant and applicable in their own right apart from their role in Governance, and therefore described separately, they also from an integral part of any effective Architecture Governance Strategy:

- A cross-organizational Architecture Board must be established with the backing of top management to oversee the implementation of the IT governance strategy.
- A comprehensive set of Architectural Principles should be established, to guide, inform and support the way in which an organization sets about fulfilling its mission through the use of IT.
- An Architecture Compliance strategy should be adopted - specific measures (more than just a statement of policy) to ensure compliance with the architecture, including project impact assessments, a formal architecture compliance review process, and possibly including the involvement of the architecture team in product procurement.

Architecture Maturity Models

Overview

Organizations that can manage change effectively are generally more successful than those that can not. Many organizations know that they need to improve their IT related development processes in order to successfully manage change, but don't know how. Such organizations typically either spend very little on process improvement, because they are unsure how best to proceed; or spend a lot, on a number of parallel and unfocussed efforts, to little or no avail.

Capability Maturity Models address this problem by providing an effective and proven method for an organization to gradually gain control over and improve its IT related development processes. Such models provide the following benefits:

- They describe the practices that any organization must perform in order to improve its processes
- They provide a yardstick against which to periodically measure improvement
- They constitute a proven framework within which to manage the improvement efforts.

The various practices are typically organized into 5 levels, each level representing an increased ability to control and manage the development environment.

An evaluation of the organization's practices against the model, called an *assessment*, determines the level at which the organization currently stands. It indicates the organization's maturity in the area concerned, and the practices on which the organization needs to focus in order to see the greatest improvement and the highest return on investment.

The benefits of Capability Maturity Models are well documented for Software and Systems Engineering. Their application to Enterprise Architecture has been a recent development, stimulated by the increasing interest in Enterprise Architecture in recent years, combined with the lack of maturity in this discipline.

This section introduces into TOGAF the topic of Capability Maturity Models and their associated methods and techniques, as a widely used industry standard that is mature enough to consider for use in relation to Enterprise Architecture.

Background

The Software Engineering Institute (SEI), a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University, developed the original Capability Maturity Model -- the Software CMM®, Capability Maturity Model for Software -- in the early 1990s, which is still widely used today.

Capability Maturity Models have gained wide scale acceptance over the last decade. These models and their associated methods were originally applied to IT solutions, particularly software Solutions, but a number of IT related disciplines have developed capability maturity models to support process improvement in areas such as:

- People - the People Capability Maturity Model® (P-CMM®), and the The IDEAL Life Cycle Model for Improvement
- Systems Engineering (the Systems Engineering CMM®)
- Software Acquisition (the Software Acquisition CMM®)
- Capability Maturity Model Integration – CMMI[SM]

The models have been adopted by large organizations, including the US Dept of Commerce, the US DoD, the UK Government, and a number of large services organisations, to assess competencies.

The increasing interest in applying these techniques to the IT Architecture and Enterprise Architecture fields has resulted in a series of template tools which assess:

- the state of the IT Architecture process,
- the IT Architecture,
- the organization's buy-in to both

The main issues addressed by US and UK Government use of these models include:

- eCommerce Maturity
- Process implementation & audit
- Quality measurements
- People Competencies
- Investment Management

They involve use of a multiplicity of models, and focus in particular on measuring business benefits and returns on investment.

Another key driver is the increasing use of outsourcing. Recent analyst projections indicate that around 75 percent of IS organizations are refocusing their role on brokering resources and facilitating business-driven demands, rather than on being direct providers of IT services. The CMM® is increasingly the standard by which outsourcers are being evaluated.

This section reviews the state of development in such techniques.

A closely related topic is that of the Architecture Skills Framework, which can be used to plan the target skills and capabilities required by an organization to successfully delivery an Enterprise Architecture, and to determine the training and development needs of individuals.

A Capability Model for IT Architecture - The US Department of Commerce's ACMM Framework

Overview

As an example of the trend towards increased interest in applying CMM techniques to IT Architecture, all US Federal Agencies are now expected to provide Maturity Models and ratings as part of their IT investment management and audit requirements.

In particular, the US Department of Commerce (DoC) has developed an IT Architecture Capability Maturity Model (ACMM) to aid in conducting internal assessments. The ACMM provides a framework that represents the key components of a productive IT Architecture process. The goal is to enhance the overall odds for success of IT Architecture by identifying weak areas and providing a defined evolutionary path to improving the overall Architecture process.

The ACMM comprises three sections:

1. The IT Architecture Maturity Model
2. IT Architecture Characteristics of Operating Units' Processes at Different Maturity Levels
3. The IT Architecture Capability Maturity Model Scorecard.

The first two sections explain the Architecture Capability Maturity levels and the corresponding IT Architecture characteristics for each maturity level to be used as measures in the assessment process. The third section is used to derive the Architecture Capability Maturity level that is to be reported to the Department of Commerce Chief Information Officer.

Elements of the ACCM

The Department of Commerce IT Architecture Capability Maturity Model consists of six levels and nine architecture characteristics. The six levels are:

0. None
1. Initial
2. Under Development
3. Defined
4. Managed

5. Measured.

The nine IT Architecture Characteristics are:

- IT Architecture Process
- IT Architecture Development
- Business Linkage
- Senior Management Involvement
- Operating Unit Participation
- Architecture Communication
- IT Security
- Architecture Governance
- IT Investment and Acquisition Strategy

Two complementary methods are used in the ACMM to calculate a maturity rating. The first method obtains a weighted mean IT Architecture Maturity Level. The second method shows the percent achieved at each maturity level for the nine architecture characteristics.

Example - IT Architecture Process Maturity Levels

The following example shows the detail of the IT Architecture Maturity Levels as applied to the first of the nine characteristics, IT Architecture Process.

- **Level 0** – **None**.
  No IT Architecture Program. No IT Architecture to speak of.

- **Level 1** – **Initial**.
  Informal IT Architecture Process Underway

  1. Processes are ad hoc and localized. Some IT Architecture processes are defined. There is no unified architecture process across technologies or business processes. Success depends on individual efforts.
  2. IT Architecture processes, documentation and standards are established by a variety of ad hoc means and are localized or informal.
  3. Minimal, or implicit linkage to business strategies or business drivers.
  4. Limited management team awareness or involvement in the architecture process.
  5. Limited. Operating Unit acceptance of the IT Architecture process.
  6. The latest version of the Operating Unit's IT Architecture documentation is on the Web. Little communication exists about the IT Architecture process and possible process improvements.
  7. IT Security considerations are ad hoc and localized.
  8. No explicit governance of architectural standards.
  9. Little or no involvement of strategic planning and acquisition personnel in enterprise architecture process. Little or no adherence to existing Standards.

- **Level 2** – **Under Development**.
  IT Architecture Process Is Under Development

  1. Basic IT Architecture Process program is documented based on OMB Circular A - 130 and Department of Commerce IT Architecture Guidance. The architecture process has developed clear roles and responsibilities.
  2. IT Vision, Principles, Business Linkages, Baseline, and Target Architecture are identified. Architecture standards exist, but not necessarily linked to Target Architecture. Technical Reference Model and Standards Profile framework established.
  3. Explicit linkage to business strategies.
  4. Management awareness of Architecture effort.
  5. Responsibilities are assigned and work is underway.
  6. The DoC and Operating Unit IT Architecture Web Pages are updated periodically and is used to document architecture deliverables.
  7. IT Security Architecture has defined clear roles and responsibilities.
  8. Governance of a few architectural standards and some adherence to existing Standards Profile.
  9. Little or no formal governance of IT Investment and Acquisition Strategy. Operating Unit demonstrates some adherence to existing Standards Profile.

- **Level 3** – **Defined**.

Defined IT Architecture Including Detailed Written Procedures and Technical Reference Model

1. The architecture is well defined and communicated to IT staff and business management with Operating Unit IT responsibilities. The process is largely followed.
2. Gap Analysis and Migration Plan are completed. Fully developed Technical Reference Model and Standards Profile. IT goals and methods are identified.
3. IT Architecture is integrated with capital planning & investment control.
4. Senior-management team aware of and supportive of the enterprise-wide architecture process. Management actively supports architectural standards.
5. Most elements of Operating Unit show acceptance of or are actively participating in the IT Architecture process.
6. Architecture documents updated regularly on DoC IT Architecture Web Page.
7. IT Security Architecture Standards Profile is fully developed and is integrated with IT Architecture.
8. Explicit documented governance of majority IT investments.
9. IT acquisition strategy exists and includes compliance measures to IT Enterprise Architecture. Cost-benefits are considered in identifying projects.

- **Level 4** – **Managed.**
Managed and Measured IT Architecture Process

1. IT Architecture process is part of the culture. Quality metrics associated with the architecture process are captured.
2. IT Architecture documentation is updated on a regular cycle to reflect the updated IT Architecture. Business, Information, Application and Technical Architectures defined by appropriate de-jure and de-facto standards.
3. Capital planning and investment control are adjusted based on the feedback received and lessons learned from updated IT Architecture. Periodic re-examination of business drivers.
4. Senior-management team directly involved in the architecture review process.
5. The entire Operating Unit accepts and actively participates in the IT Architecture process.
6. Architecture documents are updated regularly, and frequently reviewed for latest architecture developments/standards.
7. Performance metrics associated with IT Security Architecture are captured.
8. Explicit governance of all IT investments. Formal processes for managing variances feed back into IT Architecture.
9. All planned IT acquisitions and purchases are guided and governed by the IT Architecture.

- **Level 5 – Optimizing**.
Continuous Improvement of IT Architecture Process

1. Concerted efforts to optimize and continuously improve architecture process.
2. A standards and waivers process are used to improve architecture development process improvements.
3. Architecture process metrics are used to optimize and drive business linkages. Business involved in the continuous process improvements of IT Architecture.
4. Senior management involvement in optimizing process improvements in Architecture development and governance.
5. Feedback on architecture process from all Operating Unit elements is used to drive architecture process improvements.
6. Architecture documents are used by every decision maker in the organization for every IT-related business decision.
7. Feedback from IT Security Architecture metrics are used to drive architecture process improvements.
8. Explicit governance of all IT investments. A standards and waivers process is used to improve governance-process improvements.
9. No unplanned IT investment or acquisition activity.

Capability Maturity Models Integration (CMMI)

Introduction

The capability models that the SEI is currently involved in developing, expanding, or maintaining, include the following:

- SW-CMM® Capability Maturity Model for Software

- P-CMM People Capability Maturity Model
- SA-CMM Software Acquisition Capability Maturity Model
- SE-CMM Systems Engineering Capability Maturity Model
- IPD-CMM Integrated Product Development Capability Maturity Model
- Capability Maturity Model Integration (CMMI[SM])

As explained in the Introduction, in recent years the industry has witnessed significant growth in the area of maturity models. The multiplicity of models available has lead to problems of its own, in terms of how to integrate all the different models to produce a meaningful metric for overall process maturity.

In response to this need, the SEI has developed a Framework called Capability Maturity Model Integration (CMMI[SM]), to provide a means of managing the complexity.

According to the SEI, the use of the CMMI models improves on the best practices of previous models in many important ways, in particular enabling organizations to:

- more explicitly link management and engineering activities to business objectives
- expand the scope of and visibility into the product life cycle and engineering activities to ensure that the product or service meets customer expectations
- incorporate lessons learned from additional areas of best practice (e.g., measurement, risk management, and supplier management)
- implement more robust high-maturity practices
- address additional organizational functions critical to its products and services
- more fully comply with relevant ISO standards

CMMI is being adopted world-wide.

The SCAMPI Method

The Standard CMMI Appraisal Method for Process Improvement (SCAMPI[SM]) is the appraisal method associated with CMMI. The SCAMPI appraisal method is used to identify strengths, weaknesses, and ratings relative to CMMI reference models. It incorporates best practices found successful in the appraisal community, and is based on the features of several legacy appraisal methods. It is applicable to a wide range of appraisal usage modes, including both internal process improvement and external capability determinations.

The SCAMPI Method Definition Document describes the requirements, activities, and practices associated with each of the processes that compose the SCAMPI method.

Conclusions

This section has sought to introduce into TOGAF the topic of CMM based methods and techniques, as a widely used industry standard that is mature enough to consider for use in relation to Enterprise Architecture.

The benefits of Capability Maturity Models are well documented for Software and Systems Engineering. Their application to Enterprise Architecture has been a more recent development, stimulated by the increasing interest in Enterprise Architecture, combined with the lack of maturity in the discipline of Enterprise Architecture.

Future Versions of TOGAF will seek to build on this base, as more experience is gained on the use of these methods and techniques specifically relating to Enterprise Architecture.

Architecture Patterns

Introduction U.S. Treasury Architecture Development Guidance IBM's Patterns of e-Business

Introduction

Patterns for system architecting are very much in their infancy. They have been introduced into TOGAF essentially to draw them to the attention of the systems architecture community as an emerging important resource, and as a placeholder for hopefully more rigorous descriptions and references to more plentiful resources in future versions of TOGAF.

They have not (as yet) been integrated into TOGAF. However, in the following, we attempt to indicate the potential value to TOGAF, and to which parts of the TOGAF ADM they might be relevant.

Background

A "pattern" has been defined as “an idea that has been useful in one practical context and will probably be useful in others.” [M.Fowler, “Analysis Patterns – Reusable Object Models, Addison Wesley, ISBN 0-201-89542-0].

In TOGAF, patterns are considered to be a way of putting building blocks into context: for example, to describe a reusable solution to a problem. Building Blocks are what you use: patterns can tell you how you use them, when, why, and what trade-offs you have to make in doing so.

Patterns offer the promise of helping the architect identify combinations of architectural and/or solution building blocks that have been proven to deliver effective solutions in the past, and may provide the basis for effective solutions in the future.

Pattern techniques are generally acknowledged to have been established as a valuable architectural design technique by Christopher Alexander, a buildings architect, who described this approach in his book "The Timeless Way of Building", Oxford University Press, 1979, ISBN 0-19-502402-8. This book provides an introduction to the ideas behind the use of patterns, and Alexander followed it with two further books (A Pattern Language, and The Oregon Experiment) in which he expanded on his description of the features and benefits of a patterns approach to architecture.

Software and buildings architects have many similar issues to address, and so it was natural for software architects to take an interest in patterns as an architectural tool. Many papers and books have been published on them since Alexander's 1979 book, perhaps the most renowned being "Design Patterns: Elements of Reusable Object-Oriented Software", by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Addison Wesley, October 1994, ISBN 0-201-63361-2. This book describes simple and elegant solutions to specific problems in object-oriented software design.

Content of a Pattern

Several different formats are used in the literature for describing patterns, and no single format has achieved widespread acceptance. However, there is broad agreement on the types of things that a pattern should contain. The headings which follow are taken from *Pattern-Oriented Software Architecture: A System of Patterns*, by F. Buschmann, R. Meunier, H. Rohnert, P.Sommerlad, and M. Stal, John Wiley and Sons, 1996, ISBN 0-471-95869-7. The elements described below will be found in most patterns, even if different headings are used to describe them.

- **Name**: A meaningful and memorable way to refer to the pattern, typically a single word or short phrase.
- **Problem**: A description of the problem indicating the intent in applying the pattern - the intended goals and objectives to be reached within the context and forces described below (perhaps with some indication of their priorities).
- **Context**: The pre-conditions under which the pattern is applicable - a description of the initial state before the pattern is applied.
- **Forces**: A description of the relevant forces and constraints, and how they interact/conflict with each other and with the intended goals and objectives. The description should clarify the intricacies of the problem and make explicit the kinds of trade-offs that must be considered. (The need for such trade-offs is typically what makes the problem difficult, and generates the need for the pattern, in the first place.) The notion of "forces" equates in many ways to the "qualities" that architects seek to optimize, and the concerns they seek to address, in designing architectures. For

example:
    - Security, robustness, reliability, fault-tolerance
    - Manageability
    - Efficiency, performance, throughout, bandwidth requirements, space utilization
    - Scalability (incremental growth on-demand)
    - Extensibility, evolvability, maintainability
    - Modularity, independence, reusability, openness, composability (plug 'n play), portability
    - Completeness and correctness
    - Ease of construction
    - Ease of use
    - ... etc.
- **Solution**: A description, using text and/or graphics, of how to achieve the intended goals and objectives. The description should identify both the solution's static structure and its dynamic behavior - the people and computing actors, and their collaborations. The description may include guidelines for implementing the solution. Variants or specializations of the solution may also be described.
- **Resulting Context:** The post-conditions after the pattern has been applied. Implementing the solution normally requires trade-offs among competing forces. This element describes which forces have been resolved and how, and which remain unresolved. It may also indicate other patterns that may be applicable in the new context. (A pattern may be one step in accomplishing some larger goal.) Any such other patterns will be described in detail under *Related Patterns*.
- **Examples**: One or more sample applications of the pattern which illustrate each of the other elements: a specific problem, context, and set of forces; how the pattern is applied; and the resulting context.
- **Rationale**: An explanation / justification of the pattern as a whole, or of individual components within it, indicating how the pattern actually works, and why - how it resolves the forces to achieve the desired goals and objectives, and why this is "good". The Solution element of a pattern describes the external structure and behavior of the solution: the Rationale provides insight into its internal workings.
- **Related Patterns**: The relationships between this pattern and others. These may be **predecessor** patterns, whose resulting contexts correspond to the initial context of this one; or **successor** patterns, whose initial contexts correspond to the resulting context of this one; or **alternative** patterns, which describe a different solution to the same problem, but under different forces; or **co-dependent** patterns, which may / must be applied along with this pattern.
- **Known Uses**: Known applications of the pattern within existing systems, verifying that the pattern does indeed describe a proven solution to a recurring problem. Known Uses can also serve as Examples.

Patterns may also begin with an Abstract providing an overview of the pattern and indicating the types of problems it addresses. The Abstract may also identify the target audience and what assumptions are made of the reader.

Terminology

Although design patterns have been the focus of widespread interest in the software industry for several years, particularly in the object-oriented and component based software fields, it is only recently that there has been increasing interest in architecture patterns - extending the principles and concepts of design patterns to the architecture domain.

The technical literature relating to this field is complicated by the fact that many people in the software field use the term "architecture" to refer to software, and many patterns described as "architecture patterns" are high-level software design patterns. This simply makes it all the more important to be precise in use of terminology.

Architecture Patterns and Design Patterns

The term "design pattern" is often used to refer to any pattern which addresses issues of software architecture, design, or programming implementation. In *Pattern-Oriented Software Architecture: A System of Patterns*, by F. Buschmann, R. Meunier, H. Rohnert, P.Sommerlad, and M. Stal, John Wiley and Sons, 1996, ISBN 0-471-95869-7, the authors define these three types of patterns as follows:

- An **Architecture Pattern** expresses a fundamental structural organization or schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.
- A **Design Pattern** provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes commonly recurring structure of communicating components that solves a general design problem within a particular context.
- An **Idiom** is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language.

These distinctions are useful, but it is important to note that "architecture patterns" in this context still refers solely to software architecture. Software architecture is certainly an important part of the focus of TOGAF, but it is not its only focus.

In this section we are concerned with **patterns for enterprise system architecting**. These are analogous to software architecture and design patterns, and borrow many of their concepts and terminology, but focus on providing re-usable models and methods specifically for the architecting of enterprise information systems - comprising software, hardware, networks, and people - as opposed to purely software systems.

Patterns and the Architecture Continuum

Although architecture patterns have not (as yet) been integrated into TOGAF, each of the first four main phases of the ADM (Phases A through D) gives an indication of the stage at which relevant reusable architecture assets from the enterprise's Architecture Continuum should be considered for use. Architecture patterns are one such asset.

An enterprise that adopts a formal approach to use and reuse of architecture patterns will normally integrate their use into the enterprise's Architecture Continuum.

Patterns and Views

Architecture Views are selected parts of one or more models representing a complete system architecture, focusing on those aspects that address the concerns of one or more stakeholders. Patterns can provide help in designing such models, and in composing views based on them.

Patterns and Business Scenarios

Relevant architecture patterns may well be identified in the work on Business Scenarios.

Architecture Patterns In Use

Two examples of architecture patterns in use are outlined in the following subsections, one from the domain of an IT customer enterprise's own architectural framework, and the other from a major system vendor who has done a lot of work in recent years in the field of architecture patterns.

- The U.S. Treasury Architecture Development Guidance (TADG) document provides a number of explicit architecture patterns, in addition to explaining a rationale, structure, and taxonomy for architecural patterns as they relate to the U.S. Treasury.
- The IBM Patterns for e-Business web site gives a series of architecture patterns that go from the busines problem to specific solutions, firstly at a generic level and then in terms of specific IBM product solutions. A supporting resource is IBM's set of "Red Books".

The following material is intended to give the reader pointers to some of the places where architecture patterns are already being used and made available, in order to help readers make their own minds up as to the usefulness of this technique for their own environments.

- ---

U.S. Treasury Architecture Development Guidance (TADG) document

The U.S. Treasury Architecture Development Guidance (TADG) document - formerly known as the Treasury Information System Architecture Framework (TISAF) - provides a number of explicit architecture patterns.

Section 7 of the TADG document describes a rationale, structure, and taxonomy for architecural patterns, while the patterns themselves are formally documented in Appendix D. The architecture patterns presented embrace a larger set of systems than just object-oriented systems. Some architecture patterns are focused on legacy systems, some on concurrent and distributed systems, and some on real-time systems.

TADG Pattern Content

The content of an architecture pattern as defined in the TADG document contains the following elements:

- **Name**. Each architecture pattern has a unique, short descriptive name. The collection of architecture pattern names can be used as a vocabulary for describing, verifying, and validating information system architectures.
- **Problem**. Each architecture pattern contains a description of the problem to be solved. The problem statement may

describe a class of problems or a specific problem.

- **Rationale**. The rationale describes an explains a typical specific problem that is representative of the broad class of problems to be solved by the architecture pattern. For a specific problem, it can provide additional details of the nature of the problem and the requirements for its resolution.
- **Assumptions**. The assumptions are conditions that must be satisfied in order for the architecture pattern to be usable in solving the problem. They include constraints on the solution and optional requirements that may make the solution more easy to use.
- **Structure**. The architecture pattern is described in diagrams and words in as much detail as is required to convey to the reader the components of the pattern and their responsibilities.
- **Interactions**. The important relationships and interactions among the components of the pattern are described and constraints on these relationships and interactions are identified.
- **Consequences**. The advantages and disadvantages of using this pattern are described, particularly in terms of other patterns (either required or excluded) as well as resource limitations that may arise from using it.
- **Implementation**. Additional implementation advice that can assist designers in customizing this architectural design pattern for the best results.

TADG architecture Patterns

The TADG document contains the following patterns.

Architectural Design Pattern Name	Synopsis
Client-Proxy Server	Acts as a concentrator for many low-speed links to access a server
Customer Support	Supports complex customer contact across multiple organizations
Reactor	Decouples an event from its processing
Replicated Servers	Replicates servers to reduce burden on central server
Layered Architecture	A decomposition of services such that most interactions occur only between neighboring layers
Pipe and Filter Architecture	Transforms information in a series of incremental steps or processes
Subsystem Interface	Manages the dependencies between cohesive groups of functions (subsystems)

IBM's Patterns for e-Business

The IBM Patterns for e-business web site provides a group of reusable assets aimed at speeding the process of developing e-business applications. A supporting IBM web site is Patterns for e-business Resources (also known as the "Red Books").

The rationale for IBM's provision of these patterns is to:

- Provide a simple and consistent way to translate business priorities and requirements into technical solutions
- Assist and speed up the solution development and integration process by facilitating the assembly of a solution and minimizing custom one-of-a-kind implementations.
- Capture the knowledge and best practices of experts and make it available for use by less experienced personnel.
- Facilitate the reuse of intellectual capital such as Reference Architectures, Frameworks and other architecture assets.

IBM's patterns are focused specifically on solutions for e-business, i.e., those which allow an organization to leverage web technologies in order to re-engineer business processes, enhance communications, and lower organizational boundaries with

- customers and shareholders (across the Internet);
- employees and stakeholders (across a corporate Intranet); and
- vendors, suppliers, and partners (across an Extranet).

They are intended to address the following challenges encountered in this type of environment:

- High degree of integration with legacy systems within the enterprise and with systems outside the enterprise
- The solutions need to reach users faster. This does not mean sacrificing quality, but it does mean coming up with better and faster ways to develop these solutions
- Service Level Agreements are critical

- Need to adapt to rapidly changing technologies and dramatically reduced product cycles
- Address an acute shortage of the key skills needed to develop quality solutions

IBM define five types of Pattern:

- **Business Patterns** - which identify the primary Business Actors, and describe the Interactions between them in terms of different archetypal business interactions such as:
  - Self Service (a.k.a. User-to-Business) - Users accessing transactions on a 24x7 basis
  - Collaboration (a.k.a. User-to-User) - Users working with one another to share data and information
  - Information Aggregation (a.k.a. User-to-Data) - Data from multiple sources aggregated and presented across multiple channels
  - Extended Enterprise (a.k.a. Business-to-Business) - Integrating data and processes across enterprise boundaries
- **Integration Patterns** - provide the "glue" to combine Business Patterns to form solutions. They characterize the Business Problem, Business Processes/Rules, and Existing Environment, to determine whether front-end or back-end integration is required.
  - Front-end integration - a.k.a. Access Integration - is focused on providing seamless and consistent access to business functions. Typical functions provided include Single-signon, Personalization, Transcoding etc.
  - Back-end integration - a.k.a. Application Integration - is focused on connecting, interfacing or integrating databases and systems. Typical integration can be based on Function, Type of Integration, Mode of Integration, and by Topology.
- **Composite Patterns** are previously identifed combinations and selections of Business and Integration Patterns, for previously identified situations such as: Electronic Commerce Solutions, (public) Enterprise Portals, Enterprise Intranet Portal, Collaboration ASP, etc.
- **Application Patterns**. Each Business and Integration Pattern can be implemented using one or more Application Patterns. An Application Pattern characterizes the coarse-grained structure of the application - the main application components, the allocation of processing functions and the interactions between them, the degree of integration between them, and the placement of the data relative to the applications.
- **Runtime Patterns**. Application patterns can implemented by Runtime patterns, which demonstrate non-functional, service level characteristics, such as Performance, Capacity, Scalability, and Availability. They identify key resource constraints and best practices.

The IBM web site also provides specific (IBM) product mappings for the run-time patterns, indicating specific technology choices for implementation.

Some Pattern Resources

- The Patterns Home Page hosted by The Hillside Group provides information about patterns, links to online patterns, papers and books dealing with patterns, and patterns-related mailing lists.
- The Patterns-Discussion FAQ maintained by Doug Lea provides a very thorough and highly readable FAQ about patterns.
- Patterns and Software: Essential Concepts and Terminology by Brad Appleton provides another thorough and readable account of the patterns field.

Architecture Principles

This section builds on work done by the U.S. Air Force in establishing its "Headquarters Air Force Principles for Information Management", June 29, 1998, with the addition of other input materials.

Introduction

Principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission.

In their turn, principles may be just one element in a structured set of ideas that collectively define and guide the organization, from values through to actions and results.

Depending on the organization, principles may be established at any or all of three levels:

- **Enterprise principles** provide a basis for decision making throughout an enterprise, and inform how the organization sets about fulfilling its mission. Such enterprise-level principles are commonly found in governmental and not-for-profit organizations, but are encountered in commercial organizations also, as a means of harmonizing decision making across a distributed organization. In particular, they are a key element in a successful Architecture Governance strategy.

- **Information Technology (IT) principles** provide guidance on the use and deployment of all IT resources and assets across the enterprise. They are developed in order to make the information environment as productive and cost-effective as possible.
- **Architecture principles** are a subset of IT Principles that relate to Architecture work. They reflect a level of consensus across the enterprise, and embody the spirit and thinking the enterprise architecture. Architecture principles can be further divided into:
    - **Principles that govern the architecture process**, affecting the development, maintenance, and use of the enterprise architecture; and
    - **Principles that govern the implementation of the architecture**, establishing the first tenets and related guidance for designing and developing information systems.

These sets of principles form a hierarchy, in that IT principles will be informed by, and elaborate on, the principles at the enterprise level; and architecture principles will likewise be informed by the principles at the two higher levels.

The remainder of this subsection deals exclusively with architecture principles.

Characteristics of Architecture Principles

Architecture principles define the underlying general rules and guidelines for the use and deployment of all IT resources and assets across the enterprise. They reflect a level of consensus among the various elements of the enterprise, and form the basis for making future IT decisions.

Each architecture principle should be clearly related back to the business objectives and key architecture drivers.

Components of Architecture Principles

It is useful to have a standard way of defining principles. In addition to a definition statement, each principle should have associated rationale and implications statements, both to promote understanding and acceptance of the principles themselves,

and to support the use of the principles in explaining and justifying why specific decisions are made.

A recommended template is given below.

Name	Should both represent the essence of the rule as well as be easy to remember. Specific technology platforms should not be mentioned in the name or statement of a principle. Avoid ambiguous words in the Name and in the Statement such as: "support," "open," "consider," and for lack of good measure the word "avoid," itself, be careful with "manage(ment)", and look for unnecessary adjectives and adverbs (fluff).
Statement	Should succinctly and unambiguously communicate the fundamental rule. For the most part, the principles statements for managing information are similar from one organization to the next. It is vital that the principles statement be unambiguous.
Rationale	Should highlight the business benefits of adhering to the principle, using business terminology. Point to the similarity of information and technology principles to the principles governing business operations. Also describe the relationship to other principles, and the intentions regarding a balanced interpretation. Describe situations where one principle would be given precedence or carry more weight than another for making a decision.
Implications	Should highlight the requirements, both for the business and IT, for carrying out the principle - in terms of resources, costs and activities/tasks. It will often be apparent that current systems, standards, or practices would be incongruent with the principle upon adoption. The impact to the business and consequences of adopting a principle should be clearly stated. The reader should readily discern the answer to "How does this affect me?" It is important not to oversimplify, trivialize, or judge the merit of the impact. Some of the implications will be identified as potential impacts only, and may be speculative rather than fully analyzed.

Table 1: Recommended Format for Defining Principles

An Example Set of Architecture Principles following this template is given below.

Developing Architecture Principles

Architecture principles are typically developed by the Chief Architect, in conjunction with the enterprise CIO, Architecture Board, and other key business stakeholders.

Appropriate policies and procedures must be developed to support the implementation of the principles.

Architecture principles will be informed by overall IT principles and principles at the enterprise level, if they exist. They are chosen so as to ensure alignment of IT strategies with business strategies and visions. Specifically, the development of architecture principles is typically influenced by the following:

- Enterprise Mission and Plans: The mission, plans, and organizational infrastructure of the enterprise
- Enterprise Strategic Initiatives: The characteristics of the enterprise - its strengths, weaknesses, opportunities, and threats - and its current enterprise-wide initiatives (such as Process Improvement, Quality Management)
- External Constraints: Market factors (time-to-market imperatives, customer expectations,etc.); existing and potential legislation
- Current Systems and Technology: The set of information resources deployed within the enterprise, including systems documentation, equipment inventories, network configuration diagrams, policies, and procedures
- Computer Industry Trends: Predictions about the usage, availability, and cost of computer and communication technologies, referenced from credible sources along with associated best practices presently in use.

Qualities of Principles

Merely having a written statement that is called a principle does not mean that the principle is good, even if everyone agrees with it.

A good set of principles will be founded in the beliefs and values of the organisation and expressed in language that the business understands and uses. Principles should be few in number, future oriented, and endorsed and championed by senior management. They provide a firm foundation for making architecture and planning decisions, framing policies, procedures, and standards, and supporting resolution of contradictory situations. A poor set of principles will quickly become disused, and the resultant architectures, policies, and standards will appear arbitrary or self-serving, and thus lack credibility. Essentially, principles drive behaviour.

There are five criteria that distinguish a good set of principles:

- **Understandable:** The underlying tenets can be quickly grasped and understood by individuals throughout the organization. The intention of the principle is clear and unambiguous, so that violations, whether intentional or not, are minimized.
- **Robust:** Enable good quality decisions about architectures and plans to be made, and enforceable policies and standards to be created. Each principle should be sufficiently definitive and precise to support consistent decision making in complex, potentially controversial, situations.
- **Complete:** Every potentially important principle governing the management of information and technology for the organization is defined. The principles cover every situation perceived.
- **Consistent:** Strict adherence to one principle may require a loose interpretation of another principle. The set of principles must be expressed in a way that allows a balance of interpretations. Principles should not be contradictory to the point where adhering to one principle would violate the spirit of another. Every word in a principle statement should be carefully chosen to allow consistent yet flexible interpretation.
- **Stable:** Principles should be enduring, yet able to accommodate changes. An amendment process should be established for adding, removing, or altering principles after they are ratified initially.

Applying Architecture Principles

Architecture principles are used to capture the fundamental truths about how the enterprise will use and deploy information technology resources and assets. The principles are used in a number of different ways:

1. To provide a framework within which the enterprise can start to make conscious decisions about IT
2. As a guide to establishing relevant evaluation criteria, thus exerting strong influence on the selection of products or product architectures in the later stages of managing compliance to the IT Architecture.
3. As drivers for defining the functional requirements of the architecture.
4. As an input to assessing both existing IS/IT systems and the future strategic portfolio, for compliance with the defined architectures. These assessments will provide valuable insights into the transition activities needed to implement an architecture, in support of business goals and priorities.
5. The Rationale statements (see below) highlight the value of the architecture to the enterprise, and therefore provide a basis for justifying architecture activities.
6. The Implications statements (see below) provide an outline of the key tasks, resources and potential costs to the enterprise of following the principle. They also provide valuable inputs to future transition initiative and planning activities.
7. Support the architectural governance activities in terms of:
    - Providing a 'back-stop' for the standard compliance assessments where some interpretation is allowed or required
    - Supporting the decision to initiate a dispensation request where the implications of a particular architecture amendment cannot be resolved within local operating procedure

Principles are interrelated, and need to be applied as a set.

Principles will sometimes compete: for example, the principles of "accessibility" and "security" tend towards conflicting decisions. Each principle must be considered in the context of "all other things being equal".

At times a decision will be required as to which information principle will take precedence on a particular issue. The rationale for such decisions should always be documented.

A common reaction on first reading of a principle is "this is motherhood", but the fact that a principle seems self-evident does not mean that the principle is actually observed in an organization, even when there are verbal acknowledgements of the principle.

Although specific penalties are not prescribed in a declaration of principles, violations of principles generally cause operational problems and inhibit the ability of the organization to fulfil its mission.

Example Set of Architecture Principles

Too many principles can reduce the flexibility of the architecture. Many organizations prefer to define only high level principles, and to limit the number to between 10 and 20.

The following example illustrates both the typical content of a set of architecture principles, and the recommended format for defining them, as explained above.

Another example of architecture principles is contained in the U.S. government's Federal Enterprise Architecture Framework.

Business Principles

1. Principle: **Primacy of Principles**

Statement: These principles of information management apply to all organizations within the enterprise.

Rationale: The only way we can provide a consistent and measurable level of quality information to decision makers is if all organizations abide by the principles.

Implications:

- Without this principle, exclusions, favoritism, and inconsistency would rapidly undermine the management of information.
- Information management initiatives will not begin until they are examined for compliance with the principles.
- A conflict with a principle will be resolved by changing the framework of the initiative.

2. Principle: **Maximize Benefit to the Enterprise**

Statement: Information management decisions are made to provide maximum benefit to the Enterprise as a whole.

Rationale: This principle embodies "Service above self." Decisions made from an Enterprise-wide perspective have greater long term value than decisions made from any particular organizational perspective. Maximum return on investment requires information management decisions to adhere to Enterprise-wide drivers and priorities. No minority group will detract from the benefit of the whole. However, this principle will not preclude any minority group from getting its job done.

Implications:

- Achieving maximum Enterprise-wide benefit will require changes in the way we plan and manage information. Technology alone will not bring about this change.
- Some organizations may have to concede their own preferences for the greater benefit of the entire Enterprise.
- Application development priorities must be established by the entire Enterprise for the entire Enterprise.
- Applications components should be shared across organizational boundaries.
- Information management initiatives should be conducted in accordance with the Enterprise plan. Individual organizations should pursue information management initiatives which conform to the blueprints and priorities established by the Enterprise. We will change the plan as we need to.
- As needs arise, priorities must be adjusted. A forum with comprehensive Enterprise representation should make these decisions.

3. Principle: **Information Management is Everybody's Business**

Statement: All organizations in the Enterprise participate in information management decisions needed to accomplish business objectives.

Rationale: Information users are the key stakeholders, or customers, in the application of technology to address a business need. In order to ensure information management is aligned with the business, all organizations in the Enterprise must be involved in all aspects of the information environment. The business experts from across the Enterprise and the technical staff responsible for developing and sustaining the information environment need to come together as a team to jointly define the goals and objectives of information technology.

Implications:

- To operate as a team, every stakeholder, or customer, will need to accept responsibility for developing the information environment.
- Commitment of resources will be required to implement this principle.

4. Principle: **Business Continuity**

Statement: Enterprise operations are maintained in spite of system interruptions.

Rationale: As system operations become more pervasive, we become more dependent on them, therefore, we must consider the reliability of such systems throughout their design and use. Business premises throughout the Enterprise must be provided the capability to continue their business functions regardless of external events. Hardware failure, natural disasters, and data corruption should not be allowed to disrupt or stop Enterprise activities. The Enterprise business functions must be capable of operating on alternative information delivery mechanisms.

Implications:

- Dependency on shared system applications mandates that the risks of business interruption must be established in advance and managed. Management includes but is not limited to periodic reviews, testing for vulnerability and exposure, or designing mission-critical services to assure business function continuity through redundant or alternative capabilities.
- Recoverability, redundancy and maintainability should be addressed at the time of design.
- Applications must be assessed for criticality and impact on the Enterprise mission, in order to determine what level of continuity is required and what corresponding recovery plan is necessary.

5. Principle: **Common Use Applications**

Statement: Development of applications used across the Enterprise is preferred over the development of similar or duplicative applications which are only provided to a particular organization.

Rationale: Duplicative capability is expensive and proliferates conflicting data.

Implications:

- Organizations which depend on a capability which does not serve the entire Enterprise must change over to the replacement Enterprise-wide capability. This will require establishment of and adherence to a policy requiring this.
- Organizations will not be allowed to develop capabilities for their own use which are similar/duplicative of Enterprise-wide capabilities. In this way, expenditures of scarce resources to develop essentially the same capability in marginally different ways will be reduced.
- Data and information used to support Enterprise decision making will be standardized to a much greater extent than previously. This is because the smaller, organizational capabilities which produced different data (which was not shared among other organizations) will be replaced by Enterprise-wide capabilities. The impetus for adding to the set of Enterprise-wide capabilities may well come from an organization making a convincing case for the value of the data/information previously produced by its organizational capability, but the resulting capability will become part of the Enterprise-wide system, and the data it produces will be shared across the Enterprise.

6. Principle: **Compliance with Law**

Statement: Enterprise information management processes comply with all relevant laws, policies, and regulations

Rationale: Enterprise policy is to abide by laws, policies, and regulations. This will not preclude business process improvements that lead to changes in policies and regulations.

Implications:

- The Enterprise must be mindful to comply with laws, regulations, and external policies regarding the collection, retention, and management of data.
- Education and access to the rules. Efficiency, need and common sense are not the only drivers. Changes in the law and changes in regulations may drive changes in our processes or applications.

7. Principle: **IT Responsibility**

Statement: The IT organization is responsible for owning and implementing IT processes and infrastructure that enable solutions to meet user defined requirements for functionality, service levels, cost, and delivery timing.

Rationale: Effectively align expectations with capabilities and costs so that all projects are cost effective. Efficient and effective solutions have reasonable costs and clear benefits.

Implications:

- A process must be created to prioritize projects
- The IT function must define processes to manage business unit expectations
- Data, application, and technology models must be created to enable integrated quality solutions and to maximize results.

8. Principle: **Protection of Intellectual Property**

Statement: The enterprise's IP must be protected. This protection must be reflected in the IT Architecture, Implementation, and Governance processes.

Rationale: A major part of an enterprise's Intellectual Property is hosted in the IT domain.

Implications:

- While protection of IP assets is everybody's business, much of the actual protection is implemented in the IT domain. Even trust in non-IT processes can be managed by IT processes (email, mandatory notes, etc.).
- A Security Policy, governing human and IT actors, will be required that can substantially improve protection of IP. This must be capable of both avoiding compromises and reducing liabilities.
- Resources on such policies can be found at the SANS Institute.

Data Principles

9. Principle: **Data is an Asset**

Statement: Data is an asset that has value to the Enterprise and is managed accordingly.

Rationale: Data is a valuable corporate resource; it has real, measurable value. In simple terms, the purpose of data is to aid decision making. Accurate, timely data is critical to accurate, timely decisions. Most corporate assets are carefully managed, and data is no exception. Data is the foundation of our decision making, so we must also carefully manage data to assure that we know where it is, can rely upon its accuracy, and can obtain it when and where we need it.

Implications:

- This is one of three closely related principles regarding data: **data is an asset; data is shared;** and **data is easily accessible**. The implication is that there is an education task to ensure that all organizations within the Enterprise understand the relationship between value of data, sharing of data, and accessibility to data.
- Stewards must have the authority and means to manage the data for which they are accountable.
- We must make the cultural transition from "data-ownership" thinking to "data-stewardship" thinking.
- The role of data steward is critical because obsolete, incorrect, or inconsistent data could be passed to Enterprise personnel and adversely affect decisions across the Enterprise.
- Part of the role of data steward, who manages the data, is to ensure data quality. Procedures must be developed and used to prevent and correct errors in the information and to improve those processes that produce flawed information. Data quality will need to be measured and steps taken to improve data quality -- it is probable that policy and procedures will need to be developed for this as well.

- A forum with comprehensive Enterprise-wide representation should decide on process changes suggested by the steward.
- Since data is an asset of value to the entire Enterprise, data stewards accountable for properly managing the data must be assigned at the Enterprise level.

10. Principle: **Data is Shared**

Statement: Users have access to the data necessary to perform their duties; therefore, data is shared across Enterprise functions and organizations.

Rationale: Timely access to accurate data is essential to improving the quality and efficiency of Enterprise decision making. It is less costly to maintain timely, accurate data in a single application, and then share it, than it is to maintain duplicative data in multiple applications. The Enterprise holds a wealth of data, but it is stored in hundreds of incompatible stovepipe databases. The speed of data collection, creation, transfer, and assimilation is driven by the ability of the organization to efficiently share these islands of data across the organization.

Shared data will result in improved decisions since we will rely on fewer (ultimately one virtual) sources of more accurate and timely managed data for all of our decision making. Electronically shared data will result in increased efficiency when existing data entities can be used, without re-keying, to create new entities.

Implications:

- This is one of three closely related principles regarding data: **data is an asset; data is shared;** and **data is easily accessible**. The implication is that there is an education task to ensure that all organizations within the Enterprise understand the relationship between value of data, sharing of data, and accessibility to data.
- To enable data sharing we must develop and abide by a common set of policies, procedures and standards governing data management and access for both the short and the long term.
- For the short term, to preserve our significant investment in legacy systems, we must invest in software capable of migrating legacy system data into a shared data environment.
- We will also need to develop standard data models, data elements, and other metadata that defines this shared environment and develop a repository system for storing this metadata to make it accessible.
- For the long term, as legacy systems are replaced, we must adopt and enforce common data access policies and guidelines for new application developers to ensure that data in new applications remains available to the shared environment and that data in the shared environment can continue to be used by the new applications.
- For both the short tern and the long term we must adopt common methods and tools for creating, maintaining and accessing the data shared across the Enterprise.
- Data sharing will require a significant cultural change.
- This principle of data sharing will continually "bump up against" the principle of data security. Under no circumstances will the data sharing principle cause confidential data to be compromised.
- Data made available for sharing will have to be relied upon by all users to execute their respective tasks. This will ensure that only the most accurate and timely data is relied upon for decision making. Shared data will become the Enterprise-wide "virtual single source" of data.

11. Principle: **Data is Accessible**

Statement: Data is accessible for users to perform their functions

Rationale: Wide access to data leads to efficiency and effectiveness in decision-making, and affords timely response to information requests and service delivery. Using information must be considered from an Enterprise perspective to allow access by a wide variety of users. Staff time is saved and consistency of data are improved.

Implications:

- This is one of three closely related principles regarding data: **data is an asset; data is shared;** and **data is easily accessible**. The implication is that there is an education task to ensure that all organizations within the Enterprise understand the relationship between value of data, sharing of data, and accessibility to data.
- Accessibility involves the ease with which users obtain information.
- The way information is accessed and displayed must be sufficiently adaptable to meet a wide range of Enterprise users and their corresponding methods of access.
- Access to data does not constitute understanding of the data. Personnel should take caution not to misinterpret information.
- Access to data does not necessarily grant the user access rights to modify or disclose the data. This will require an education process and a change in the organizational culture, which currently supports a belief in "ownership" of data by functional units.

12. Principle: **Data Trustee**

Statement: Each data element has a trustee accountable for data quality.

Rationale: One of the benefits of an architected environment is the ability to share data (e.g., text, video, sound, etc.) across the Enterprise. As the degree of data sharing grows and business units rely upon common information, it becomes essential that only the data trustee make decisions about the content of data. Since data can lose its integrity when it is entered multiple times, the data trustee will have sole responsibility for data entry which eliminates redundant human effort and data storage resources. (Note that a trustee is different than steward - trustee is responsible for accuracy and currency of the data while responsibilities of a steward may be broader and include data standardization and definition tasks.)

Implications:

- Real trusteeship dissolves the data "ownership" issues and allows the data to be available to meet all users' needs. This implies that a cultural change from data "ownership" to data "trusteeship" may be required.
- The data trustee will be responsible for meeting quality requirements levied upon the data for which the trustee is accountable.
- It is essential that the trustee has the ability to provide user confidence in the data based upon attributes such as 'data source.'
- It is essential to identify the true source of the data in order that the data authority can be assigned this trustee responsibility. This does not mean that classified sources will be revealed nor does it mean the source will be the trustee.
- Information should be captured electronically once and immediately validated as close to the source as possible. Quality control measures must be implemented to ensure the integrity of the data.
- As a result of sharing data across the Enterprise, the trustee is accountable and responsible for the accuracy and currency of their designated data element(s) and subsequently, must then recognize the importance of this trusteeship responsibility.

13. Principle: **Common Vocabulary and Data Definitions**

Statement: Data is defined consistently throughout the Enterprise, and the definitions are understandable and available to all users.

Rationale: The data that will be used in the development of applications must have a common definition throughout the Headquarters to enable sharing of data. A common vocabulary will facilitate communications and enable dialogue to be effective. In addition, it is required to interface systems and exchange data.

Implications:

- We are lulled into thinking that this issue is adequately addressed because there are people with "data administration" job titles and forums with charters implying responsibility. Significant additional energy and resources must be committed to this task. It is a key to the success of efforts to improve the information environment. This is separate from but related to the issue of data element definition, which is addressed by a broad community - this is more like a common vocabulary and definition.
- The Enterprise must establish the initial common vocabulary for the business. The definitions will be used uniformly throughout the Enterprise.
- Whenever a new data definition is required, the definition effort will be co-ordinated and reconciled with the corporate "glossary" of data descriptions. The Enterprise Data Administrator will provide this co-ordination.
- Ambiguities resulting from multiple parochial definitions of data must give way to accepted Enterprise wide definitions and understanding.
- Multiple data standardization initiatives need to be co-ordinated.
- Functional data administration responsibilities must be assigned.

14. Principle: **Data Security**

Statement: Data is protected from unauthorized use and disclosure. In addition to the traditional aspects of national security classification, this includes, but is not limited to, protection of pre-decisional, sensitive, source selection sensitive, and proprietary information.

Rationale: Open sharing of information and the release of information via relevant legislation must be balanced against the need to restrict the availability of classified, proprietary, and sensitive information.

Existing laws and regulations require the safeguarding of national security and the privacy of data, while permitting free and

open access. Pre-decisional (work-in-progress, not yet authorized for release) information must be protected to avoid unwarranted speculation, misinterpretation, and inappropriate use.

Implications:

- Aggregation of data, both classified and not, will create a large target requiring review and declassification procedures to maintain appropriate control. Data owners and/or functional users must determine if the aggregation results in an increased classification level. We will need appropriate policy and procedures to handle this review and declassification. Access to information based on a need-to-know policy will force regular reviews of the body of information.
- The current practice of having separate systems to contain different classifications needs to be rethought. Is there a software solution to separating classified and unclassified data? The current hardware solution is unwieldy, inefficient, and costly. It is more expensive to manage unclassified data on a classified system. Currently, the only way to combine the two is to place the unclassified data on the classified system, where it must remain.
- In order to adequately provide access to open information while maintaining secure information, security needs must be identified and developed at the data level, not the application level.
- Data security safeguards can be put in place to restrict access to "view only", or "never see". Sensitivity labeling for access to pre-decisional, decisional, classified, sensitive, or proprietary information must be determined.
- Security must be designed into data elements from the beginning; it cannot be added later. Systems, data, and technologies must be protected from unauthorized access and manipulation. Headquarters information must be safeguarded against inadvertent or unauthorized alteration, sabotage, disaster, or disclosure.
- Need new policies on managing duration of protection for pre-decisional information and other works-in-progress-- in consideration of content freshness.

Application Principles

15. Principle: **Technology Independence**

Statement: Applications are independent of specific technology choices and therefore can operate on a variety of technology platforms.

Rationale: Independence of applications from the underlying technology allows applications to be developed, upgraded, and operated in the most cost-effective and timely way. Otherwise technology, which is subject to continual obsolescence and vendor dependence, becomes the driver rather than the user requirements themselves.
Realizing that every decision made respect to information technology makes us dependent on that technology, the intent of this principle is to ensure that applications software is not dependent on specific hardware and operating systems software.

Implications:

- This principle will require standards which support portability.
- For COTS and GOTS applications, there may be limited current choices, as many of these applications are technology and platform dependent.
- Application Programming Interfaces (APIs) will need to be developed to enable legacy applications to interoperate with applications and operating environments developed under the Enterprise architecture.
- Middleware should be used to decouple applications from specific software solutions.
- As an example, this principle could lead to use of JAVA, and future JAVA-like protocols, which give a high degree of priority to platform independence.

16. Principle: **Ease of Use**

Statement: Applications are easy to use. The underlying technology is transparent to users, so they can concentrate on tasks at hand.

Rationale: The more a user has to understand the underlying technology the less productive that user is. Ease of use is a positive incentive for use of applications. It encourages users to work within the integrated information environment instead of developing isolated systems to accomplish the task outside of the Enterprise's integrated information environment. Most of the knowledge required to operate one system will be similar to others. Training is kept to a minimum, and the risk of using a system improperly is low.

Using an application should be as intuitive as driving a different car.

Implications:

- Applications will be required to have a common "look and feel" and support ergonomic requirements. Hence, the common look and feel standard must be designed and usability test criteria must be developed.
- Guidelines for user interfaces should not be constrained by narrow assumptions about user location, language, systems training, or physical capability. Factors such as linguistics, customer physical infirmities (visual acuity, ability to use keyboard/mouse) and proficiency in the use of technology have broad ramifications in determining the ease of use of an application.

Technical Principles

17. Principle: **Requirements-Based Change**

Statement: Only in response to business needs are changes to applications and technology made.

Rationale: This principle will foster an atmosphere where the information environment changes in response to the needs of the business, rather than having the business change in response to information technology changes. This is to ensure that the purpose of the information support -- the transaction of business -- is the basis for any proposed change. Unintended effects on business due to information technology changes will be minimized. A change in technology may provide an opportunity to improve the business process and hence, change business needs.

Implications:

- Changes in implementation will follow full examination of the proposed changes using the Enterprise architecture.
- We don't fund a technical improvement or system development unless a documented business need exists.
- Change management processes conforming to this principle will be developed and implemented.
- This principle may bump up against the responsive change principle. We must ensure the requirements documentation process does not hinder responsive change to meet legitimate business needs. Purpose of this principle is to keep us focused on business, not technology, needs--responsive change is also a business need.

18. Principle: **Responsive Change Management**

Statement: Changes to the Enterprise information environment are implemented in a timely manner.

Rationale: If people are to be expected to work within the Enterprise information environment, that information environment must be responsive to their needs.

Implications:

- We have to develop processes for managing and implementing change that do not create delays.
- A user who feels a need for change will need to connect with a "business expert" to facilitate explanation and implementation of that need.
- If we are going to make changes, we must keep the architectures updated.
- Adopting this principle might require additional resources.
- This will conflict with other principles (e.g., Maximum Enterprise-wide benefit, Enterprise-wide Applications, etc.).

19. Principle: **Control Technical Diversity**

Statement: Technological diversity is controlled to minimize the non-trivial cost of maintaining expertise in and connectivity between multiple processing environments.

Rationale: There is a real, non-trivial cost of infrastructure required to support alternative technologies for processing environments. There are further infrastructure costs incurred to keep multiple processor constructs interconnected and maintained.

Limiting the number of supported components will simplify maintainability and reduce costs.

The business advantages of minimum technical diversity include: standard packaging of components; predictable implementation impact; predictable valuations and returns; redefined testing; utility status; and increased flexibility to

accommodate technological advancements. Common technology across the Enterprise brings the benefits of economies of scale to the Enterprise. Technical administration and support costs are better controlled when limited resources can focus on this shared set of technology.

Implications:

- Policies, standards, and procedures that govern acquisition of technology must be tied directly to this principle.
- Technology choices will be constrained by the choices available within the technology blueprint. Procedures for augmenting the acceptable technology set to meet evolving requirements will have to be developed and emplaced.
- We are not freezing our technology baseline. We welcome technology advances and will change the technology blueprint when compatibility with the current infrastructure, improvement in operational efficiency, or a required capability has been demonstrated.

20. Principle: **Interoperability**

Statement: Software and hardware should conform to defined standards that promote interoperability for data, applications and technology.

Rationale: Standards help ensure consistency, thus improving the ability to manage systems and improve user satisfaction, and protect existing IT investments, thus maximizing return on investment and reducing costs. Standards for interoperability additionally help ensure support from multiple vendors for their products, and facilitate supply chain integration.

Implications:

- Interoperability standards and industry standards will be followed unless there is a compelling business reason to implement a non-standard solution.
- A process for setting standards, reviewing and revising them periodically, and granting exceptions must be established.
- The existing IT platforms must be identified and documented.

TOGAF Architecture Skills Framework

Introduction to Skills Frameworks

Skills Frameworks provide a view of the competency levels required for specific roles. They define:

- the roles within a work area
- the skills required by each role
- the depth of knowledge required to fulfil the role successfully

They are relatively common for defining the skills required for a consultancy and/or project management assignment, to deliver a specific project or work package. They are also widely used by recruitment and search agencies to match candidates and roles.

Their value derives from their ability to provide a means of rapidly identifying skill matches and gaps. Successfully applied, they can ensure that candidates are fit for the jobs assigned to them.

Their value in the context of Enterprise Architecture arises from the immaturity of the Enterprise Architecture discipline, and the problem that arise from this.

The Need for An IT Architecture Skills Framework

Definitional Rigor

"IT Architecture" and "IT Architect" are widely used but poorly defined terms in the IT industry today. They are used to denote a variety of practices and skills applied in a wide variety of IT domains. There is a need for better classification to enable more implicit understanding of what type of Architect / Architecture is being described.

This lack of uniformity leads to difficulties for organizations seeking to recruit or assign/promote staff to fill positions in the Architecture field. Because of the different usages of terms, there are often misunderstanding and miscommunication between those seeking to recruit for, and those seeking to fill, the various roles of Architect.

The Basis of an Internal Architecture Practice

Despite the lack of uniform terminology, Architecture skills are in increasing demand, as the discipline of Architecture gains increasing attention within the IT industry.

Many enterprises have set up, or are considering setting up, an IT Architecture practice, as a means of fostering development of the necessary skills and experience among in-house staff to undertake the various architecting tasks required by the enterprise.

An IT Architecture practice is a formal program of development and certification, by which an enterprise formally recognizes the skills of its practising IT Architects, as demonstrated by their work. Such a program is essential in order to ensure the alignment of staff skills and experience with the IT Architecture tasks that the enterprise wishes to be performed.

The role and skill definitions on which such a program needs to be based are also required, by both recruiting and supplying organizations, in cases where external personnel are to be engaged to perform architecture work (for example, as part of a consultancy engagement).

An IT Architecture practice is both difficult and costly to set up. It is normally built around a process of peer review, and involves the time and talent of the strategic technical leadership of an enterprise. Typically it involves establishment of a peer review board, and documentation of the process,and of the requirements for internal certification. Time is also required of candidates to prepare for peer review, by creating a portfolio of their work to demonstrate their skills, experiences, and contributions to the profession.

The TOGAF Enterprise Architecture Skills Framework attempts to address this need by providing definitions of the Architecting skills and proficiency levels required of personnel, internal or external, who are to perform the various architecting roles defined within the TOGAF framework.

Because of the complexity, time and cost involved, many enterprises do not have an internal IT Architect certification program, preferring instead to simply interview and recruit architecture staff on an ad hoc basis. There are serious risks associated with this ad hoc approach:

- Communication between recruiting organizations, consultancies and employment agencies is very difficult.
- Time is wasted interviewing staff who may have applied in all good faith, but still lack the skills and/or experience required by the employer.
- Staff who are capable of filling architecture roles may be overlooked, or may not identify themselves with advertised positions and hence not even apply.
- There is increased risk of unsuitable personnel being employed or engaged, through no-one's fault, and despite everyone involved acting in good faith. This in turn can:
    - Increase personnel costs, through the need to rehire or reassign staff.
    - Adversely impact the time, cost and quality of operational IT systems, and the projects that deliver them.

Goals / Rationale

Enterprise Certification of IT Architects

The main purpose behind an enterprise setting up an internal IT Architect certification program are twofold:

- to formally recognize the skill of its practising IT Architects, as part of the task of establishing and maintaining a professional IT Architecting organization; and
- to ensure the alignment of necessary staff skills and experience with the IT Architecture tasks that the enterprise wishes to be performed, whether these are to be performed internally to the enterprise or externally - for example, as part of a consultancy engagement.

Specific Benefits

Specific benefits anticipated from use of the TOGAF Skills Framework include:

- Reduces the time, cost, and risk in training, hiring, and managing IT Architecture professionals, both internal and external.
    - Simplifies communication between recruiting organizations, consultancies and employment agencies.
    - Avoids wasting time interviewing staff who may have applied in all good faith, but still lack the skills and/or experience required by the employer.
    - Avoids staff who are capable of filling IT architecture roles being overlooked, or not identifying themselves with advertised positions and hence not even applying.
- Reduces the time and cost to set up an internal IT Architecture practice.
    - Many enterprises do not have an internal IT Architecture practice due to the complexity involved in setting one up, preferring instead to simply interview and recruit architecture staff on an ad hoc basis.
    - By providing definitions of the Architecting skills and proficiency levels required of personnel who are to perform the various architecting roles defined within TOGAF, the TOGAF Architecture Skills Framework greatly reduces the time, cost and risk of setting up a practice from scratch, and avoids "re-inventing wheels".
    - Enterprises that already have an internal IT Architecture practice are able to set enterprise-wide norms, but still experience difficulties as outlined above in recruiting staff, or engaging consultants, from external sources, due to the lack of uniformity between different enterprises. By aligning its existing skills framework with the industry-accepted definitions provided by The Open Group, an enterprise can greatly simplify these problems.
- By reducing the time and cost to implement an IT Architect Practice, helps reduce the time, cost and risk of overall IT development.
    - Enterprises do not have an internal IT Architecture practice run the risk of unsuitable personnel being employed

or engaged, through no-one's fault, and despite everyone involved acting in good faith. The resultant time and cost penalties far outweigh the time and cost of having an internal IT Architecture practice
    - Personnel costs are increased, through the occasional need to rehire or reassign staff
    - Even more important is the adverse impact on the time, cost and quality of operational IT systems, and the projects to deliver them, resulting from poor staff assignments.

IT Architecture Role and Skill Categories

Overview

This section describes the role of an IT Architect, the fundamental skills required, and some possible disciplines in which an IT Architect might specialize.

TOGAF Version 8 delivers an Enterprise Architecture, and therefore requires both Business and IT trained professionals to develop the Enterprise Architecture.

The TOGAF Skills Framework provides a view of the competency levels for specific roles within the Enterprise Architecture team. The Framework defines:

- The roles within an enterprise architecture work area
- The skills required by those roles
- The depth of knowledge required to fulfil each role successfully

The value is in providing a rapid means of identifying skills and gaps. Successfully applied, the Framework can be used as a measure for:

- staff development
- ensuring that the right person does the right job

TOGAF Roles

A typical IT Architecture team undertaking the development of an Enterprise Architecture as described in TOGAF would comprise the following roles:

- Architecture Board Members
- Architecture Sponsor
- IT Architecture Manager
- IT Architects for:
    - Enterprise Architecture
    - Business Architecture
    - Data Architecture
    - Applications Architecture
    - Technology Architecture
- Programme and/or Project Managers
- IT Designer
- And many others…..

The tables that follow show, for each of these roles, the skills required and the desirable level of proficiency in each skill.

Of all the roles listed above, the one that needs particularly detailed analysis and definition is of course the central role of IT Architect. As explained above, "IT Architecture" and "IT Architect" are terms that are very widely used but very poorly defined in the IT industry today, denoting a wide variety of practices and skills applied in a wide variety of IT domains. There is often confusion between the role of an IT Architect and that of an IT Designer or IT Builder. Many of the skills required by an IT Architect are also required by the IT Designer, who delivers the solutions. While their skills are complimentary, those of the IT Designer are primarily technology focussed and translate the architecture into deliverable components.

The final subsection below therefore explores in some detail the generic characteristics of the role of IT Architect, and the key skill requirements, whatever the particular IT domain (Enterprise Architecture, Business Architecture, Data Architecture, Applications Architecture, Technology Architecture,....).

Categories of Skills

The TOGAF Team skill set will need to include the following main categories of skills:

- **Generic Skills**; typically comprising:
  - Leadership, team working, inter-personal skills, etc.
- **Business Skills and Methods**; typically comprising:
  - Business cases, business process, strategic planning, etc.
- **Enterprise Architecture Skills**; typically comprising:
  - Modelling, building block design, applications and role design, systems integration
- **Program or Project Management Skills** ; typically comprising:
  - Managing business change, project management methods and tools.
- **IT General Knowledge Skills**; typically comprising:
  - Brokering applications, asset management, migration planning, SLAs
- **Technical IT Skills**; typically comprising:
  - Software engineering, security, data interchange, data management (TRM)AS
- **Legal Environment**; typically comprising:
  - Data protection laws, contract law, procurement law, fraud

The tables that follow illustrate each of these categories of skills.

The tables that follow show, for each of these skills, the roles to which they are relevant and the desirable level of proficiency in each skill.

Proficiency Levels

The TOGAF Skills Framework identifies four levels of knowledge or proficiency in any area:

Level	Achievement	Description
1	Background	Not a required skill though should be able to define and manage skill if required
2	Awareness	Understands the background, issues and implications sufficiently to be able to understand how to proceed further and advise client accordingly.
3	Knowledge	Detailed knowledge of subject area and capable of providing professional advice and guidance. Ability to integrate capability into architecture design
4	Expert	Extensive and substantial practical experience and applied knowledge on the subject.

IT Architecture Role and Skill Definitions

General Skills

IT Architect Roles	Architecture Board Member	Architecture Sponsor	IT Architect Manager	IT Architect Technology	IT Architect Data	IT Architect Application	IT Architect Business	Programme or Project Manager	IT Designer
Framework Skills Areas									
Generic Skills									
Leadership	4	4	4	3	3	3	3	4	1
Team Work	3	3	4	4	4	4	4	4	2
Inter-personal skills	4	4	4	4	4	4	4	4	2
Oral Communications	3	3	4	4	4	4	4	4	2
Written Communications	3	3	4	4	4	4	4	3	3
Logical Analysis	2	2	4	4	4	4	4	3	3
Stakeholder Management	4	3	4	3	3	3	3	4	2
Risk Management	3	3	4	3	3	3	3	4	1

Business Architecture Skills & Methods

IT Architect Roles	Architecture Board Member	Architecture Sponsor	IT Architect Manager	IT Architect Technology	IT Architect Data	IT Architect Application	IT Architect Business	Programme or Project Manager	IT Designer
Business Skills & Methods									
Business Case	3	4	4	4	4	4	4	4	2
Business Scenario	2	3	4	4	4	4	4	3	2
Organisation	3	3	4	3	3	3	4	3	2
Business Process	3	3	4	4	4	4	4	3	2
Strategic Planning	2	3	3	3	3	3	4	3	1
Budget Management	3	3	3	3	3	3	3	4	3
Visioning	3	3	4	3	3	3	4	3	2
Business Metrics	3	4	4	4	4	4	4	4	3
Business Culture	4	4	4	3	3	3	3	3	1
Legacy Investments	4	4	3	2	2	2	2	3	2
Business Functions	3	3	3	3	4	4	4	3	2

Enterprise Architecture Skills & Methods

IT Architect Roles	Architecture Board Member	Architecture Sponsor	IT Architect Manager	IT Architect Technology	IT Architect Data	IT Architect Application	IT Architect Business	Programme or Project Manager	IT Designer
Enterprise Architecture Skills									
Business Modelling	2	2	4	3	3	4	4	2	2
Business Process design	1	1	4	3	3	4	4	2	2
Role design	2	2	4	3	3	4	4	2	2
Organization Design	2	2	4	3	3	4	4	2	2
Data Design	1	1	3	3	4	3	3	2	3
Application Design	1	1	3	3	3	4	3	2	3
Systems Integration	1	1	4	4	3	3	3	2	2
IT Industry Standards	1	1	4	4	4	4	3	2	3
Services Design	2	2	4	4	3	4	3	2	2
Architecture Principles design	2	2	4	4	4	4	4	2	2
Architecture Views & Viewpoints design	2	2	4	4	4	4	4	2	2
Building Block Design	1	1	4	4	4	4	4	2	3
Solutions Modelling	1	1	4	4	4	4	4	2	3
Benefits Analysis	2	2	4	4	4	4	4	4	2
Business Inter-working	3	3	4	3	3	4	4	3	1
Systems Behaviour	1	1	4	4	4	4	3	3	2
Project Management	1	1	3	3	3	3	3	4	2

Program or Project Management Skills & Methods

IT Architect Roles	Architecture Board Member	Architecture Sponsor	IT Architect Manager	IT Architect Technology	IT Architect Data	IT Architect Application	IT Architect Business	Programme or Project Manager	IT Designer
Programme or Project Management									
Programme Management	1	2	3	3	3	3	3	4	2
Project Management	1	2	3	3	3	3	3	4	2
Managing Business Change	3	3	4	3	3	3	4	4	2
Change Management	3	3	4	3	3	3	4	3	2
Value Management	4	4	4	3	3	3	4	3	2

IT Knowledge Skills & Methods

IT Architect Roles	Architecture Board Member	Architecture Sponsor	IT Architect Manager	IT Architect Technology	IT Architect Data	IT Architect Application	IT Architect Business	Programme or Project Manager	IT Designer
IT Knowledge Skills									
IT Application Development Methodologies & Tools	2	2	3	4	4	4	2	3	3
Programming Languages	1	1	3	4	4	4	2	2	3
Brokering Applications	1	1	3	2	4	4	3	2	3
Information Consumer Applications	1	1	3	2	4	4	3	2	3
Information Provider Applications	1	1	3	2	4	4	3	2	3
Storage Management	1	1	3	4	4	2	2	2	3
Networks	1	1	3	4	3	2	2	2	3
Web based Services	1	1	3	3	4	4	2	2	3
IT Infrastructure	1	1	3	4	3	2	2	2	3
Asset Management	1	1	4	4	3	3	3	2	3
Service Level Agreements	1	1	4	4	3	4	3	2	3
Systems	1	1	3	4	3	3	2	2	3
COTS	1	1	3	4	3	4	2	2	3
Enterprise continuums	1	1	4	4	4	4	4	2	3
Migration Planning	1	1	4	3	4	3	3	2	3
Management Utilities	1	1	3	2	4	4	2	2	3
Infrastructure Applications	1	1	3	4	3	4	2	2	3

IT Knowledge Skills & Methods

IT Architect Roles	Architecture Board Member	Architecture Sponsor	IT Architect Manager	IT Architect Technology	IT Architect Data	IT Architect Application	IT Architect Business	Programme or Project Manager	IT Designer
Technical IT Skills									
Software Engineering	1	1	3	3	4	4	3	2	3
Security	1	1	3	4	3	4	3	2	3
Systems & Network Management	1	1	3	4	3	3	3	2	3
Transaction Processing	1	1	3	4	3	4	3	2	3
Location & Directory	1	1	3	4	4	3	3	2	3
User Interface	1	1	3	4	4	4	3	2	3
International Operations	1	1	3	4	3	3	2	2	2
Data Interchange	1	1	3	4	4	3	2	2	3
Data Management	1	1	3	4	4	3	2	2	3
Graphics and Image	1	1	3	4	3	3	2	2	3
Operating Systems Services	1	1	3	4	3	3	2	2	3
Network Services	1	1	3	4	3	3	2	2	3
Communications Infrastructure	1	1	3	4	3	3	2	2	3

Legal Environment

IT Architect Roles	Architecture Board Member	Architecture Sponsor	IT Architect Manager	IT Architect Technology	IT Architect Data	IT Architect Application	IT Architect Business	Programme or Project Manager	IT Designer
Legal Environment									
Contract Law	2	2	2	2	2	2	2	3	1
Data Protection Laws	3	3	4	3	4	4	3	2	2
Procurement Law	3	2	2	2	2	2	2	4	1
Fraud	3	3	3	3	3	3	3	3	1
Commercial Law	3	3	2	2	2	2	3	3	1

The Generic Role and Skills of the IT Architect

Of all the roles listed above, the one that needs particularly detailed analysis and definition is of course the central role of IT Architect. As explained above, "IT Architecture" and "IT Architect" are terms that are very widely used but very poorly defined in the IT industry today, denoting a wide variety of practices and skills applied in a wide variety of IT domains.

This subsection therefore explores in some detail the generic characteristics of the role of IT Architect, and some key skill requirements, whatever the particular IT domain (Enterprise Architecture, Business Architecture, Data Architecture, Applications Architecture, Technology Architecture,....).

Generic Role

IT Architects are visionaries, coaches, team leaders, business-to-technical liaisons, computer scientists, and industry experts.

The following quote from Part I of TOGAF is effectively a job description for an IT Architect:

> "The architect has a responsibility for ensuring the completeness (fitness-for-purpose) of the architecture, in terms of adequately addressing all the pertinent concerns of its stakeholders; and the integrity of the architecture, in terms of connecting all the various views to each other, satisfactorily reconciling the conflicting concerns of different stakeholders, and showing the trade-offs made in so doing (as between security and

performance, for example).

The choice of which particular architecture views to develop is one of the key decisions that the IT Architect has to make. The choice has to be constrained by considerations of practicality, and by the principal of fitness-for-purpose (i.e., the architecture should be developed only to the point at which it is fit for purpose, and not reiterated ad infinitum as an academic exercise)."

The role of the IT Architect is more like that of a city planner than that of a building architect, and the product of the IT Architect is more aptly characterized as a planned community (as opposed to an unconstrained urban sprawl), rather than as a well-designed building or set of buildings.

An IT Architect does not create the technical vision of the enterprise, but has professional relationships with executives of the enterprise to gather and articulate the technical vision, and to produce the strategic plan for realizing it. This plan is always tied to the business plans of the enterprise, and design decisions are traceable to the business plan.

The strategic plan of the IT Architect is tied to the governance process for the enterprise, so design decisions are not circumvented for tactical convenience.

The IT Architect produces documentation of design decisions for application development teams or product implementation teams to execute.

An architect is involved in the entire process, beginning with working with the customer to understand real needs, as opposed to wants, and then throughout the process to translate those needs into capabilities verified to meet the needs. Additionally the architect may present different models to the customer that communicate how those needs may be met, and is therefore an essential participant in the consultative selling process.

However the architect is not the builder, and must remain at a level of abstraction necessary to ensure s/he does not get in the way of practical implementation.

The following excerpt from The Art of Systems Architecting by Eberhardt Rechtin and Mark W. Maier depicts this notion: "It is the responsibility of the architect to know and concentrate on the critical few details and interfaces that really matter and not to become overloaded with the rest."

The architect's focus is on understanding what it takes to satisfy the client, where qualitative worth is used more than quantitative measures. The architect uses more inductive skills than the deductive skills of the builder. The architect deals more with guidelines, rather than rules that builders use as a necessity.

It also must be clear that the role of an architect may be performed by an engineer. A goal of this document is to describe the role what should be done, regardless of who is performing it.

Thus the role of the architect can be summarized as to:

- **Understand and interpret requirements** - Probe for information, listen to information, influence people, facilitate consensus building, synthesize and translate ideas into actionable requirements, articulate those ideas to others. Identify use or purpose, constraints, risks, etc. The architect participates in the discovery and documentation of the customer's business scenarios that are driving the solution. The architect is responsible for requirements understanding and embodies that requirements understanding in the architecture specification.
- **Create a useful model** - Take the requirements and develop well formulated models of the components of the solution, augmenting the models as necessary to fit all of the circumstances. Show multiple views through models to communicate the ideas effectively. The architect is responsible for the overall architecture integrity and maintaining the vision of the offering from an architectural perspective. The architect also ensures leverage opportunities are identified, using building blocks, and is a liaison between the functional groups (especially development and marketing) to ensure that the leverage opportunities are realized. The architect provides and maintains these models as a framework for understanding the domain(s) of development work, guiding what should be done within the organization, or outside the organization. The architect must represent the organization view of the architecture by understanding all the necessary business components.
- **Validate, refine and expand the model** - Verify assumptions, bring in Subject Matter Experts, etc. in order to improve the model and to further define it, adding as necessary new ideas to make the result more flexible and more tightly linked to current and expected requirements. The architects additionally should assess the value of solution-enhancing developments emanating from field work and incorporate these into the architecture models as appropriate.
- **Manage the architecture** - Continuously monitor the models and update them as necessary to show changes, additions and alterations. Represent architecture and issues during development and decision points of the program. The architect is an "agent of change," representing that need for the implementation of the architecture. Through this

development cycle the architect continuously fosters the sharing of customer, architecture and technical information between organizations.

Characterization in Terms of the Enterprise Continuum

Under certain circumstances, the complexity of a solution may require additional architects to support the architecture effort. The different categories of architects are described below, but as they are architects, they all perform the tasks described above. Any combination of Foundation, Systems, Solutions, and Customer Architects may be utilized, as a team. In such cases each member may have a specific focus, if not specific roles and responsibilities, within the phases of the development process. In cases where a team of architects is deemed necessary, a lead architect should be assigned to manage and lead the team members.

- The **Foundation Architect** has the responsibility for architectural design and documentation at a technical reference model level. The Foundation Architect often leads a group of the System and/or Industry Architects related to a given program. The focus of the Foundation Architect is on enterprise level business functions required.
- The **System Architect** has the responsibility for architectural design and documentation at a system or sub-system level, such as management or security. A system architect may shield the Foundations Architect from the unnecessary details of the systems, products and/or technologies. The focus of the System Architect is on system technology solutions, for example a component of a solution such as enterprise data warehousing.
- The **Industry Architect** has the responsibility for architectural design and documentation at an industry level. The focus of the Industry Architect is on industry problems and solutions, for examples petrochemical solutions, banking solutions, retail solutions.
- The **Organization Architect** has the responsibility for architectural design and documentation of specific organizations. An Organization Architect re-uses the output from all other architects. The focus of the Organization Architect The focus of the Industry Architect is on enterprise level business solutions in a given domain, such as finance, human resources, sales, etc.

Key Characteristics of an IT Architect

Skills and experience in producing designs

An IT Architect must be proficient in the techniques that go into producing designs of complex IT systems, including requirements discovery and analysis, formulation of solution context, identification of solution alternatives and their assessment, technology selection and design configuration.

Extensive technical breadth, with technical depth in one or a few disciplines

An IT Architect should possess an extensive technical breadth through experience in the IT industry. This breadth should be in areas of application development and deployment, and in the areas of creation and maintenance of the infrastructure to support the complex application environment. Current IT environments are heterogeneous by nature, and the experienced IT Architect will have skills across multiple platforms, including distributed systems and traditional mainframe environments. IT Architects will have as a result of their careers, skills in at least one discipline that is considered to be at the level of a subject matter expert.

Method-driven approach to execution

The IT Architect approaches his or her job through the consistent use of recognized design methods such as The Open Group Architecture Framework Architecture Development Method. The IT Architect should have working knowledge of more than one design method and be comfortable deploying parts of methods appropriate to the situation in which he or she is working. This should be seen in the body of design work the IT Architect has produced through repeated successful use of more than one design method. Proficiency in methodology use is in knowing what parts of methods to use in a given situation, and what methods not to use.

Full project scope experience

While the IT Architect is responsible for design and handoff of the project to implementers, it is vital that he or she have experience with all aspects of a project from design through development, testing, implementation, and production. This scope of experience will serve to keep IT Architects grounded in the notion of fitness-for-purpose and the practical nature of system implementation. The impact of full project scope experience should lead the IT Architect to make better design decisions, and better inform the trade offs made in those decisions.

Leadership

Communication and team building are key to the successful role of IT Architect. The mix of good technical skill and the ability to lead are crucial to the job. He or she should be viewed as a leader in the enterprise by the IT organization, the clients they serve, and management.

Personal and professional skills

The IT Architect must have strong communications and relationship skills. A major task of the IT Architect is to communicate complex technical information to all stakeholders of the project, including those who do not have a technical background. Strong negotiation and problem solving skills are also required. The IT Architect must work with the project management team to make decisions in a timely manner to keep projects on track.

Skills and experience in one or more industries

Industry skill and experience will make the task of gathering requirements and deciding priorities easier and more effective for the IT Architect. The IT Architect must understand the business processes of the enterprise in which he works, and how those processes work with other peer enterprises in the industry. He or she should also be able to spot key trends and correct flawed processes, giving the IT organization the capability to lead the enterprise, not just respond to requests. The mission of the IT Architect is strategic technical leadership.

Conclusions

The TOGAF Skills Framework provides an assessment of the skills required to deliver a successful Enterprise Architecture.

It is hoped that the provision of this Architecture Skills Framework will help reduce the time, cost, and risk involved in training, recruiting, and managing IT Architecture professionals, and at the same time enable and encourage more organizations to institute an internal IT Architecture Practice, hopefully based on (or at least leveraging) the Role and Skill definitions provided.

Developing Architecture Views – Introduction

The Role of Architecture Views

Introduction

Architecture views are representations of the overall architecture that are meaningful to one or more stakeholders in the system. The architect chooses and develops a set of views that will enable the architecture to be communicated to, and understood by, all the stakeholders, and enable them to verify that the system will address their concerns.

An architecture is usually represented by means of one or more architecture models that together provide a coherent description of the system's architecture. A single, comprehensive model is often too complex to be understood and communicated in its most detailed form, showing all the relationships between the various business and technical components. As with the architecture of a building, it is normally necessary to develop multiple *views* of the architecture of an information system, to enable the architecture to be communicated to, and understood by, the different stakeholders in the system.

For example, just as a building architect might create wiring diagrams, floor plans and elevations to describe different facets of a building to its different stakeholders (electricians, owners, planning officials), so an IT architect might create physical and security views of an IT system for the stakeholders who have concerns related to these aspects.

TOGAF and Standards for IT Architecture Description

An important recent development in IT architecture practice has been the emergence of standards for architecture description, principally through the adoption by ANSI and the IEEE of ANSI/IEEE Std 1471-2000 *Recommended Practice for Architectural Description of Software-Intensive Systems*. One of the aims of this standard is to promote a more consistent, systematic approach to the creation of views.

At the present time, TOGAF encourages but does not mandate the use of ANSI/IEEE Std 1471-2000.

Organizations that have incorporated, or plan to incorporate, ANSI/IEEE Std 1471-2000 into their IT architecture practice should find that none of the key concepts in TOGAF is incompatible with this standard, although some of the terminology used is not completely consistent with it.

In TOGAF we endeavor to strike a balance between promoting the concepts and terminology of ANSI/IEEE Std 1471-2000 - ensuring that our usage of terms defined by ANSI/IEEE Std 1471-2000 is consistent with the standard - and retaining other commonly accepted terminology that is familiar to the majority of the TOGAF readership.

An example of common terminology retained in TOGAF is the use of the terms "Business Architecture", "Technology Architecture", etc. These terms reflect common usage, but are at variance with ANSI/IEEE Std 1471-2000 (in which "architecture" is a property of a thing, not a thing in its own right). This situation will be reviewed in future Versions of TOGAF. The process of gradual convergence between TOGAF and relevant standards for architecture description will continue as ANSI/IEEE Std 1471-2000 gains increased acceptance within the industry.

More general information about ANSI/IEEE Std 1471-2000 can be obtained from the IEEE Architecture Working Group.

A Note on Terminology

It is arguable that the term "architecture" in this document should be replaced with the term "view", in accordance with ANSI/IEEE Std 1471-2000 recommended practice. There are practical problems with this.

Firstly, there is common usage. Typically an overall enterprise architecture comprising all four "architectures" (Business, Data, Applications, Technology) will not be undertaken as a single project. Rather each "architecture" - and in some cases, subsets

of them - will be undertaken as individual projects. The ultimate deliverable of such a project is commonly referred to as an "... architecture" (for example, a "business architecture"). Within such an "architecture" there will very likely be views, in the true ANSI/IEEE Std 1471-2000 sense.

Secondly, such individual projects - leading to a "Business Architecture", or an "Applications Architecture", etc. - are often undertaken without any intent to develop all four "architectures" and integrate them into an overall enterprise architecture. (Or at least, there may be a long-term strategic goal to develop all four, but the initial development may intended as a free-standing "architecture" and not a View of some larger entity.)

In summary, therefore, choice of terminology will depend largely on the extent to which the enterprise concerned regards each of the above as a part of a larger "enterprise architecture".

For the present, TOGAF retains the terminology of “Business Architecture”, "Technology Architecture”, etc., since the terminology associated with ANSI/IEEE Std 1471-2000 recommended practice is still relatively new to the industry and not yet in widespread use. This situation will be reviewed in future Versions of TOGAF.

Basic Concepts

The following concepts are central to the topic of views. These concepts have been adapted from more formal definitions contained in ANSI/IEEE Std 1471-2000 *Recommended Practice for Architectural Description of Software-Intensive Systems*.

A **system** is a collection of components organized to accomplish a specific function or set of functions.

The **architecture** of a system is the system's fundamental organization, embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution.

An **architecture description** is a collection of artifacts that document an architecture. In TOGAF, architecture views are the key artifacts in an architecture description.

Stakeholders are people who have key roles in, or concerns about, the system: for example, as users, developers, or managers. Different stakeholders with different roles in the system will have different concerns. Stakeholders can be individuals, teams, or organizations (or classes thereof).

Concerns are the key interests that are crucially important to the stakeholders in the system, and determine the acceptability of the system. Concerns may pertain to any aspect of the system' s functioning, development, or operation, including considerations such as performance, reliability, security, distribution, and evolvability.

A **view** is a representation of a whole system from the perspective of a related set of concerns.

In capturing or representing the design of a system architecture, the architect will typically create one or more architecture **models**, possibly using different tools. A view will comprise selected parts of one or more models, chosen so as to demonstrate to a particular stakeholder or group of stakeholders that their concerns are being adequately addressed in the design of the system architecture.

A **viewpoint** defines the perspective from which a view is taken. More specifically, a viewpoint defines: how to construct and use a view (by means of an appropriate schema or template); the information that should appear in the view; the modelling techniques for expressing and analysing the information; and a rationale for these choices (e.g., by describing the purpose and intended audience of the view).

- A *view* is what you see. A *viewpoint* is where you are looking from - the vantage point or perspective that determines what you see.
- Viewpoints are generic, and can be stored in libraries for reuse. A view is always specific to the architecture for which it is created.
- Every view has an associated viewpoint that describes it, at least implicitly. ANSI/IEEE Std 1471-2000 encourages architects to define viewpoints explicitly. Making this distinction between the content and schema of a view may seem at first to be an unnecessary overhead, but it provides a mechanism for reusing viewpoints across different architectures.

In summary, then, architecture views are representations of the overall architecture in terms meaningful to stakeholders.

They enable the architecture to be communicated to and understood by the stakeholders, so they can verify that the system will address their concerns.

> *Note: The terms 'concern' and 'requirement' are not synonymous. A concern is an area of interest. So, system reliability might be a concern/area of interest for some stakeholders. The reason why architects should identify concerns and associate them with viewpoints, is to insure that those concerns will be addressed in some fashion by the models of the architecture. For example, if the only viewpoint selected by an architect is a structural viewpoint, then reliability concerns are almost certainly not being addressed, since they cannot be represented in a structural model. Within that concern, stakeholders may have many distinct requirements: different classes of users may have very different reliability requirements for different capabilities of the system.*
>
> *Concerns are the root of the process of decomposition into requirements. Concerns are represented in the architecture by these requirements. Requirements should be SMART (e.g., specific metrics).*

A Simple Example of a Viewpoint and View

For many architectures, a useful viewpoint is that of Business Domains, which can be illustrated by an example from The Open Group itself.

The viewpoint is specified as follows:

Viewpoint element	*Description*
Stakeholders:	Management Board, Chief Information Officer
Concerns:	Show the top-level relationships between geographical sites and business functions.
Modeling technique:	Nested boxes diagram. Blue = locations; brown = business functions. Semantics of nesting = functions performed in the locations.

The corresponding view of The Open Group (in 2001) is shown in Figure 1.

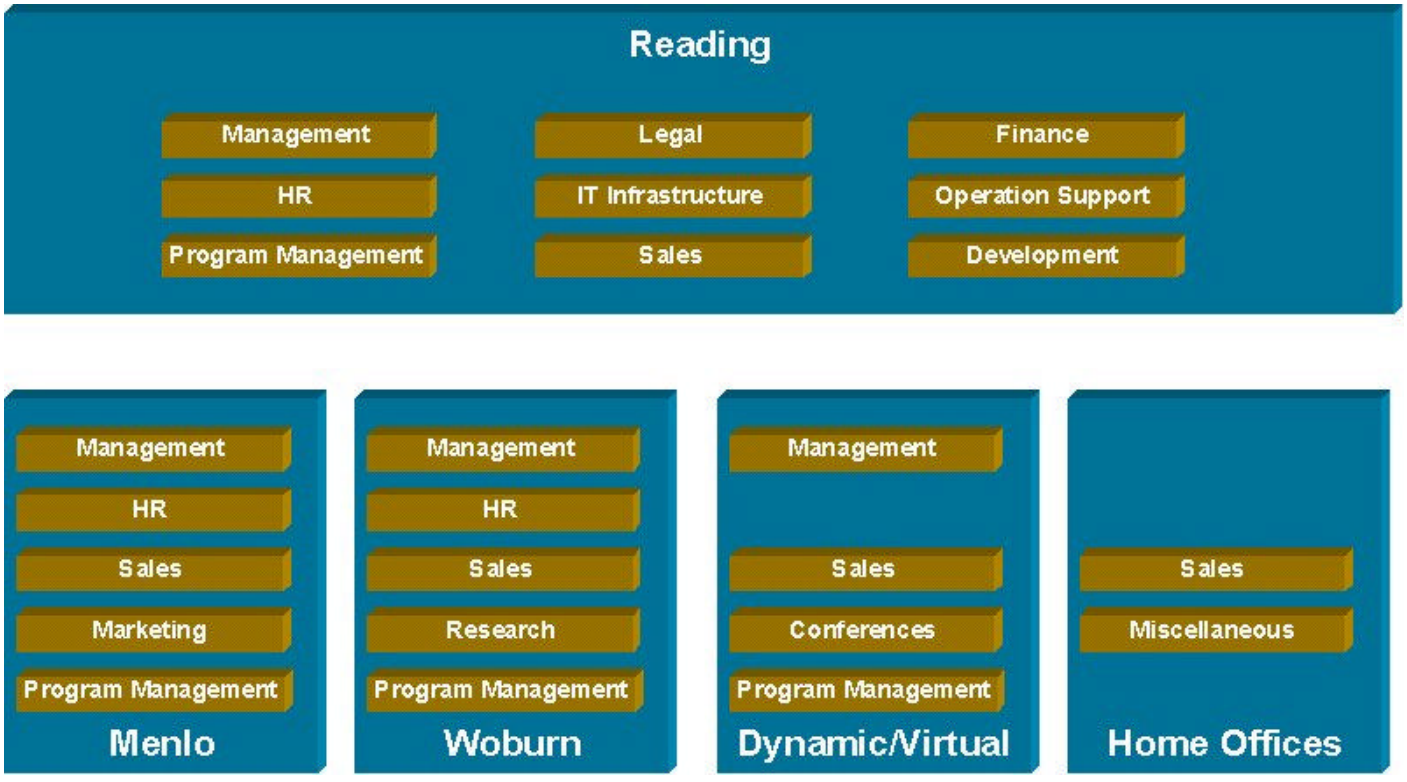


Figure 1: Example View - The Open Group Business Domains in 2001

Developing Views in the ADM

General Guidelines

The choice of which particular architecture views to develop is one of the key decisions that the architect has to make.

The architect has a responsibility for ensuring the *completeness* (fitness-for-purpose) of the architecture, in terms of adequately addressing all the pertinent concerns of its stakeholders; and the *integrity* of the architecture, in terms of connecting all the various views to each other, satisfactorily reconciling the conflicting concerns of different stakeholders, and showing the trade-offs made in so doing (as between security and performance, for example).

The choice has to be constrained by considerations of practicality, and by the principle of fitness-for-purpose (i.e., the architecture should be developed only to the point at which it is fit for purpose, and not reiterated ad infinitum as an academic exercise).

As explained in Part II, Architecture Development Method, the development of architecture views is an iterative process. The typical progression is from business to technology, using a technique such as Business Scenarios to properly identify all pertinent concerns; and from high level overview to lower level detail, continually referring back to the concerns and requirements of the stakeholders throughout the process.

Moreover, each of these progressions has to be made for two distinct environments: the existing environment (referred to as the *baseline* in the ADM) and the target environment. The architect must develop pertinent business and technical architecture views of both the existing system and the target system. This provides the context for the gap analysis at the end of Phase C of the ADM, which establishes which elements of the current system must be carried forward and which must be removed or replaced.

This whole process is explained in Part II, Architecture Development Method, under Target Architecture Development.

The View Creation Process

As mentioned above, at the present time TOGAF encourages but does not mandate the use of ANSI/IEEE Std 1471-2000. The following description therefore covers both the situation where ANSI/IEEE Std 1471-2000 has been adopted and where it has not.

IEEE 1471-2000 itself does not require any specific process for developing viewpoints or creating views from them. Where ANSI/IEEE Std 1471-2000 has been adopted and become well-established practice within an organization, it will often be possible to create the required views for a particular architecture by following these steps:

1. Refer to an existing library of viewpoints.
2. Select the appropriate viewpoints (based on the stakeholders and concerns that need to be covered by views).
3. Generate views of the system by using the selected viewpoints as templates.

This approach can be expected to bring the following benefits:

- Less work for the architects (because the viewpoints have already been defined and therefore the views can be created faster).
- Better comprehensibility for stakeholders (because the viewpoints are already familiar).
- Greater confidence in the validity of the views (because their viewpoints have a known track record).

However, situations can always arise in which a view is needed for which no appropriate viewpoint has been pre-defined. This is also the situation, of course, when an organization has not yet incorporated ANSI/IEEE Std 1471-2000 into its architecture practice and established a library of viewpoints.

In each case, the architect may choose to develop a new viewpoint that will cover the outstanding need, and then generate a view from it. (This is the ANSI/IEEE Std 1471-2000 recommended practice.) Alternatively, a more pragmatic approach can be equally successful: the architect can create an *ad hoc* view for a specific system and later consider whether a generalised form of the implicit viewpoint should be defined explicitly and saved in a library, so that it can be reused. (This is one way of establishing a library of viewpoints initially.)

Whatever the context, the architect should be aware that every view has a viewpoint, at least implicitly, and that defining the viewpoint in a systematic way (as recommended by ANSI/IEEE Std 1471-2000) will help in assessing its effectiveness (i.e.,

does the viewpoint cover the relevant stakeholder concerns?).

Core Taxonomy of Architecture Views

Overview

TOGAF's core taxonomy of architecture views defines the minimum set of views that should be considered in the development of an architecture.

Since in ANSI/IEEE Std 1471-2000 every view has an associated viewpoint that defines it, this taxonomy may also be regarded as a taxonomy of viewpoints by those organizations that have adopted ANSI/IEEE Std 1471-2000.

Stakeholders

The minimum set of stakeholders for a system that should be considered in the development of architecture viewpoints and views is:

- Users
- System and Software Engineers
- Operators, Administrators, and Managers
- Acquirers

Views / Viewpoints

The architecture views, and corresponding viewpoints, that may be created to support each of these stakeholders fall into the following categories. (As mentioned above, this taxonomy may be regarded as a taxonomy of viewpoints by those organizations that have adopted ANSI/IEEE Std 1471-2000.)

- **Business Architecture Views**, which address the concerns of the users of the system, and describe the flows of business information between people and business processes
- **Data Architecture Views**, which address the concerns of Database Designers and Database Administrators, and System Engineers responsible for developing and integrating the various database components of the system.
- **Applications Architecture Views**, which address the concerns of System and Software Engineers responsible for developing and integrating the various application software components of the system.
- **Technology Architecture Views**, which address the concerns of Acquirers (procurement personnel responsible for acquiring the commercial-off-the-shelf (COTS) software and hardware to be included in the system), Operations staff, Systems Administrators, and Systems Managers.

Examples of specific views that may be created in each category are tabulated below, and explained in detail in the following subsection.

To address the concerns of the following stakeholders...			
Users, Planners, Business Management	**Database Designers and Administrators, System Engineers**	**System and Software Engineers**	**Acquirers, Operators, Administrators & Managers**
... the following views may be developed:			
Business Architecture Views	**Data Architecture Views**	**Applications Architecture Views**	**Technology Architecture Views**
Business Function View	Data Entity View	Software Engineering View	Networked Computing/ Hardware

Business Services View				View
Business Process View				
Business information View				
Business Locations View				Communications Engineering View
Business Logistics View	Data Flow View (Organization Data Use)	Applications Interoperability View		
People View (organization chart)				Processing View
Workflow View				
Usability View				
Business Strategy and Goals View	Logical Data View	Software Distribution View		Cost View
Business Objectives View				
Business Rules View				Standards View
Business Events View				
Business Performance View				
	System Engineering View			
Enterprise Security View				
Enterprise Manageability View				
Enterprise Quality of Service View				
Enterprise Mobility View				

Table 1: Example Taxonomy of Architecture Views

A mapping of these views to the schema of the well known Zachman Framework is illustrated below.

	Stakeholder	**DATA**	**FUNCTION**	**NETWORK**	**PEOPLE**	**TIME**	**MOTIVATION**
SCOPE	*Planner*	Data Entity View (Class Model)	Business Function View	Business Locations View	People View (org chart)	Business Events View	Business Strategy and Goals View
			Business Services View	Enterprise Mobility View		Enterprise Quality of Service View	
ENTERPRISE MODEL	*Owner*	Data Flow View (Organization Data Use)	Business Services View	Business Logistics View (Business Function to Location mapping)	Workflow View	Business Performance View (master schedule)	Business Objectives View (SMART objectives from Business Scenario)
			Business Process View	Enterprise Mobility View		Enterprise Quality of Service View	
SYSTEM MODEL	*Designer*	System Engineering View	System Engineering View	System Engineering View	Usability View	System Engineering View	Business Rules View
			Software Engineering View			Processing View	
				Standards			

		Logical Data View Standards View	Application-to-application Communication View Standards View	View Enterprise Mobility View	Standards View	Standards View Enterprise Quality of Service View	Cost View
TECHNOLOGY CONSTRAINED MODEL	*Builder*	Physical Data View **(out of TOGAF scope)**	Software Distribution View	Networked Computing/ Hardware View Communications Engineering View	Usability View	Control structure **(out of TOGAF scope)**	Business Logic (Rules) Design **(out of TOGAF scope)**
DETAILED REPRESENTATIONS	*Subcontractor*	Data Definitions **(out of TOGAF scope)**	Application Program Code **(out of TOGAF scope)**	**(out of TOGAF scope)**	**(out of TOGAF scope)**	Timing Definitions **(out of TOGAF scope)**	Application Program (Rules Specification) **(out of TOGAF scope)**
FUNCTIONING ENTERPRISE		Enterprise Security View Enterprise Mobility View Enterprise Quality of Service View Enterprise Manageability View	Enterprise Security View Enterprise Mobility View Enterprise Quality of Service View Enterprise Manageability View	Enterprise Security View Enterprise Mobility View Enterprise Quality of Service View Enterprise Manageability View	Enterprise Security View Enterprise Mobility View Enterprise Quality of Service View	Enterprise Security View Enterprise Mobility View Enterprise Quality of Service View Enterprise Manageability View	Enterprise Security View Enterprise Mobility View Enterprise Quality of Service View

Table 2: Mapping of Example Taxonomy of Architecture Views to Zachman Framework

The Architect may or may not need to develop separate views for different stakeholders: for example, for engineers and operations personnel. Engineering views provide information needed by engineering staff to design and implement the hardware and software system. Operations Views provide information needed by Operations staff in order to operate and administer the implemented system.

Description

The following description explains some of the views listed above. A more detailed description of each view, and guidelines for developing it, can be obtained by following the relevant hyperlink in the following.

1. Business Architecture Views address the concerns of **Users, Planners, and Business Managers,** and focuses on the functional aspects of the system from the perspective of the users of the system - that is, on what the new system is intended to do, including performance, functionality, and useability. These can be built up from an analysis of the existing environment and of the requirements and constraints affecting the new system.

- The **People View** focuses on the human resource aspects of the system. It examines the human actors involved in the system.
- The **Business Process View** deals with the user processes involved in the system.
- The **Business Function View** deals with the functions required to support the processes.
- The **Business Information View** deals with the information required to flow in support of the processes.
- The **Usability View** considers the usability aspects of the system and its environment.
- The **Business Performance View** considers the performance aspects of the system and its environment.

2. **Data Architecture Views and Applications Architecture Views** address the concerns of the **Database Designers and Administrators**, and the **System and Software Engineers** of the system. They focus on how the system is

implemented from the prespective of different types of engineers (security, software, data, computing components, communications), and how that affects its properties. Systems and software engineers are typically concerned with modifiability, reusability and availability of other services:

- The Data Flow View deals with the architecture of the storage, retrieval, processing, archiving and security of data. It looks at the flow of data as it is stored and processed, and at what components will be required to support and manage both storage and processing.
- The Software Engineering View deals with aspects of interest to software developers. It considers what software development constraints and opportunities exist in the new system, and looks at how development can be carried out, both in terms of technology and resources. The Software Engineering View is particularly important in that it provides a reference for selection of building blocks related to elements of the existing system that may be reused in the target architecture.
- The System Engineering View presents a number of different ways in which software and hardware components can be assembled into a working system. To a great extent the choice of model determines the properties of the final system. It looks at technology which already exists in the organization, and what is available currently or in the near future. This reveals areas where new technology can contribute to the function or efficiency of the new architecture, and how different types of processing platform can support different parts of the overall system.

3. **Technology Architecture Views** address the concerns of the **Acquirers, Operators, Communications Engineers, Administrators & Managers** of the system.

- The Communications Engineering View addresses the concerns of the Communications Engineers. It examines various ways of structuring communications facilities to simplify the business of network planning and design. It examines the networking elements of the architecture in the light of geographic constraints, bandwidth requirements and so on.
- Acquirers Views address the needs of an acquirer or procurer, providing appropriate guidance for purchasing components that "fit" the architecture. Acquirer's views of the architecture are primarily concerned with costs, and standards that must be adhered to; for example:
    - The Cost View
    - The Standards View

  These views typically depict building blocks of the architecture that can be purchased, and the standards that the building blocks must adhere to in order for the building block to be most useful.

4. Composite Views:

- The Enterprise Manageability View addresses the concerns of the Operations, Administration, and Management of the system, and concentrates more on the details of location, type and power of the equipment and software in order to manage the health and availability of the system. It covers issues such as initial deployment, upgrading, availability, security, performance, asset management, fault and event management of system components, from the management perspective of the following subject matters.
    - Security
    - Software
    - Data
    - Computing/Hardware
    - Communications
- The Enterprise Security View focuses on the security aspects of the system for the protection of information within the organization. It examines the system to establish what information is stored and processed, how valuable it is, what threats exist, and how they can be addressed.

Architects also have concerns of their own which basically define the fitness for purpose of an effort. Architects must understand completeness, where completeness includes considering all relevant views, the relationships between those views, and dealing with the conflicts that arise from those different views. Architects also must deal with viability of the architecture: if the architecture is not capable of being implemented, then its value is in doubt.

Views, Tools, and Languages

The need for architecture views, and the process of developing them following the Architecture Development Method, are explained above. This subsection describes the relationships between architecture views, the tools used to develop and analyse them, and a standard language enabling interoperability between the tools.

Overview

In order to achieve the goals of completeness and integrity in an architecture, architecture views are usually developed, visualized, communicated, and managed using a tool.

In the current state of the market, different tools normally have to be used to develop and analyse different views of the architecture. It is highly desirable that an architecture description be encoded in a standard language, to enable a standard approach to the description of architecture semantics and their reuse among different tools.

A viewpoint is also normally developed, visualized, communicated, and managed using a tool, and it is also highly desirable that standard viewpoints (i.e., templates, or schemas) be developed, so that different tools that deal in the same views can interoperate, the fundamental elements of an architecture can be reused, and the architecture description can be shared among tools.

Issues relating to the evaluation of tools for architecture work are discussed in detail in the section Architecture Tools.

Views and Viewpoints

Example of Views and Viewpoints

To illustrate the concepts of views and viewpoints, consider the example of a very simple airport system with two different stakeholders, the pilot and the air traffic controller.

The pilot has one view of the system, and the air traffic controller has another. Neither view represents the whole system, because the perspective of each stakeholder constrains (and reduces) how each sees the overall system.

The view of the pilot comprises some elements not viewed by the controller, such as passengers and fuel, while the view of the controller comprises some elements not viewed by the pilot, such as other planes. There are also elements shared between the views, such as the communication model between the pilot and the controller, and the vital information about the plane itself.

A viewpoint is a model (or description) of the information contained in a view. In our example, one viewpoint is the description of how the pilot sees the system, and the other viewpoint is how the controller sees the system.

Pilots describe the system from their perspective, using a model of their position and vector toward or away from the runway. All pilots use this model, and the model has a specific language that is used to capture information and populate the model.

Controllers describe the system differently, using a model of the airspace and the locations and vectors of aircraft within the airspace. Again, all controllers use a common language derived from the common model in order to capture and communicate information pertinent to their viewpoint.

Fortunately, when controllers talk with pilots, they use a common communication language! (In other words, the models representing their individual viewpoints partially intersect.) Part of this common language is about location and vectors of aircraft, and is essential to safety.

So in essence each viewpoint is an abstract model of how all the stakeholders of a particular type - all pilots, or all controllers - view the airport system.

Tools exist to assist stakeholders, especially when they are interacting with complex models such as the model of an airspace, or the model of air flight.

The interface to the human user of a tool is typically close to the model and language associated with the viewpoint. The unique tools of the pilot are fuel, altitude, speed, and location indicators. The main tool of the controller is radar. The common tool is a radio.

To summarise from the above example, we can see that a view can subset the system through the perspective of the stakeholder, such as the pilot versus the controller. This subset can be described by an abstract model called a viewpoint, such as an air flight versus an air space model. This description of the view is documented in a partially specialized language, such as "pilot-speak" versus "controller-speak". Tools are used to assist the stakeholders, and they interface with each other in terms of the language derived from the viewpoint ("pilot-speak" versus "controller-speak").

When stakeholders use common tools, such as the radio contact between pilot and controller, a common language is essential.

Views and Viewpoints in Information Systems

Now let us map this example to information systems architecture. Consider two stakeholders in a new small computing system: the users, and the developers.

The users of the system have a view of the system, and the developers of the system have a different view. Neither view represents the whole system, because each perspective reduces how each sees the system.

The view of the user is comprised of all the ways in which s/he interacts with the system, not seeing any details such as applications or database management systems.

The view of the developer is one of productivity and tools, and doesn't include things such as actual live data and connections with consumers.

However, there are things that are shared, such as descriptions of the processes that are enabled by this system and / or communications protocols set up for users to communicate problems directly to development.

In this example, one viewpoint is the description of how the user sees the system, and the other viewpoint is how the developer sees the system. Users describe the system from their perspective, using a model of availability, response time, and access to information. All users of the system use this model, and the model has a specific language.

Developers describe the system differently than users, using a model of software connected to hardware distributed over a network, etc. However, there are many types of developers (database, security, …) of the system, and they do not have a common language derived from the model.

The Need for a Common Language and Interoperable Tools for Architecture Description

Tools exist for both users and developers. Tools such as on-line help are there specifically for users, and attempt to use the language of the user. Many different tools exist for different types of developers, but they suffer from the lack of a common language that is required to bring the system together. It is difficult, if not impossible, in the current state of the tools market to have one tool interoperate with another tool.

Issues relating to the evaluation of tools for architecture work are discussed in detail in the section Architecture Tools.

Conclusions

This section of TOGAF attempts to deal with views in a structured manner, but this is by no means a complete treatise on views.

In general, TOGAF embraces the concepts and definitions presented in ANSI/IEEE Std 1471-2000, specifically the concepts that help guide the development of a view and make the view actionable. These concepts can be summarized as:

- selecting a key stakeholder;
- understanding their concerns and generalizing/documenting those concerns; and
- understanding how one models and deals with those concerns.

In the following subsections TOGAF presents some recommended views, some or all of which may be appropriate in a particular architecture development. This is not intended as an exhaustive set of views, but simply as a starting point. Those described may be supplemented by additional views as required. These TOGAF subsections on views should be considered as guides for the development and treatment of a view, not as a full definition of a view.

Each subsection describes the stakeholders related to the view, their concerns, and the entities modeled and the language used to depict the view (the viewpoint). The viewpoint provides architecture concepts from the different perspectives, including components, interfaces, and allocation of services critical to the view. The viewpoint language, analytical methods and modeling methods associated with views are typically applied with the use of appropriate tools.

Developing a Business Architecture View

Stakeholder and Concerns Modeling the View Key Issues

Stakeholder and Concerns

This view should be developed for the users. It focuses on the functional aspects of the system from the perspective of the users of the system.

Addressing the concerns of the users includes consideration of the following:

- People - the human resource aspects of the system. It examines the human actors involved in the system.
- Process - deals with the user processes involved in the system.
- Function - deals with the functions required to support the processes.
- Business Information - deals with the information required to flow in support of the processes.
- Usability - considers the usability aspects of the system and its environment.
- Performance - considers the performance aspects of the system and its environment.

Modeling the View

Business scenarios are an important technique that may be used prior to, and as a key input to, the development of the Business Architecture View, to help identify and understand business needs, and thereby to derive the business requirements and constraints that the architecture development has to address. Business scenarios are an extremely useful way to depict what should happen when planned and unplanned events occur. It is highly recommended that business scenarios be created for planned change, and for unplanned change.

The following paragraphs describe some of the key issues that the architect might consider when constructing business scenarios.

Key Issues

The Business Architecture View considers the functional aspects of the system - that is, what the new system is intended to do. This can be built up from an analysis of the existing environment and of the requirements and constraints affecting the new system.

The new requirements and constraints will appear from a number of sources, possibly including:

- Existing internal specifications and lists of approved products
- Business goals and objectives
- Business Process Re-engineering activities
- Changes in technology

What should emerge from the Business Architecture View is a clear understanding of the functional requirements for the new architecture, with statements like "Improvements in handling customer inquiries are required through wider use of Computer/Telephony Integration".

The Business Architecture View considers the usability aspects of the system and its environment. It should also consider impacts on the user such as skill levels required, the need for specialized training, and migration from current practice. When considering usability the architect should take into account:

- the ease-of-use of the user interface, and how intuitive it is
- whether or not there is transparent access to data and applications, irrespective of location
- ease of management of the user environment by the user
- application interoperability through means such as drag and drop
- on-line help facilities
- clarity of documentation

- security and password aspects, such as avoiding the requirement for multiple sign-on and password dialogues
- access to productivity applications such as mail or a spreadsheet

Note that, although security and management are thought about here, it is from a usability and functionality point of view. The technical aspects of security and management are considered in the Security View and the Operations View.

Developing an Enterprise Security View

Stakeholder and Concerns Modeling the View Concepts Generic View Services Allocation

Stakeholder and Concerns

This view should be developed for security engineers of the system. It focuses on how the system is implemented from the perspective of security, and how security affects the system properties. It examines the system to establish what information is stored and processed, how valuable it is, what threats exist, and how they can be addressed.

Major concerns for this view are understanding how to assure that the system is available to only those that have permission, and how to protect the system from unauthorized tampering.

Modeling the View

The subjects of the general architecture of a 'security system' are components that are secured, or components that provide security services. Additionally Access Control Lists and security schema definitions are used to model and implement security.

Basic Concepts

This section presents basic concepts required for an understanding of information system security.

The essence of security is the controlled use of information. The purpose of this section is to provide a brief overview of how security protection is implemented in the components of an information system. Doctrinal or procedural mechanisms, such as physical and personnel security procedures and policy, are not discussed here in any depth.

Figure 1 depicts an abstract view of an information system architecture, which emphasizes the fact that an information system from the security perspective is either part of a local subscriber environment (LSE) or a communications network (CN). An LSE may be either fixed or mobile. The LSEs by definition are under the control of the using organization. In an open system distributed computing implementation, secure and non-secure LSEs will almost certainly be required to interoperate.

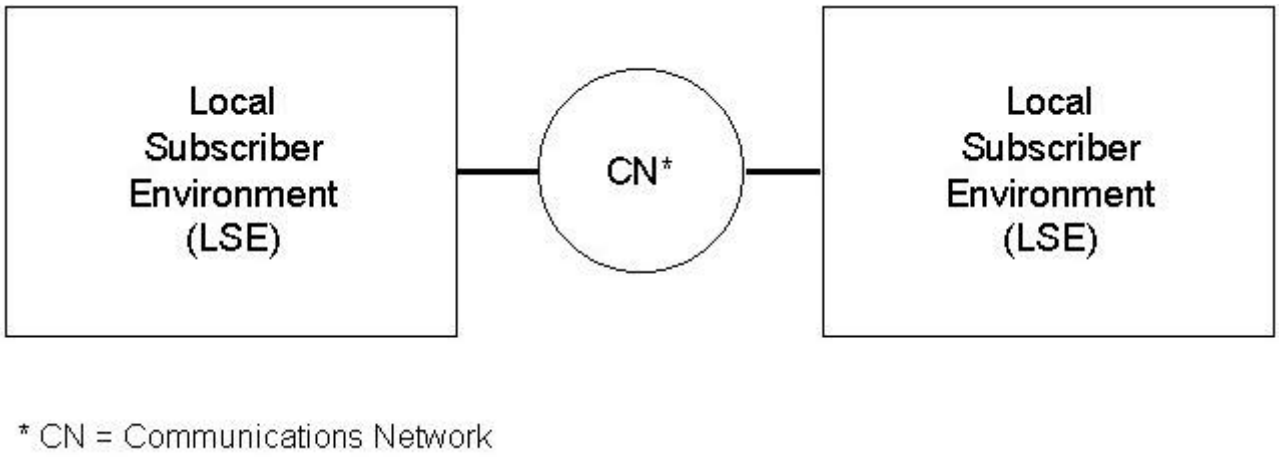


Figure 1 Abstract Security Architecture View

Information Domains

The concept of an information domain provides the basis for discussing security protection requirements. An information domain is defined as a set of users, their information objects, and a security policy. An information domain security policy is the statement of the criteria for membership in the information domain and the required protection of the information objects. Breaking an organization's information down into domains is the first step in reducing the task of security policy development

to a manageable size.

The business of most organizations requires that their members operate in more than one information domain. The diversity of business activities and the variation in perception of threats to the security of information will result in the existence of different information domains within one organization security policy. A specific activity may use several information domains, each with its own distinct information domain security policy.

Information domains are not necessarily bounded by information systems or even networks of systems. The security mechanisms implemented in information system components may be evaluated for their ability to meet the information domain security policies.

Strict Isolation

Information domains can be viewed as being strictly isolated from one another. Information objects should be transferred between two information domains only in accordance with established rules, conditions, and procedures expressed in the security policy of each information domain.

Absolute Protection

The concept of "absolute protection" is used to achieve the same level of protection in all information systems supporting a particular information domain. It draws attention to the problems created by interconnecting LSEs that provide different strengths of security protection. This interconnection is likely because open systems may consist of an unknown number of heterogeneous LSEs. Analysis of minimum security requirements will ensure that the concept of absolute protection will be achieved for each information domain across LSEs.

Security Generic Architecture View

Figure 2 shows a generic architectural view which can be used to discuss the allocation of security services and the implementation of security mechanisms. This view identifies the architectural components within a LSE. The LSEs are connected by CNs. The LSEs include end systems, relay systems, and local communications systems (LCSs), described below.

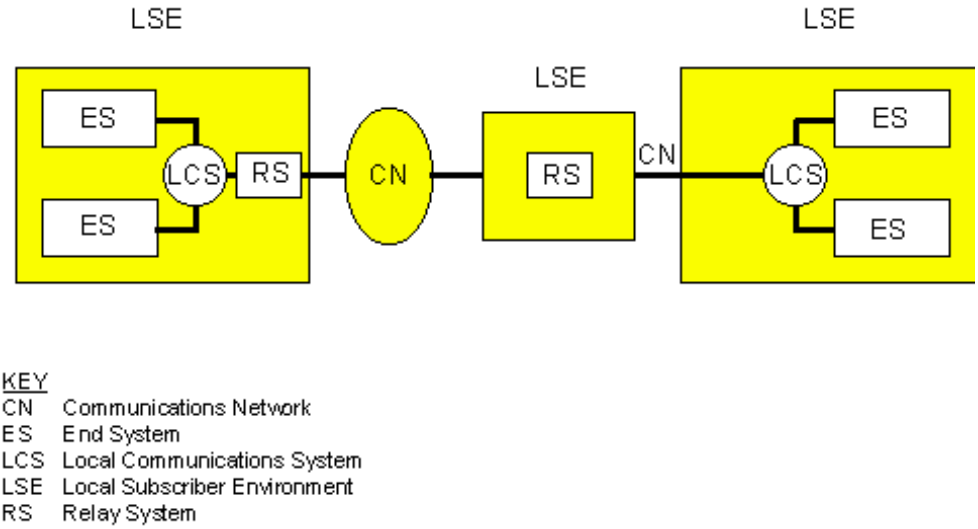


Figure 2: Generic Security Architecture View

- Relay System (RS) - The component of an LSE, the functionality of which is limited to information transfer and is only indirectly accessible by users (e.g., router, switch, multiplexor, message transfer agent). It may have functionality similar to an end system, but an end user does not use it directly. Note that relay system functions may be provided in an end system.
- Local Communication System (LCS) - A network that provides communications capabilities between LSEs or within a LSE with all of the components under control of a LSE.
- Communication Network (CN) - A network that provides inter-LSE communications capabilities, but is not controlled by LSEs (e.g., commercial carriers).

The end system and the relay system are viewed as requiring the same types of security protection. For this reason, a

discussion of security protection in an end system generally also applies to a relay system. The security protections in an end system could occur in both the hardware and software.

Security Services Allocation

Security protection of an information system is provided by mechanisms implemented in the hardware and software of the system and by the use of doctrinal mechanisms. The mechanisms implemented in the system hardware and software are concentrated in the end system or relay system. This focus for security protection is based on the open system, distributed computing approach for information systems. This implies use of commercial common carriers and private common-user communications systems as the CN provider between LSEs. Thus, for operation of end systems in a distributed environment, a greater degree of security protection can be assured from implementation of mechanisms in the end system or relay system.

However, communications networks (CNs) should satisfy the availability element of security in order to provide appropriate security protection for the information system. This means that CNs must provide an agreed level of responsiveness, continuity of service, and resistance to accidental and intentional threats to the communications service availability.

Implementing the necessary security protection in the end system occurs in three system service areas of TOGAF. They are operating system services, network services, and system management services.

Most of the implementation of security protection is expected to occur in software. The hardware is expected to protect the integrity of the end system software. Hardware security mechanisms include protection against tampering, undesired emanations, and cryptography.

Operating System Services

A "security context" is defined as a controlled process space subject to an information domain security policy. The security context is therefore analogous to a common operating system notion of user process space. Isolation of security contexts is required. Security contexts are required for all applications (e.g., end user and security management applications). The focus is on strict isolation of information domains, management of end system resources, and controlled sharing and transfer of information among information domains. Where possible, security-critical functions should be isolated into relatively small modules that are related in well-defined ways.

The operating system will isolate multiple security contexts from each other using hardware protection features (e.g., processor state register, memory mapping registers) to create separate address spaces for each of them. Untrusted software will use end system resources only by invoking security-critical functions through the separation kernel. Most of the security-critical functions are the low-level functions of traditional operating systems.

Network Services

Two basic classes of communications are envisioned for which distributed security contexts may need to be established. These are interactive and staged (store and forward) communications.

The concept of a "security association" forms an interactive distributed security context. A security association is defined as all the communication and security mechanisms and functions that extend the protections required by an information domain security policy within an end system to information in transfer between multiple end systems. The security association is an extension or expansion of an OSI application layer association. An application layer association is composed of appropriate application layer functions and protocols plus all of the underlying communications functions and protocols at other layers of the OSI model. Multiple security protocols may be included in a single security association to provide for a combination of security services.

For staged delivery communications (e.g., electronic mail), use will be made of an encapsulation technique (termed "wrapping process") to convey the necessary security attributes with the data being transferred as part of the network services. The wrapped security attributes are intended to permit the receiving end system to establish the necessary security context for processing the transferred data. If the wrapping process cannot provide all the necessary security protection, interactive security contexts between end systems will have to be used to ensure the secure staged transfer of information.

System Security Management Services

Security management is a particular instance of the general information system management functions discussed in earlier chapters of TOGAF. Information system security management services are concerned with the installation, maintenance, and enforcement of information domain and information system security policy rules in the information system intended to provide

these security services. In particular, the security management function controls information needed by operating system services within the end system security architecture. In addition to these core services, security management requires event handling, auditing, and recovery. Standardization of security management functions, data structures, and protocols will enable interoperation of security management application processes (SMAPs) across many platforms in support of distributed security management.

[1]

Developing a Software Engineering View

Stakeholders and Concerns

Building a software-intensive system is both expensive and time consuming. Because of this, it is necessary to establish guidelines to help minimize the effort required and the risks involved. This is the purpose of the Software Engineering View, which should be developed for the software engineers who are going to develop the system.

Major concerns for these stakeholders are:

- Development Approach
- Software Modularity and Re-Use
- Portability
- Migration and Interoperability

Development Approach

There are many lifecycle models defined for software development (waterfall, prototyping, etc.). A consideration for the architect is how best to feed architectural decisions into the lifecycle model that is going to be used for development of the system.

Software Modularity and Re-Use

As a piece of software grows in size, so the complexity and interdependencies between different parts of the code increase. Reliability will fall dramatically unless this complexity can be brought under control.

Modularity is a concept by which a piece of software is grouped into a number of distinct and logically cohesive sub-units, presenting services to the outside world through a well defined interface. Generally speaking, the components of a module will share access to common data, and the interface will provide controlled access to this data. Using modularity, it becomes possible to build a software application incrementally on a reliable base of pre-tested code.

A further benefit of a well defined modular system is that the modules defined within it may be re-used in the same or on other projects, cutting development time dramatically by reducing both development and testing effort.

In recent years, the development of Object Oriented Programming Languages has greatly increased programming language support for module development and code re-use. Such languages allow the developer to define 'classes' (a unit of modularity) of objects that behave in a controlled and well-defined manner. Techniques such as inheritance - which enables parts of an existing interface to an object to be changes- enhance the potential for re-usability by allowing pre-defined classes to be tailored or extended when the services they offer do not quite meet the requirement of the developer.

If modularity and *software re-use* are likely to be key objectives of new software developments, consideration must be given to whether the component parts of any proposed architecture may facilitate or prohibit the desired level of modularity in the appropriate areas.

Portability

Software portability, the ability to take a piece of software written in one environment and make it run in another, is important in many projects, especially product developments. It requires that all software and hardware aspects of a chosen

technical architecture (not just the newly developed application) be available on the new platform. It will, therefore, be necessary to ensure that the component parts of any chosen architecture are available across all the appropriate target platforms.

Migration and Interoperability

Interoperability is always required between the component parts of a new architecture. It may also, however, be required between a new architecture and parts of an existing legacy system - for example during the staggered replacement of an old system. Interoperability between the new and old architectures may, therefore, be a factor in architectural choice.

Key Issues

- Data Intensive Versus Information Intensive Software Systems
- Achieving Interoperability
- Software Tiers
- Uses of a Data Access Tier
- Distribution

Data Intensive Versus Information Intensive Software Systems

This View considers two general categories of software systems. First, there are those systems that require only a user interface to a database, requiring little or no business logic built into the software. These systems can be called “Data Intensive.” Second, there are those systems that require users to manipulate information that might be distributed across multiple databases, and to do this manipulation according to a predefined business logic. These systems can be called “Information Intensive.”

Data intensive systems can be built with reasonable ease through the use of 4GL tools. In these systems, the business logic is in the mind of the user, i.e., the user understands the rules for manipulating the data and uses those rules while doing his work.

Information intensive systems are different. Information is defined as “meaningful data,” i.e., data in a context that includes business logic. Information is different from data. Data is the tokens that are stored in databases or other data stores. Information is multiple tokens of data combined to convey a message. For example, “3” is data, but “3 widgets” is information. Typically, information reflects a model. Information intensive systems also tend to require information from other systems, and, if this path of information passing is automated, usually some mediation is required to convert the format of incoming information into a format that can be locally used. Because of this, information intensive systems tend to be more complex than others, and require the most effort to build, integrate, and maintain.

This view is concerned primarily with information intensive systems. In addition to building systems that can manage information, though, systems should also be as flexible as possible. This has a number of benefits. It allows the system to be used in different environments, for example, the same system should be usable with different sources of data, even if the new data store is a different configuration. Similarly, it might make sense to use the same functionality but with users who need a different user interface. So information systems should be built so that they can be reconfigured with different data stores or different user interfaces. If a system is built to allow this, it enables the enterprise to reuse parts (or “components”) of one system in another.

Achieving Interoperability [2]

Interoperability can only be achieved when information is passed, not when data is passed. Most information systems today get information both from their own data stores and other information systems. In some cases the web of connectivity between information systems is quite extensive. The United States Air Force, for example, has a concept known as “A5 Interoperability.” This means that the required data is available Anytime, Anywhere, by Anyone, who is Authorized, in Any way. This requires that many information systems are architecturally linked and provide information to each other.

There must be some kind of physical connectivity between the systems. This might be a local area network, it might be a wide area network, or, in some cases, it might simply be the passing of a disk or CD between systems.[3] Assuming a network connects the systems, there must be agreement on the protocols used. This enables the transfer of bits.

When the bits are assembled at the receiving system, they must be placed in the context that the receiving system needs. In other words, both the source and destination systems must agree on an information model. The source system uses this model to convert its information into data to be passed, and the destination system uses this same model to convert the received data into information it can use.

This usually requires an agreement between the architects and designers of the two systems. In the past, this agreement was often documented in the form of an "Interface Control Document" (ICD). The ICD defines the exact syntax and semantics that the sending system will use so that the receiving system will know what to do when the data arrives. The biggest problem with ICDs is that they tend to be unique solutions between two systems. If a given system must share information with 'n' other systems, there is the potential need for n^2 ICDs. This extremely tight integration prohibits flexibility and the ability of a system to adapt to a changing environment. Maintaining all these ICDs is also a challenge.

New technology such as eXtensible Markup Language (XML) has the promise of making data "self describing". Use of new technologies such as XML, once they become reliable and well documented, might eliminate the need for an ICD. Further, there would be Commercial Off The Shelf (COTS) products available to parse and manipulate the XML data, eliminating the need to develop these products inhouse. It should also ease the pain of maintaining all the interfaces.

Another approach is to build "mediators" between the systems. Mediators would use metadata that is sent with the data to understand the syntax and semantics of the data and convert it into a format usable by the receiving system. However, mediators do require that well formed metadata be sent, adding to the complexity of the interface.

Software Tiers

Typically, software architectures are either 2-tier or 3-tier.[4] Each tier typically presents at least one capability.

Two-Tier

In a two-tier architecture, the user interface and business logic are tightly coupled while the data is kept independent. This gives the advantage of allowing the data to reside on a dedicated data server. It also allows the data to be independently maintained. The tight coupling of the user interface and business logic assure that they will work well together, *for this problem in this domain*. However, the tight coupling of the user interface and business logic dramatically increases maintainability risks while reducing flexibility and opportunities for reuse.

Three-Tier

A 3-tier approach adds a tier that separates the business logic from the user interface. This in principle allows the business logic to be used with different user interfaces as well as with different data stores. With respect to the use of different user interfaces, users might want the same user interface but using different COTS presentation servers, for example, Java Virtual Machine (JVM) or Common Desktop Environment (CDE)[5]. Similarly, if the business logic is to be used with different data stores, then each data store must use the same data model[6] ("data standardization"), or a mediation tier must be added above the data store ("data encapsulation").

Five-Tier

To achieve maximum flexibility, software should utilize a 5-tier scheme for software which extends the three-tier paradigm (see Figure 1). The scheme is intended to provide strong separation of the three major functional areas of the architecture. Since there are client and server aspects of both the user interface and the data store, the scheme then has 5 tiers.[7]

The presentation tier is typically COTS-based. The presentation interface might be an X-server, Win32, etc. There should be a separate tier for the user interface client. This client establishes the look and feel of the interface; the server (presentation tier) actually performs the tasks by manipulating the display. The user interface client hides the presentation server from the application business logic.

The application business logic, e.g., a scheduling engine, should be a separate tier. This tier is called the “application logic” and functions as a server for the user interface client. It interfaces to the user interface typically through callbacks. The application logic tier also functions as a client to the data access tier.

If there is a user need to use an application with multiple databases with different schema, then a separate tier is needed for data access. This client would access the data stores using the appropriate COTS interface[8] and then convert the raw data into an abstract data type representing parts of the information model. The interface into this object network would then provide a generalized data access interface (DAI) which would hide the storage details of the data from any application that uses that data.

Each tier in this scheme can have zero or more components. The organization of the components within a tier is flexible and can reflect a number of different architectures based on need. For example, there might be many different components in the application logic tier (scheduling, accounting, inventory control, …) and the relationship between them can reflect whatever architecture makes sense, but none of them should be a client to the presentation server.

This clean separation of user interface, business logic, and information will result in maximum flexibility and componentized software that lends itself to product line development practices. For example, it is conceivable that the same functionality should be built once and yet be usable by different presentation servers (e.g., on PCs or UNIX boxes), displayed with different looks and feels depending on user needs, and usable with multiple legacy databases. Moreover, this flexibility should not require massive rewrites to the software whenever a change is needed.

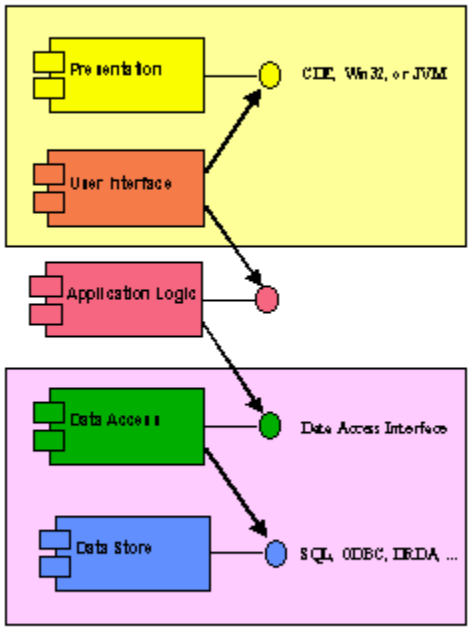


Figure 2: The 5-Tier Organization

Some Uses of a Data Access Tier

The data access tier provides a standardized view of certain classes of data, and as such functions as a server to one or more application logic tiers. If implemented correctly, there would be no need for application code to “know” about the implementation details of the data. The application code would only have to know about an interface that presents a level of abstraction higher than the data. This interface is called the Data Access Interface (DAI).

For example, should a scheduling engine need to know what events are scheduled between two dates, that query should not require knowledge of tables and joins in a relational database. Moreover, the DAI could provide standardized access techniques for the data. For example, the DAI could provide a Publish and Subscribe (P&S) interface whereby systems which require access to data stores could register an interest in certain types of data, perhaps under certain conditions, and the DAI would provide the required data when those conditions occur.

One Possible Instantiation of a Data Access Interface

One means to instantiate a data access component is with three layers, as is shown in Figure 3. This is not the only means to build a DAI, but is presented as a possibility.

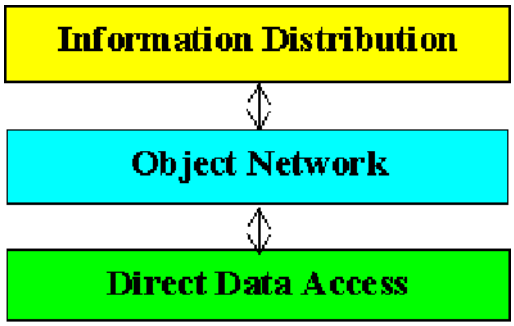


Figure 3: The DAI

Whereas the Direct Data Access layer contains the implementation details of one or more specific data stores, the Object Network and the Information Distribution layer require no such knowledge. Instead, the upper two layers reflect the need to standardize the interface for a particular domain. The Direct Data Access layer spans the gap between the Data Access tier and the Data Store tier, and therefore has knowledge of the implementation details of the data. SQL statements, either embedded or via a standard such as DRDA or ODBC, are located here.

The Object Network layer is the instantiation in software of the information model. As such, it is an efficient means to show the relationships that hold between pieces of data. The translation of data accesses to objects in the network would be the role of the Direct Data Access layer.

Within the Information Distribution layer lies the interface to the "outside world." This interface typically uses a data bus to distribute the data (see below)[9]. It could also contain various information-related services, for example, a P&S registry and publication service or an interface to a security server for data access control[10]. The Information Distribution layer might also be used to distribute applications or applets required to process distributed information. Objects in the object network would point to the applications or applets, allowing easy access to required processing code.

DAIs Enable Flexibility

The DAI enables a very flexible architecture. Multiple raw capabilities can access the same or different data stores all through the same DAI. Each DAI might be implemented in many ways, according to the specific needs of the raw capabilities using it. Figure 4 illustrates a number of possibilities, including multiple different DAIs in different domains accessing the same database, a single DAI accessing multiple databases, and multiple instantiations of the same DAI access the same database.

It is not always clear that a DAI is needed, and it appears to require additional work during all phases of development. However, should a database ever be redesigned, or if an application is to be reused and there is no control over how the new data is implemented, using a DAI saves time in the long run.

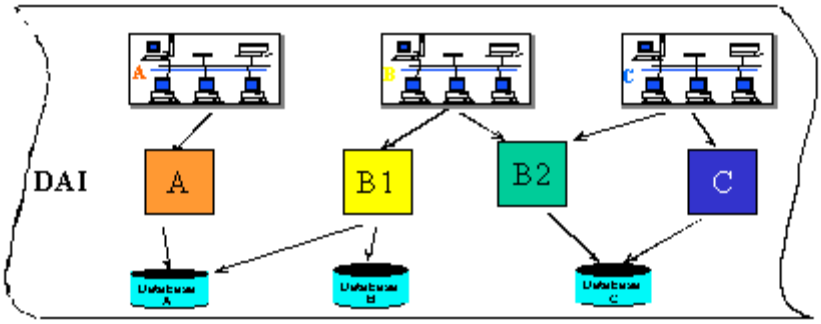


Figure 4: Multiple Uses of a DAI

Distribution

The International Standards Organization Reference Model for Open Distributed Processing (RM-ODP)[11] offers a meta-standard that is intended to allow more specific standards to emerge. This Reference Model defines a set of distribution transparencies that are applicable to the TOGAF Software View.

Transparency	Definition
Access	Masks differences in data representation and invocation mechanisms to enable interworking between objects. This transparency solves many of the problems of interworking between heterogeneous systems, and will generally be provided by default.
Failure	Masks from an object the failure and possible recovery of other objects (or itself) to enable fault tolerance. When this transparency is provided, the designer can work in an idealized world in which the corresponding class of failures does not occur.
Location	Masks the use of information about location in space when identifying and binding to interfaces. This transparency provides a logical view of naming, independent of actual physical location.
Migration	Masks from an object the ability of a system to change the location of that object. Migration is often used to achieve load balancing and reduce latency.
Relocation	Masks relocation of an interface from other interfaces bound to it. Relocation allows system operation to continue even when migration or replacement of some objects creates temporary inconsistencies in the view seen by their users.
Replication	Masks the use of a group of mutually behaviorally compatible objects to support an interface. Replication is often used to enhance performance and availability.
Transaction	Masks coordination of activities amongst a configuration of objects to achieve consistency.

Table 1: RM-ODP Distribution Transparencies

The Infrastructure Bus

The Infrastructure Bus represents the middleware that establishes the client/server relationship. This commercial software is like a backplane onto which one can plug capabilities. A system should adhere to a commercial implementation of a middleware standard. This is to ensure that capabilities using different commercial implementations of the standard can interoperate. If more than one commercial standard is used (e.g., COM and CORBA), then the system should allow for interoperability between implementations of these standards via the use of commercial bridging software.[12] Wherever practical, the interfaces should be specified in the appropriate Interface Description Language (IDL). Taken this way, every interface in the 5-tier scheme represents an opportunity for distribution.
Clients can interact with servers via the Infrastructure Bus. In this interaction, the actual network transport (TCP/IP, HTTP, etc.), the platform/vendor of the server, and the operating system of server are all transparent.

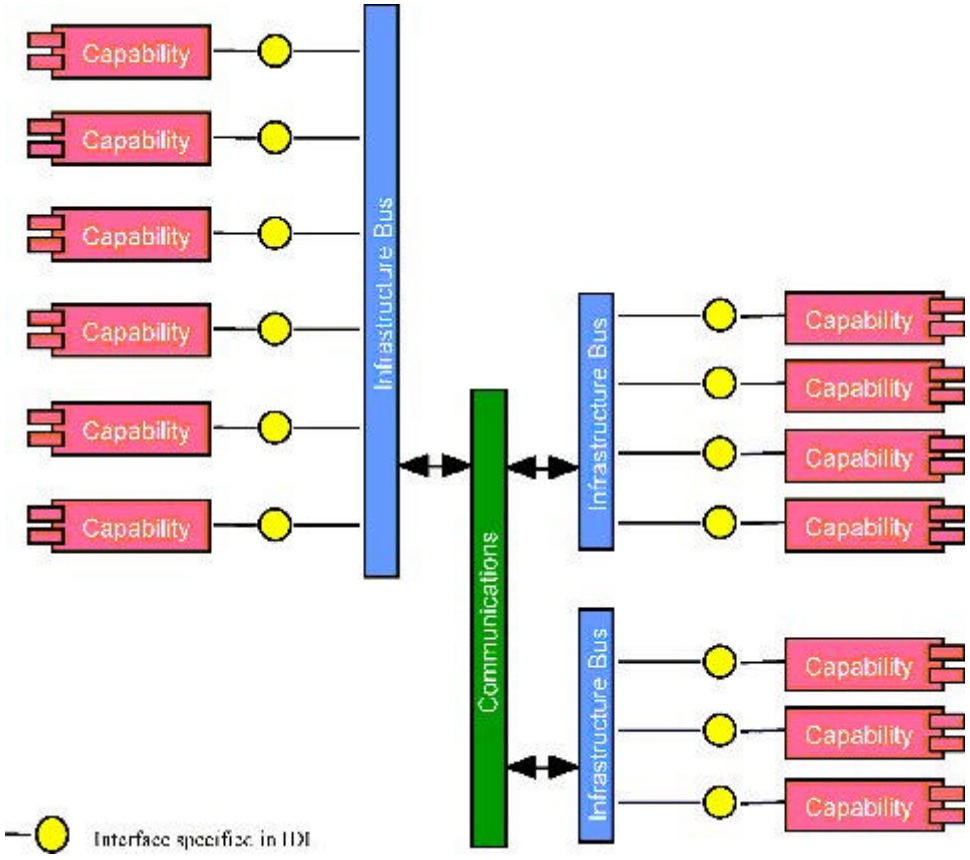


Figure 5: Notional Distribution Model

Conclusion

The Software Engineering View gives guidance on how to structure software in a very flexible manner. By following these guidelines, the resulting software will be componentized. This enables the reuse of components in different environments. Moreover, through the use of an infrastructure bus and clean interfaces, the resulting software will be location independent, enabling its distribution across a network.

---

[1] Some of the material in this document is from "The Command and Control System Target Architecture (C2STA)" which was developed by the Electronic Systems Center (ESC) of the United States Air Force between 1997 and 2000.

[2] The word "interoperate" implies that one processing system performs an operation on behalf of or at the behest of, another processing system. In practice the request is a complete sentence containing a verb (operation) and one or more nouns (identities of resources - where the resources can be information, data, physical devices, etc.). Interoperability comes from shared functionality.

[3] At usable Ethernet speeds (usually about 4 mb/s), it takes about 33 minutes to transfer a 1GB file. Today, many databases are considerably larger

than 1GB, and the fastest way to transfer these extremely large databases might well be to put them on CDs and send them by an overnight courier.

[4] These are different from 2 and 3 tiered system architectures in which the middle tier is usually middleware. In the approach being presented here, middleware is seen as an enabler for the software components to interact with each other. See the section below on the Infrastructure Bus for more details.

[5] This allows for the same user interface to be run on PCs, workstations, and mainframes, for example.

[6] If, for example, SQL statements are to be embedded in the business logic.

[7] Note that typical "layered" architectures require each layer to be a client of the layer below it and a server to the layer above it. The scheme presented here is not compliant with this description and therefore we have used the word "tier" instead of "layer".

[8] The interface to the data store might utilize embedded SQL. A more flexible way would be to use the Distributed Relational Database Architecture (DRDA) or ODBC since either of these standards would enable an application to access different DBMS' in a location-independent manner using the same SQL statements.

[9] Although it could use other mechanisms. For example, the DAI could be built as a shared library to be linked with the application logic at compile time.

[10] The security server itself would use a 5 tier architecture. The security application logic tier would interface with the DAI of other systems to provide data access control.

[11] See http://www-cs.open.ac.uk/~m_newton/odissey/transparency.html

[12] For example, many people believe that the user interface should be built on COM while the data access tiers should be built on CORBA.

Developing a System Engineering View

Stakeholder and Concerns Key Issues Introduction Client/Server Master/Slave Peer-to-Peer Distributed Object Management

Stakeholder and Concerns

This view should be developed for the systems engineering personnel of the system, and should focus on how the system is implemented from the perspective of hardware/software and networking.

Systems engineers are typically concerned with location, modifiability, reusability and availability of all components of the system. The System Engineering View presents a number of different ways in which software and hardware components can be assembled into a working system. To a great extent the choice of model determines the properties of the final system. It looks at technology which already exists in the organization, and what is available currently or in the near future. This reveals areas where new technology can contribute to the function or efficiency of the new architecture, and how different types of processing platform can support different parts of the overall system.

Major concerns for this view are understanding the system requirements. In general these stakeholders are concerned with assuring that the appropriate components are developed and deployed within the system in an optimal manner.

Developing this view assists in the selection of the best configurations for the system.

Key Issues

This view of the architecture focuses on computing models that are appropriate for a distributed computing environment. To support the migration of legacy systems, this section also presents models that are appropriate for a centralized environment. The definitions of many of the computing models (e.g., host-based, master-slave, and three-tiered) historically preceded the definition of the client/server model, which attempts to be a general-purpose model. In most cases the models have not been redefined in the computing literature in terms of contrasts with the client/server model. Therefore, some of the distinctions of features are not always clean. In general, however, the models are distinguished by the allocation of functions for an information system application to various components (e.g., terminals, computer platforms). These functions that make up an information system application are presentation, application function, and data management.

Client/Server Model

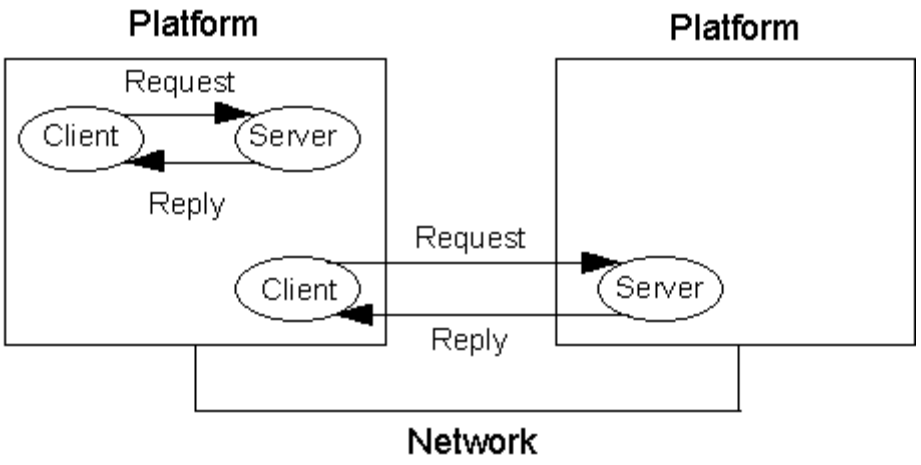


Figure 1: Basic Client/Server Model

Client/server processing is a special type of distributed computing termed co-operative processing because the clients and servers co-operate in the processing of a total application (presentation, functional processing, data management). In the model, clients are processes that request services, and servers are processes that provide services. Clients and servers can be located on the same processor, different multiprocessor nodes, or on separate processors at remote locations. The client typically initiates communications with the server. The server typically does not initiate a request with a client. A server may support many clients and may act as a client to another server. Figure 1 depicts a basic client/server model, which emphasizes the request-reply relationships. Figure 2 shows the same model drawn following The Open Group Technical Reference Model, showing how the various entities and interfaces can be used to support a client/server model, whether the server is local or remote to the client. In these representations, the request-reply relationships would be defined in the API.

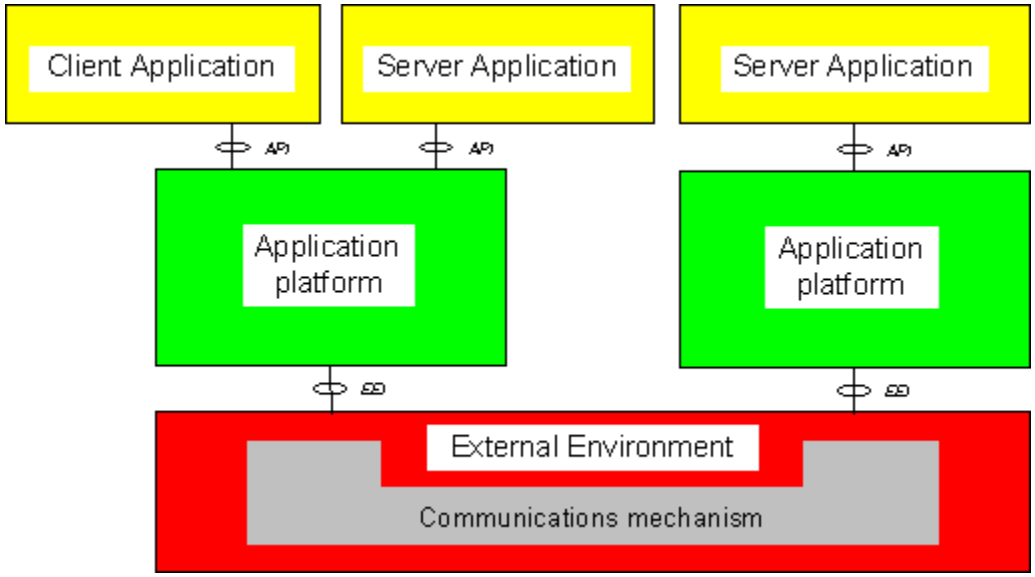


Figure 2: Reference Model Representation of Client/Server Model

Clients tend to be generalized and can run on one of many nodes. Servers tend to be specialized and run on a few nodes. Clients are typically implemented as a call to a routine. Servers are typically implemented as a continuous process waiting for service requests (from clients). Many client/server implementations involve remote communications across a network. However, nothing in the client/server model dictates remote communications, and the physical location of clients is usually transparent to the server. The communication between a client and a server may involve a local communication between two independent processes on the same machine.

An application program can be considered to consist of three parts:

- data handling
- application function
- presentation

In general, each of these can be assigned to either a client or server application, making appropriate use of platform services. This assignment defines a specific client/server configuration.

Master/Slave and Hierarchic Models

In this model, slave computers are attached to a master computer. In terms of distribution, the master/slave model is one step up from the host-based model. Distribution is provided in one direction-from the master to the slaves. The slave computers perform application processing only when directed to by the master computer. In addition, slave processors can perform limited local processing, such as editing, function key processing, and field validation. A typical configuration might be a mainframe as the master with personal computers (PCs) as the slaves acting as intelligent terminals, as illustrated in Figure 3.

The hierarchic model is an extension of the master-slave model with more distribution capabilities. In this approach, the top layer is usually a powerful mainframe, which acts as a server to the second tier. The second layer consists of LAN servers and clients to the first layer as well as servers to the third layer. The third layer consists of PCs and workstations. This model has been described as adding true distributed processing to the master-slave model. Figure 3 shows an example hierarchic model in the third configuration, and below, Figure 4 shows the hierarchic model represented in terms of the entities and interfaces of the Technical Reference Model.

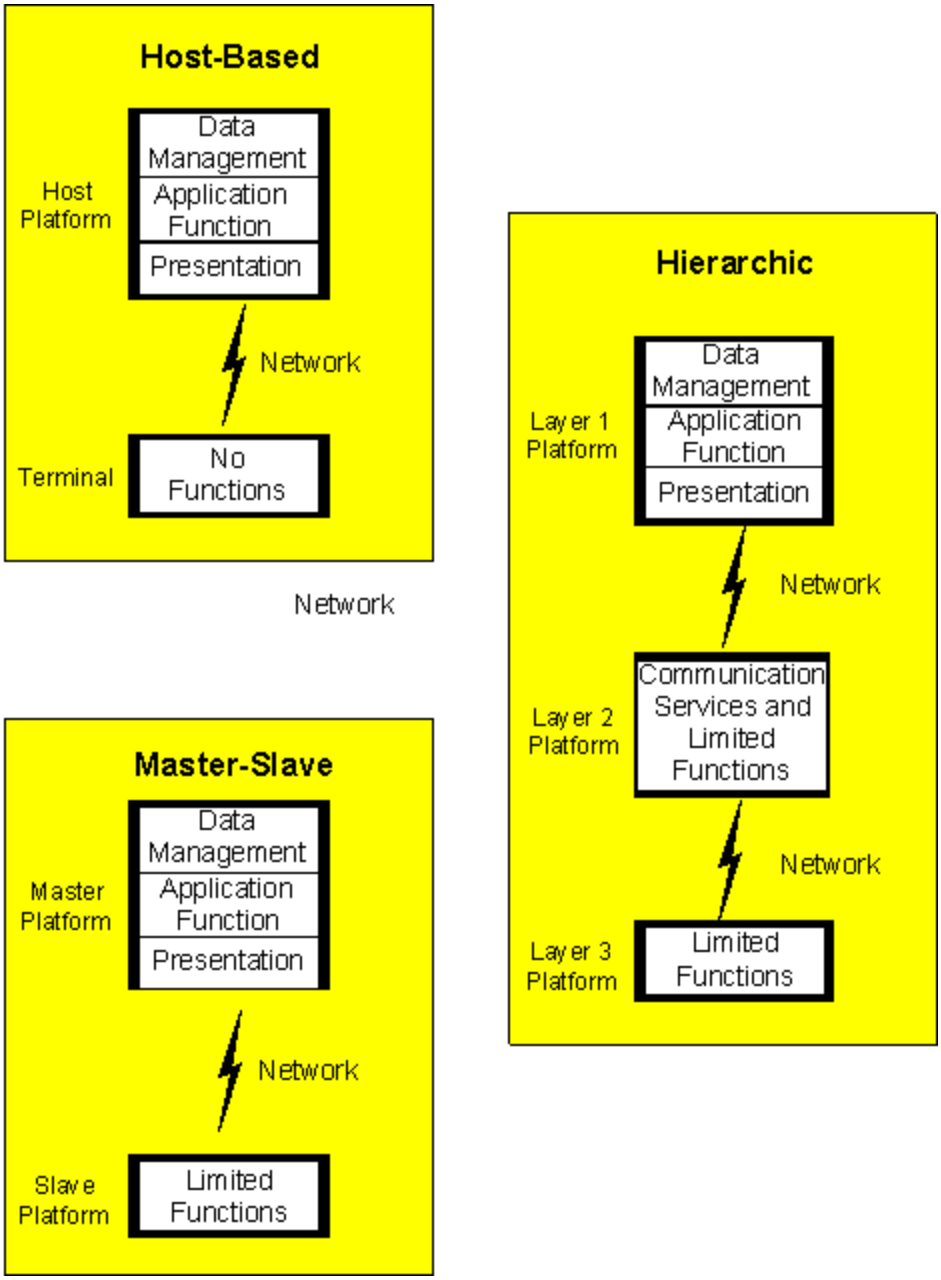


Figure 3: Host-Based, Master-Slave, and Hierarchic Models

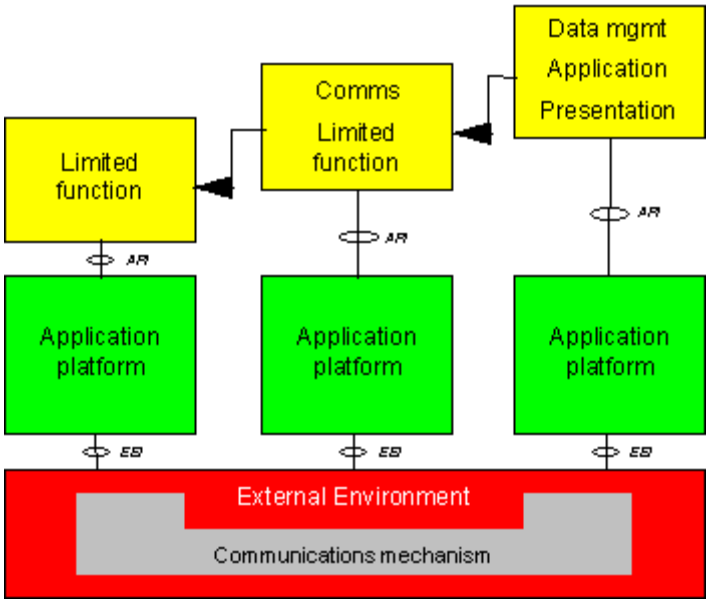


Figure 4: Hierarchic Model Using the Reference Model

Peer-to-Peer Model

In the peer-to-peer model there are co-ordinating processes. All of the computers are servers in that they can receive requests for services and respond to them; and all of the computers are clients in that they can send requests for services to other computers. In current implementations, there often are redundant functions on the participating platforms.

Attempts have been made to implement the model for distributed heterogeneous (or federated) database systems. This model could be considered a special case of the client/server model, in which all platforms are both servers and clients. Figure 5 (A) shows an example peer-to-peer configuration in which all platforms have complete functions.

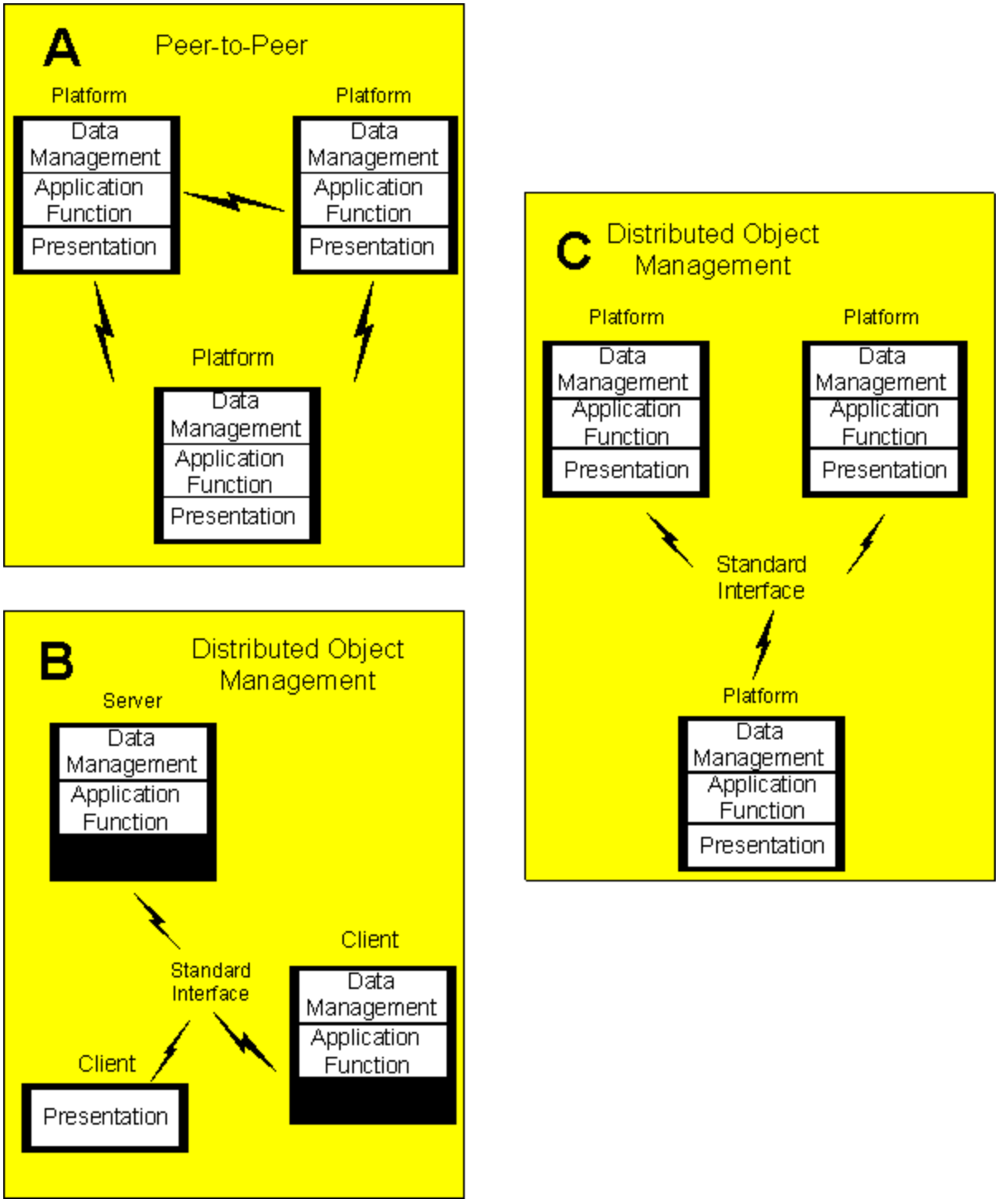


Figure 5: Peer-to-Peer and Distributed Object Management Models

Distributed Object Management Model

In this model the remote procedure calls typically used for communication in the client/server and other distributed processing models are replaced by messages sent to objects. The services provided by systems on a network are treated as objects. A requester need not know the details of how the object is configured. The approach requires:

- a mechanism to dispatch messages;
- a mechanism to co-ordinate delivery of messages; and
- applications and services that support a messaging interface.

This approach does not contrast with client/server or peer-to-peer models but specifies a consistent interface for communicating between co-operating platforms. It is considered by some as an implementation approach for client/server

and peer-to-peer models. Figure 5 presents two distributed object model examples. Example B shows how a client/server configuration would be altered to accommodate the distributed object management model. Example C shows how a peer-to-peer model would be altered to accomplish distributed object management.

The Object Management Group (OMG), a consortium of industry participants working toward object standards, has developed an architecture - the Common Object Request Broker Architecture (CORBA), which specifies the protocol a client application must use to communicate with an Object Request Broker (ORB), which provides services. The ORB specifies how objects can transparently make requests and receive responses. In addition, Microsoft's Object Linking and Embedding (OLE) standard for Windows is an example of an implementation of distributed object management, whereby any OLE-compatible application can work with data from any other OLE-compatible application.

Developing a Communications Engineering View

Stakeholder and Concerns Key Issues Infrastructure Models

Stakeholder and Concerns

This view should be developed for the communications engineering personnel of the system, and should focus on how the system is implemented from the perspective of the communications engineer.

Communications engineers are typically concerned with location, modifiability, reusability and availability of communications and networking services. Major concerns for this view are understanding the network and communications requirements. In general these stakeholders are concerned with assuring that the appropriate communications and networking services are developed and deployed within the system in an optimal manner.

Developing this view assists in the selection of the best model of communications for the system.

Key Issues

Communications networks are constructed of end devices (e.g., printers), processing nodes, communication nodes (switching elements), and the linking media that connect them. The communications network provides the means by which information is exchanged. Forms of information include data, imagery, voice, and video. Because automated information systems accept and process information using digital data formats rather than analogue formats, the TOGAF communications concepts and guidance will focus on digital networks and digital services. Integrated multimedia services are included.

The communications engineering view describes the communications architecture with respect to geography, discusses the Open Systems Interconnection (OSI) reference model, and describes a general framework intended to permit effective system analysis and planning.

Communications Infrastructure

The communications infrastructure may contain up to three levels of transport - local, regional/metropolitan, and global, as shown in Figure 1. The names of the transport components are based on their respective geographic extent, but there is also a hierarchical relationship among them. The transport components correspond to a network management structure in which management and control of network resources are distributed across the different levels.

The local components relate to assets that are located relatively close together geographically. This component contains fixed communications equipment and small units of mobile communications equipment. Local area networks (LANs), to which the majority of end devices will be connected, are included in this component. Standard interfaces will facilitate portability, flexibility, and interoperability of LANs and end devices.

Regional and metropolitan area networks (MANs) are geographically dispersed over a large area. A regional or metropolitan network could connect local components at several fixed bases or connect separate remote outposts. In most cases, regional and metropolitan networks are used to connect local networks. However, shared databases, regional processing platforms, and network management centers may connect directly or through a LAN. Standard interfaces will be provided to connect local networks and end devices.

Global or wide area networks (WANs) are located throughout the world, providing connectivity for regional and metropolitan networks in the fixed and deployed environment. In addition, mobile units, shared databases, and central processing centers can connect directly to the global network as required. Standard interfaces will be provided to connect regional and metropolitan networks and end devices.

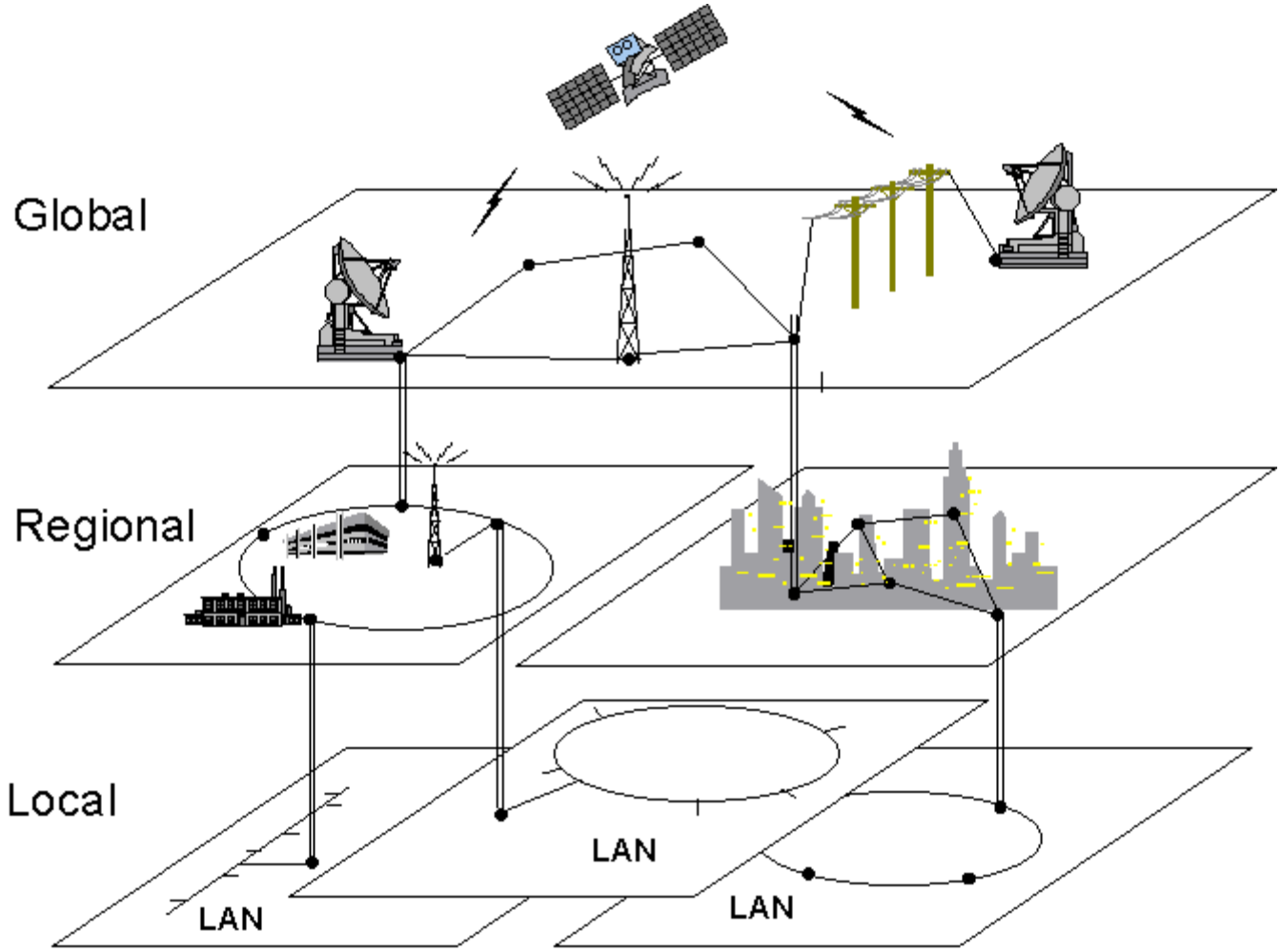


Figure 1: Communications Infrastructure

Communications Models

The geographically divided infrastructure described above forms the foundation for an overall communications framework. These geographic divisions permit the separate application of different management responsibilities, planning efforts, operational functions, and enabling technologies to be applied within each area. Hardware and software components and services fitted to the framework form the complete model.

The following sections describe the OSI reference model and a grouping of the OSI layers that facilitates discussion of interoperability issues.

The OSI Reference Model

The Open Systems Interconnection (OSI) reference model, portrayed in Figure 2, is the model used for data communications in TOGAF. Each of the seven layers in the model represents one or more services or protocols (a set of rules governing communications between systems), which define the functional operation of the communications between user and network elements. Each layer (with the exception of the top layer) provides services for the layer above it. This model aims at establishing open systems operation and implies standards-based implementation. It strives to permit different systems to accomplish complete interoperability and quality of operation throughout the network.

The seven layers of the OSI model are structured to facilitate independent development within each layer and to provide for changes independent of other layers. Stable international standard protocols in conformance with the OSI reference model layer definitions have been published by various standards organizations. This is not to say that the only protocols which fit into TOGAF are OSI protocols. Other protocol standards such as SNA or TCP/IP can be described using the OSI seven layer model as a reference.

Support and business-area applications, as defined in TOGAF, are above the OSI Reference Model protocol stack and use its services via the applications layer.

A communications system based on the OSI reference model includes services in all the relevant layers, the support and business-area application software which sits above the Application layer of the OSI model, and the physical equipment carrying the data. These elements may be grouped into architectural levels that represent major functional capabilities, such as switching and routing, data transfer, and the performance of applications.

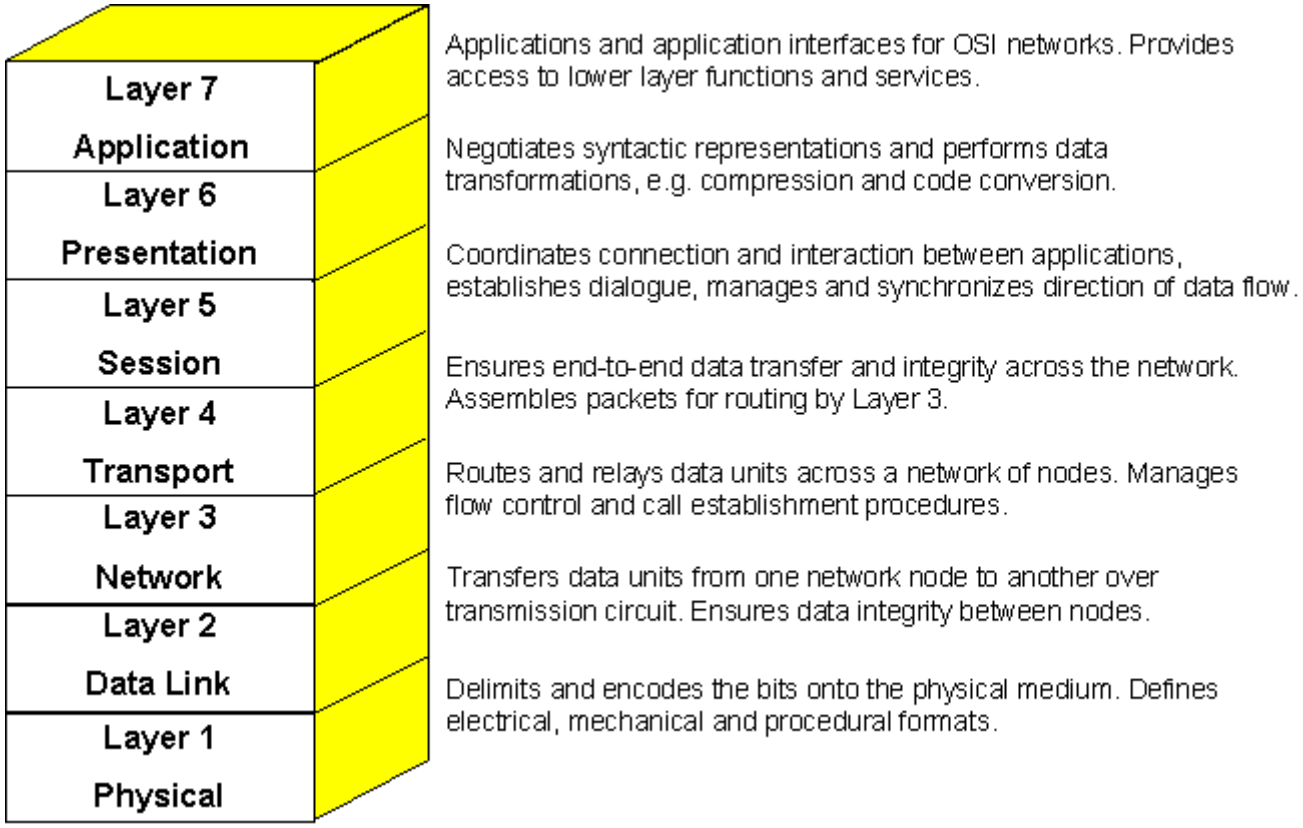


Figure 2: Open Systems Interconnection Model

These architectural levels are:

- The Transmission Level (below the physical layer of the OSI model) provides all of the physical and electronic capabilities, which establish a transmission path between functional system elements (wires, leased circuits, interconnects, etc.).
- The Network Switching Level (OSI layers 1 through 3) establishes connectivity through the network elements to support the routing and control of traffic (switches, controllers, network software, etc.).
- The Data Exchange Level (OSI layers 4 through 7) accomplishes the transfer of information after the network has been established (end-to-end, user-to-user transfer) involving more capable processing elements (hosts, workstations, servers, etc.).
- In the TRM, OSI Application Layer Services are considered to be part of the application platform entity, since they offer standardized interfaces to the application programming entity.
- The Applications Program Level (above the OSI) includes the support and business-area applications (non-management application programs).

The communications framework is defined to consist of the three geographical components of the communications infrastructure (local, regional, and global) and the four architectural levels (transmission, network switching, data exchange, and application program), and is depicted in Figure 3. Communications services are performed at one or more of these architectural levels within the geographical components. Figure 3 shows computing elements (operating at the applications program level) with supporting data exchange elements, linked with each other through various switching elements (operating at the network switching level), each located within its respective geographical component. Figure 3 also identifies the relationship of The Open Group Architectural Framework to the communication architecture.

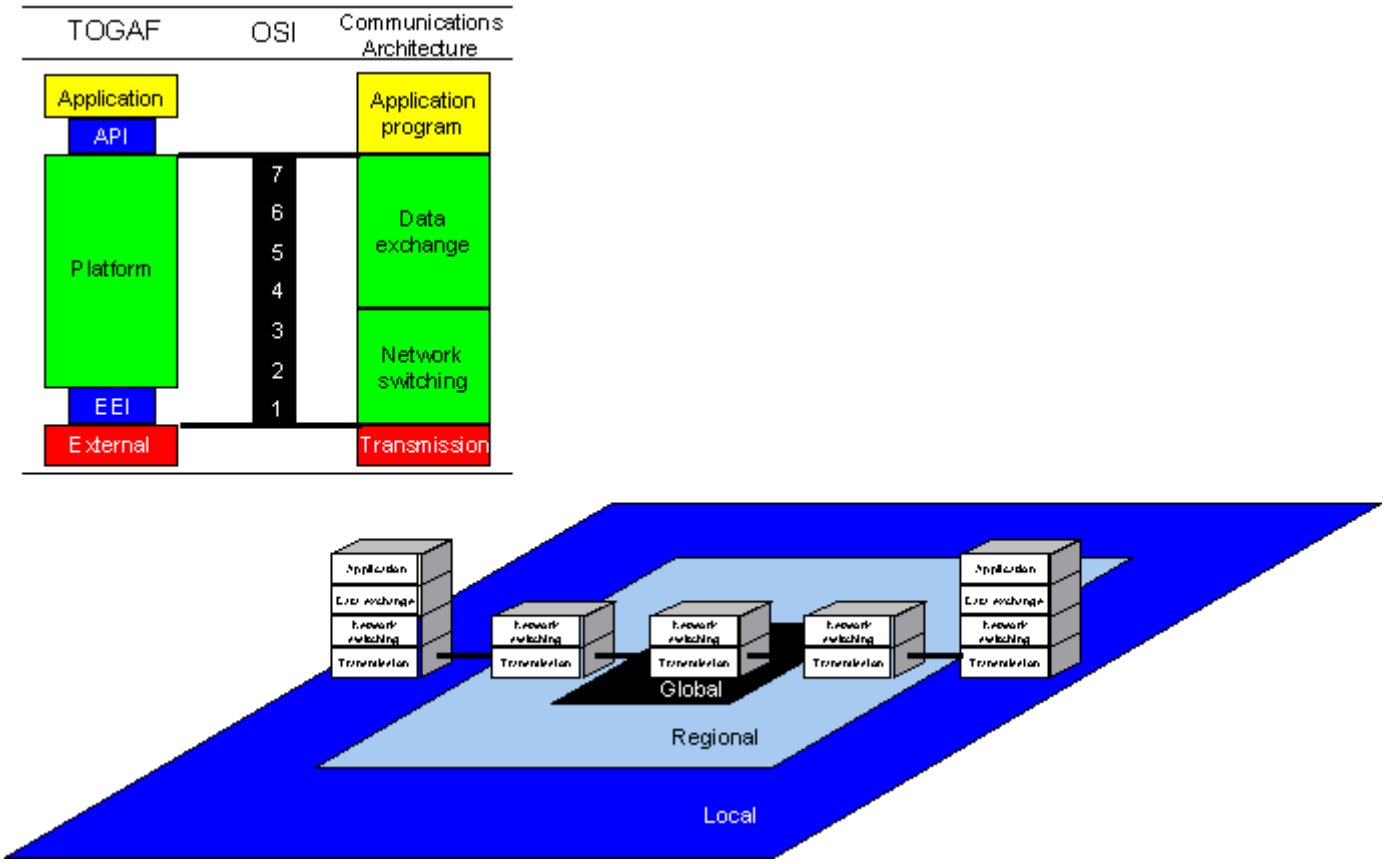


Figure 3: Communications Framework

Allocation of Services to Components

The communications infrastructure consists of the local, regional, and global transport components. The services allocated to these components are identical to the services of the application program, data exchange, network switching, or transmission architectural levels that apply to a component. Data exchange and network switching level services are identical to the services of the corresponding OSI reference model layers. Typically, only network switching and transmission services are allocated to the regional and global components, which consist of communications nodes and transmission media. All services may be performed in the local component, which includes end devices, processing nodes, communications nodes, and linking media. Transmission, switching, transport, and applications are all performed in this component.

Developing a Data Flow View

Stakeholder and Concerns

This view should be developed for database engineers of the system.

Major concerns for this view are understanding how to provide data to the right people and applications with the right interfaces at the right time. This view deals with the architecture of the storage, retrieval, processing, archiving and security of data. It looks at the flow of data as it is stored and processed, and at what components will be required to support and manage both storage and processing. In general these stakeholders are concerned with assuring ubiquitous access to high quality data.

Modeling the View

The subjects of the general architecture of a "database system" are database components or components that provide database services.

The modeling of a "database" is typically done with entity-relationship diagrams and schema definitions, including document type definitions.

Key Issues

Data management services may be provided by a wide range of implementations. Some examples are:

- Mega centers providing functionally oriented corporate databases supporting local and remote data requirements
- Distributed database management systems that support the interactive use of partitioned and partially replicated databases
- File systems provided by operating systems, which may be used by both interactive and batch processing applications.

Data management services include the storage, retrieval, manipulation, backup, restart/recovery, security, and associated functions for text, numeric data, and complex data such as documents, graphics, images, audio, and video. The operating system provides file management services, but they are considered here because many legacy databases exist as one or more files without the services provided by a DBMS.

Major components that provide data management services that are discussed in this section are:

- Database Management Systems
- Data Dictionary/Directory Systems
- Data Administration
- Data Security

These are critical aspects of data management for the following reasons. The DBMS is the most critical component of any data management capability, and a data dictionary/directory system is necessary in conjunction with the DBMS as a tool to aid the administration of the database. Data security is a necessary part of any overall policy for security in information processing.

Database Management Systems

A database management system (DBMS) provides for the systematic management of data. This data management component provides services and capabilities for defining the data, structuring the data, accessing the data, as well as security and recovery of the data. A DBMS performs the following functions:

- Structures data in a consistent way

- Provides access to the data
- Minimizes duplication
- Allows reorganization, that is, changes in data content, structure, and size
- Supports programming interfaces
- Provides security and control.

A DBMS must provide:

- Persistence- The data continues to exist after the application's execution has completed
- Secondary storage management
- Concurrency
- Recovery
- Data definition/data manipulation language (DDL/DML), which may be a graphical interface.

Database Models

The logical data model that underlies the database characterizes a DBMS. The common logical data models are listed below, and discussed in detail in the subsections that follow.

- Relational
- Hierarchical
- Network
- Object-Oriented
- Flat File

The Relational Model

A relational DBMS (RDBMS) structures data into tables that have certain properties:

- Each row in the table is distinct from every other row.
- Each row contains only atomic data; that is, there is no repeating data or such structures as arrays.
- Each column in the relational table defines named data fields or attributes.

A collection of related tables in the relational model makes up a database. The mathematical theory of relations underlies the relational model - both the organization of data and the languages that manipulate the data. Edgar Codd, then at IBM, developed the relational model in 1973. It has been popular, in terms of commercial use, since the early 1980s.

The Hierarchical Model

The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. For example, an organization might store information about an employee, such as name, employee number, department, salary. The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment. If an employee has three children, then there would be three child segments associated with one employee segment. In a hierarchical database the parent-child relationship is one to many. This restricts a child segment to having only one parent segment. Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.

The Network Model

The popularity of the network data model coincided with the popularity of the hierarchical data model. Some data were more naturally modeled with more than one parent per child. So, the network model permitted the modeling of many-to-many relationships in data. In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model. The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type. A member record type can have that role in more than one set, hence the multiparent concept is supported. An owner record type can also be a member or owner in another set. The CODASYL network model is based on mathematical set theory.

The Object-Oriented Model

An object-oriented DBMS (OODBMS) must be both a DBMS and an object-oriented system. As a DBMS it must provide the

capabilities identified above. OODBMSs typically can model tabular data, complex data, hierarchical data, and networks of data. The following are important features of an object-oriented system:

- Complex objects - e.g., objects may be composed of other objects.
- Object identity - Each object has a unique identifier external to the data.
- Encapsulation - An object consists of data and the programs (or methods) that manipulate it.
- Types or classes - A class is a collection of similar objects.
- Inheritance - Subclasses inherit data attributes and methods from classes.
- Overriding with late binding - The method particular to a subclass can override the method of a class at run time.
- Extensibility - e.g., a user may define new objects.
- Computational completeness - A general purpose language, such as Ada, C, or C++, is computationally complete. The special-purpose language SQL is not. Most OODBMSs incorporate a general-purpose programming language.

Flat Files

A flat file system is usually closely associated with a storage access method. An example is IBM's indexed sequential access method (ISAM). The models discussed earlier in this section are logical data models; flat files require the user to work with the physical layout of the data on a storage device. For example, the user must know the exact location of a data item in a record. In addition, flat files do not provide all of the services of a DBMS, such as naming of data, elimination of redundancy, and concurrency control. Further, there is no independence of the data and the application program. The application program must know the physical layout of the data.

Distributed DBMSs

A distributed DBMS manages a database that is spread over more than one platform. The database can be based on any of the data models discussed above (except the flat file). The database can be replicated, partitioned, or a combination of both. A replicated database is one in which full or partial copies of the database exist on the different platforms. A partitioned database is one in which part of the database is on one platform and parts are on other platforms. The partitioning of a database can be vertical or horizontal. A vertical partitioning puts some fields and the associated data on one platform and some fields and the associated data on another platform. For example, consider a database with the following fields: employee ID, employee name, department, number of dependents, project assigned, salary rate, tax rate. One vertical partitioning might place employee ID, number of dependents, salary rate, and tax rate on one platform and employee name, department, and project assigned on another platform. A horizontal partitioning might keep all the fields on all the platforms but distribute the records. For example, a database with 100,000 records might put the first 50,000 records on one platform and the second 50,000 records on a second platform.

Whether the distributed database is replicated or partitioned, a single DBMS manages the database. There is a single schema (description of the data in a database in terms of a data model, e.g., relational) for a distributed database. The distribution of the database is generally transparent to the user. The term "distributed DBMS" implies homogeneity.

Distributed Heterogeneous DBMSs

A distributed, heterogeneous database system is a set of independent databases, each with its own DBMS, presented to users as a single database and system. "Federated" is used synonymously with "distributed heterogeneous." The heterogeneity refers to differences in data models (e.g., network and relational), DBMSs from different suppliers, different hardware platforms or other differences. The simplest kinds of federated database systems are commonly called gateways. In a gateway, one vendor (e.g., Oracle) provides single-direction access through its DBMS to another database managed by a different vendor's DBMS (e.g., IBM's DB2). The two DBMSs need not share the same data model. For example, many RDBMS vendors provide gateways to hierarchical and network DBMSs.

There are federated database systems both on the market and in research that provide more general access to diverse DBMSs. These systems generally provide a schema integration component to integrate the schemas of the diverse databases and present them to the users as a single database, a query management component to distribute queries to the different DBMSs in the federation, and a transaction management component, to distribute and manage the changes to the various databases in the federation.

Data Dictionary/Directory Systems

The second component providing data management services, the data dictionary/directory system (DD/DS), consists of utilities and systems necessary to catalogue, document, manage, and use metadata (data about data). An example of metadata is the following definition: a 6-character long alphanumeric string, for which the first character is a letter of the alphabet and each of the remaining 5 characters is an integer between 0 and 9; the name for the string is employee ID. The DD/DS utilities make use of special files that contain the database schema. (A schema, using metadata, defines the content

and structure of a database.) This schema is represented by a set of tables resulting from the compilation of data definition language (DDL) statements. The DD/DS is normally provided as part of a DBMS but is sometimes available from alternate sources. In the management of distributed data, distribution information may also be maintained in the network directory system. In this case, the interface between the DD/DS and the network directory system would be through the API of the network services component on the platform.

In current environments, data dictionaries are usually integrated with the DBMS, and directory systems are typically limited to a single platform. Network directories are used to expand the DD/DS realms. The relationship between the DD/DS and the network directory is an intricate combination of physical and logical sources of data.

Data Administration

Data administration properly addresses the data architecture, which is outside the scope of TOGAF. We discuss it briefly here because of areas of overlap. It is concerned with all of the data resources of an enterprise, and as such there are overlaps with data management, which addresses data in databases. Two specific areas of overlap are the repository and database administration, which are discussed briefly below.

Repository

A repository is a system that manages all of the data of an enterprise, which includes data and process models and other enterprise information. Hence, the data in a repository is much more extensive than that in a DD/DS, which generally defines only the data making up a database.

Database Administration

Data administration and database administration are complementary processes. Data administration is responsible for data, data structure, and integration of data and processes. Database administration, on the other hand, includes the physical design, development, implementation, security, and maintenance of the physical databases. Database administration is responsible for managing and enforcing the enterprise's policies related to individual databases.

Data Security

The third component providing data management services is data security. This includes procedures and technology measures implemented to prevent unauthorized access, modification, use, and dissemination of data stored or processed by a computer system. Data security also includes data integrity (i.e., preserving the accuracy and validity of the data), and protecting the system from physical harm (including preventative measures and recovery procedures).

Authorization control allows only authorized users to have access to the database at the appropriate level. Guidelines and procedures can be established for accountability, levels of control, and type of control. Authorization control for database systems differs from that in traditional file systems because, in a database system, it is not uncommon for different users to have different rights to the same data. This requirement encompasses the ability to specify subsets of data and to distinguish between groups of users. In addition, decentralized control of authorizations is of particular importance for distributed systems.

Data protection is necessary to prevent unauthorized users from understanding the content of the database. Data encryption, as one of the primary methods for protecting data, is useful for both information stored on disk and for information exchanged on a network.

Developing an Enterprise Manageability View

Stakeholders and Concerns Modeling the View Key Issues

Stakeholders and Concerns

This view should be developed for the operations, administration and management personnel of the system.

Major concerns for these stakeholders are understanding how the system is managed as a whole, and how all components of the system are managed. The key concern is managing change in the system and predicting necessary preventative maintenance.

In general these stakeholders are concerned with assuring that the availability of the system does not suffer when changes occur. Managing the system includes managing components such as:

- security components
- data assets
- software assets
- hardware assets
- networking assets

Modeling the View

Business scenarios are an extremely useful way to depict what should happen when planned and unplanned events occur. It is highly recommended that business scenarios be created for planned change, and for unplanned change.

The following paragraphs describe some of the key issues that the architect might consider when constructing business scenarios.

Key Issues

The Enterprise Manageability View acts as a check and balance on the difficulties and day-to-day running costs of systems built within the new architecture. Often, system management is not considered until after all the important purchasing and development decisions have been taken, and taking a separate management view at an early stage in architecture development is one way to avoid this pitfall. It is good practice to develop the Enterprise Manageability View with close consideration of the System Engineering View, since in general management is difficult to retrofit into an existing design.

Key elements of the Enterprise Manageability View are:

- the policies, procedures and guidelines drive your management requirements. (such as a policy to restrict downloading software from the internet)
- how your shop measure system availability.
- the management services and utilities required.
- the likely quantity, quality and location of management and support personnel.
- the ability of users to take on system management tasks, such as password maintenance.

- the manageability of existing and planned components in each of the component categories.
- whether management should be centralized or distributed.
- whether security is the responsibility of system managers or a separate group, bearing in mind any legal requirements.

Key technical components categories that are the subject of the Enterprise Manageability View deal with change, either planned upgrades, or unplanned outages. The following table lists specific concerns in for each component category.

Component Category	Planned Change Considerations	Unplanned Change Considerations
Security Components	How does one propagate a security change throughout the system? Who is responsible for making changes, end users, or security stewards?…	What should happen when security is breached? What should happen if a security component fails?…
Data Assets	How does one add new data elements? How does one import/export or load/unload data? How is backup managed while running continuously? How is data change propagated in distributed environment?…	What are your backup procedures and are all the system capabilities there to backup in time?
Software Assets	How does one introduce a new application into the systems? What procedures do you have to control software quality? How does one propagate application changes in a distributed environment? How does one restrict unwanted software introduction given the internet? …	What do you want to happen when an application fails? What do you want to happen when a resource of the application fails?…
Hardware Assets	How do you assess the impact of new hardware on the system, especially network load?	What do you want to happen when hardware outages occur?
Networking Assets	How do you assess the impact of new networking components? How do you optimize your networking components?…	What do you want to happen when networking outages occur?

Developing an Acquirer's View

Stakeholders and Concerns Modeling the View Key Issues

Stakeholders and Concerns

This view should be developed for personnel involved in the acquisition of any components of the subject architecture.

Major concerns for these stakeholders are understanding what building blocks of the architecture can be bought, and what constraints (or rules) exist that are relevant to the purchase. The acquirer will shop with multiple vendors looking for the best cost solution while adhering to the constraints (or rules) applied by the architecture, such as standards.

The key concern is to make purchasing decisions that fit the architecture, and thereby to reduce the risk of added costs arising from non-compliant components.

Modeling the View

The acquirer's view is normally represented as an architecture of solution building blocks, supplemented by views of the standards to be adhered to by individual building blocks.

Key Issues

The acquirer typically executes a process similar to the one below. Within the step descriptions one can see the concerns and issues that the acquirer faces.

Procurement Process Steps	Step Description and Output
Acquisition planning	Creates the plan for the purchase of some component. For IT systems the following considerations are germane to building blocks. This step requires access to architecture and solution building blocks. § The procurer needs to know what architecture building blocks apply constraints (standards) for use in assessment and for creation of RFO/RFIs. § The procurer needs to know what candidate solution building blocks adhere to these standards. § The procurer also needs to know what suppliers provide accepted solution building blocks and where they have been deployed. § The procurer needs to know what budget this component was given relative to the overall system cost.
Concept exploration	In this step the procurer looks at the viability of the concept. Building blocks give the planner a sense of the risk involved; if many architecture or solution building blocks exist that match the concept, the risk is lower. This step requires access to architecture and solution building blocks. The planner needs to know what architecture building blocks apply constraints (standards), and needs to know what candidate solution building blocks adhere to these standards.

Concept demonstration and validation	In this step, the procurer works with development to prototype an implementation. The procurer recommends the reusable solution building blocks based upon standards fit, and past experience with suppliers. This step requires access to reusable solution building blocks.
Development	In this step the procurer works with development to manage the relationship with the vendors supplying the solution building blocks. Building blocks that are proven to be fit for purpose get marked as approved. This step requires an update of the status to "procurement approved" of a solution building block.
Production	In this step, the procurer works with development to manage the relationship with the vendors supplying the solution building blocks. Building blocks that are put into production get marked appropriately. This step requires an update of the status to "in production" of solution building blocks, with the system identifier of where the building block is being developed.
Deployment	In this step, the procurer works with development to manage the relationship with the vendors supplying the solution building blocks. Building blocks that are fully deployed get marked appropriately. This step requires an update of the status to "deployed" of solution building blocks, with the system identifier of where the building block was deployed.

Building Blocks - Overview

This whole section is intended to explain and illustrate the concept of building blocks in architecture.

Following this Overview, there are three main parts to the section.

- The first part, Introduction to Building Blocks, discusses the general concepts of building blocks, and explains the differences between Architectural and Solution Building Blocks.
- The second, Building Blocks and the ADM, summarises the stages at which building block design and specification occurs within the TOGAF Architecture Development Method.
- The third part, Building Blocks Example, comprises a series of separate subsections that together provide a detailed worked example showing how building block context is captured, how building blocks are identified, and how building blocks are defined when executing the major steps of the Architecture Development Method.

Introduction to Building Blocks

Overview

This subsection describes the characteristics of building blocks. The uses of building blocks in the ADM is described separately, under Building Blocks and the TOGAF Architecture Development Method.

Generic Characteristics of Building Blocks

Building Blocks have generic characteristics as follows:

- A Building Block is a package of functionality defined to meet the business needs across an organization
- A Building Block has published interfaces to access the functionality
- A Building Block may interoperate with other, interdependent, Building Blocks.
- A good Building Block has the following characteristics:
    - It considers implementation and usage, and evolves to exploit technology and standards
    - It may be assembled from other Building Blocks
    - It may be a subassembly of other Building Blocks
    - Ideally a Building Block is reusable and replaceable, and well specified
- A Building Block may have multiple implementations but with different interdependent Building Blocks

A Building Block is therefore simply a package of functionality defined to meet business needs. The way in which functionality, products and custom developments are assembled into building blocks will vary widely between individual architectures. Every organization must decide for itself what arrangement of building blocks works best for it. A good choice of building blocks can lead to improvements in legacy system integration, interoperability, and flexibility in the creation of new systems and applications.

Systems are built up from collections of building blocks, so most building blocks have to interoperate with other building blocks. Wherever that is true, it is important that the interfaces to a building block are published and reasonably stable.

Building blocks can be defined at various levels of detail, depending on what stage of architecture development has been reached.

For instance, at an early stage, a building block can simply consist of a grouping of functionality such as a customer database and some retrieval tools. Building blocks at this functional level of definition are described in TOGAF as *Architecture Building Blocks (ABBs)*. Later on, real products or specific custom developments replace these simple definitions of functionality, and the building blocks are then described as *Solution Building Blocks (SBBs)*.

More detail on each of these aspects of building blocks is given below.

Architectural Building Blocks

Architectural Building Blocks (ABBs) relate to the Architecture Continuum, and are defined or selected as a result of the application of the Architecture Development Method.

Characteristics

Architectural Building Blocks:

- define what functionality will be implemented
- capture business and technical requirements
- are technology aware
- direct and guide the development of Solution Building Blocks

Specification Content

- Architectural Building Block specifications include the following as a minimum:
- Fundamental Functionality and Attributes - semantic, unambiguous, including security capability and manageability
- Interfaces - chosen set, supplied (APIs, data formats, protocols, hardware interfaces, standards)
- Dependent BBs with required functionality and named used interfaces
- Map to business / organizational entities and policies

Solution Building Blocks

Solution Building Blocks (SBBs) relate to the Solutions Continuum, and may be either procured or developed.

Characteristics

Solution Building Blocks:

- define what products and components will implement the functionality
- define the implementation
- fulfil business requirements
- are product or vendor aware

Specification Content

Solution Building Block specifications include the following as a minimum:

- Specific Functionality and Attributes
- Interfaces - the implemented set
- Required Solution Building Blocks used with required functionality and names of the interfaces used
- Mapping from the Solution Building Blocks to the IT topology and operational policies
- Specifications of attributes shared across the environment (not to be confused with functionality) such as security, manageability, localizability, scalability,
- performance, configurability
- Design drivers and constraints, including the physical architecture
- Relationships between Solution Building Blocks and Architectural Building Blocks

Building Blocks and the Architecture Development Method

Principles Specification Process Levels of Modelling

Basic Principles

This subsection focuses on the use of building blocks in the ADM. General considerations and characteristics of Building Blocks are described under Introduction to Building Blocks.

Building Blocks in Architecture Design

An architecture is a set of building blocks depicted in an architectural model, and a specification of how those building blocks are connected to meet the overall requirements of an information system.

The various building blocks in an architecture specify the services required in an enterprise-specific system.

There are some general principles underlying the use of building blocks in the design of specific architectures:

- An architecture need only contain building blocks to implement those services that it requires.
- Building blocks may implement one, more than one, or only part of a service identified in the architecture framework.
- Building blocks should conform to standards relevant to the services they implement.

Building Block Design

The process of identifying building blocks includes looking for collections of functions which require integration to draw them together or make them different:

- Consider three classes of building blocks
  - re-usable building blocks such as legacy items
  - building blocks to be the subject of development such as new applications
  - building blocks to be the subject of purchase, i.e. commercial off-the-shelf applications
- Use the desired level of integration to bind or combine functions into building blocks. For instance legacy elements could be treated as large building blocks to avoid breaking them apart.

In the early stages and during views of the highest level enterprise, the Building Blocks are often kept at a broad integration definition. It is during these exercises that the services definitions can often be best viewed. As implementation considerations are addressed, more detailed views of Building Blocks can often be used to address implementation decisions, focus on the critical strategic decisions or aid in assessing the value and future impact of commonality and reusability.

Building Block Specification Process in the ADM

The process of building block definition takes place gradually as the Architecture Development Method is followed, mainly in Phases A, B, C, and D. It is an iterative process because as definition proceeds, detailed information about the functionality required, the constraints imposed on the architecture and the availability of products may affect the choice and the content of building blocks.

The key parts of the ADM at which building blocks are designed and specified are summarized below.

The major work in these steps consists of identifying the architectural building blocks (ABBs) required to meet the business goals and objectives. The selected set of ABBs is then refined in an iterative process to arrive at a set of Solution Building Blocks (SBBs) which can either be bought off the shelf or custom developed.

The specification of building blocks using the ADM is an evolutionary and iterative process. The key Phases and Steps of the

ADM at which building blocks are evolved and specified are summarized below, and illustrated in Figure 1.

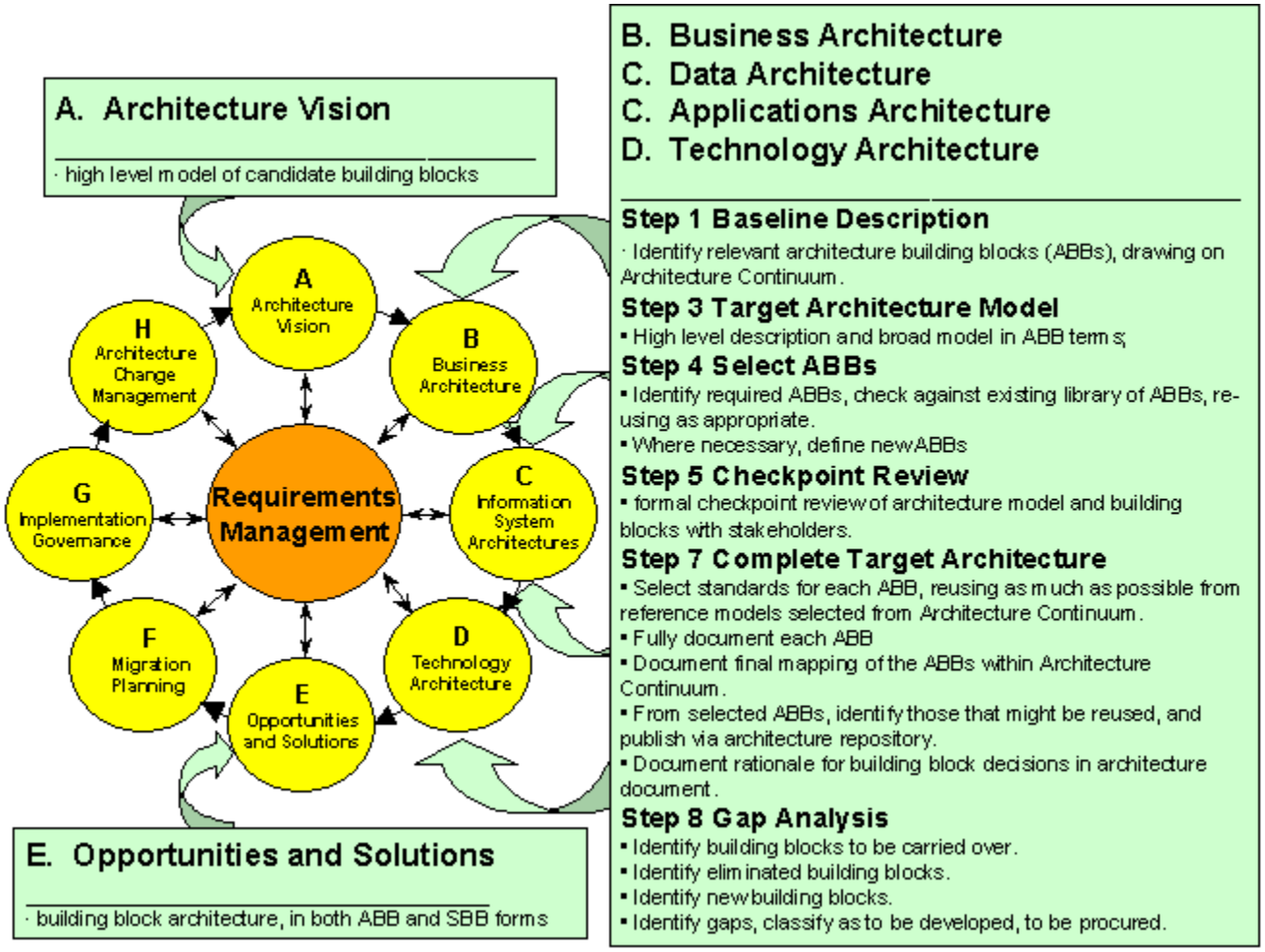


Figure 1: Key Phases / Steps of ADM at which Building Blocks are Evolved / Specified

In Phase A the earliest building block definitions start as relatively abstract entities within the Architecture Vision.

In Phases B, C, and D, building blocks within the Business, Data, Applications, and Technology Architectures are evolved to a common pattern of steps:

- Step 1, Baseline Description produces:
  - A list of candidate building blocks, from the analysis of the baseline.
- Step 3, Target Architecture Model, takes this list and high level model as inputs, and evolves them iteratively into a definition of the target architecture, specified in terms of architectural building blocks. Specifically, Step 3 produces:
  - a high level description and broad model of the target system in terms of Architectural Building Blocks.
  - a rationale for each building block decision
- Step 4 produces:
  - for each Architectural Building Block, a service description portfolio, built up as a set of non-conflicting services.
- Step 5 produces:
  - a confirmation of the merit and completeness of the model and service description portfolio, and a description of how the emerging target architecture meets the objectives of the architecture development
- Step 7 produces:
  - A target architecture, fully specified in terms of Architectural Building Blocks
  - a fully defined (by service) list of all the standards that make up the target architecture, and all the Architectural building blocks that will be used to implement it.
  - a diagrammatic depiction of the building blocks at the levels needed to describe the strategic and implementation aspects of the architecture.
- Step 8 produces:
  - a gap analysis of eliminated building blocks, carried over building blocks, and new building blocks

Finally in Phase E, Opportunities and Solutions, the building blocks become more implementation specific as Solution Building

Blocks, and their interfaces become the detailed architecture specification. The output of Phase E is the building block architecture, both in Architectural (i.e. functionally defined) and Solution (i.e. product-specific) Building Block forms.

The minimum contents of an Architectural Building Block specification and a Solution Building Block specification are described under Introduction to Building Blocks.

Levels of Modelling

Defining and developing the *context* for a set of building blocks takes place at two levels:

- The **business process** level (colored green in the diagrams in this section). This deals only at the highest level with what has to happen for a business process to be carried out.
- The **technical functionality and constraints** level (colored blue in the diagrams). This level deals with the component activities that form part of the business process and whether they can be supported or not.

Defining and developing an actual set of building blocks also takes place at two levels:

- The **architectural model** level (colored red in the diagrams). This level identifies the systems and components that will implement the technical functionality and expresses the relationships between them. This level introduces the idea of a notation to describe the architectural elements and relationships.
- The **solution model** level (colored black in the diagrams). This is the level where the individual products and/or product components that will implement the architecture are identified.

Working through the four levels is an iterative process. Figure 2 shows how considerations at any level can result in change at any or all of the other levels.

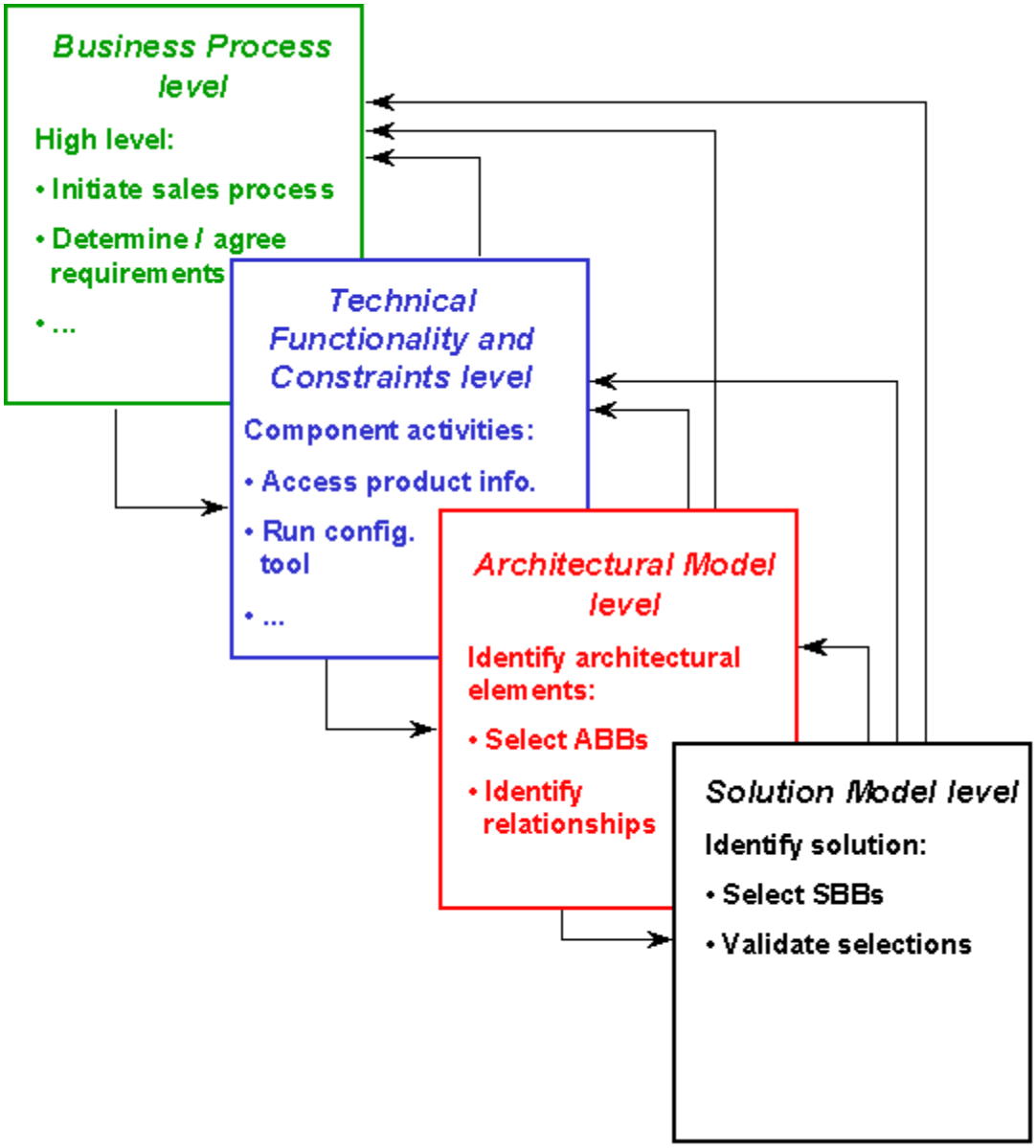


Figure 2: Iteration between the Four Levels of Modeling

Mapping the Modelling Levels to the ADM

The **business process** level of definition takes place in Phases A and B of the ADM.

The **technical functionality and constraints** level work happens early in Phases B, C, and D, once the characteristics of the current system have been established. At that stage it is possible to identify the constraints imposed on new architecture work by the legacy of the old system (the baseline).

The **architectural model** and **solution model** levels consist of work done later in Phases B, C, and D, the Target Architecture steps of each phase, and Phase E, Opportunity Identification. The architectural model work is mostly done when taking different views of the architecture, and the solution model work in Phase E, where selection of products and projects takes place.

To show how building block definition happens in practice, the remainder of this appendix consists of a fictional worked example. In this fictional example much detail has been left out in order to emphasise the process.

Building Blocks Example

Introduction Structure

Introduction

This and the following subsections provide a detailed worked example showing how building block context is captured, how building blocks are identified, and how building blocks are defined when executing the major steps of the Architecture Development Method.

Structure

The levels of modelling within the ADM are explained under Building Blocks and the ADM, and the example follows the structure of modelling explained there.

- Background to the Example.
- the Business Process level
- the Technical Functionality and Constraints level
- the Architectural Model level
- the Opportunity Identification level
- the Building Block Reuse level

Background to the example

In this example, a fictional company called XYZ Manufacturing has decided to improve the efficiency of its mobile sales force by replacing paper-based configuration and ordering systems with an IT solution.

The XYZ team have already done the preliminary stages of describing their existing system and reviewing it from a number of different viewpoints, and as a result have established a number of goals and objectives for the new system.

The principal goal is to give the sales force in the field direct access to the sales process back at base. This will allow sales staff to create and verify the product configuration, to check the price and availability of the goods and to place the order while actually with the customer.

Other stages of the sales process such as initiating the sale and determining the customer requirements are considered to be outside the scope of this example.

The Business Process level.

Phase B - the Business Process Level

The inputs to ADM Phase B are:

- Request for Architecture Work
- Statement of Architecture Work
- Business Principles, Business Goals and Strategic Drivers
- Architecture Principles
- Architecture Continuum
- Architecture Vision

The outputs of this step are:

- Statement of Architecture Work (updated if necessary)
- Validated principles, business goals, and strategic drivers
- Target Business Architecture
- Business Baseline
- Views corresponding to the selected viewpoints addressing key stakeholder concerns
- Gap analysis results
- Technical requirements (drivers for the Technical Architecture work)
- Business Architecture Report
- Updated business requirements

As a preliminary to this step it is necessary to define the scope of activity, including what is in scope, what is out of scope, what the limits are and what the financial envelope is. Within this step fall defining the business process, recording the assumptions made and developing any new requirements. The information collected is used to gauge the current system and to determine the return on investment of potential changes. Use-cases are a useful tool in this step to describe the business processes and they can be used to do a sanity check against the resulting architecture.

The business goals driving improvements in the sales process were:

- to improve the quality of the sales process
- to reduce the number of errors in the sales process
- to speed up the sales process

In this example financial and time constraints, and business return have not been dealt with in detail, but normally these constraints would be used to guide the process along the entire way to avoid over-engineering or "creeping elegance". The architect should especially look at these constraints whenever iterating between steps. Also not shown in this example are the use-cases scenarios. However, the process described below does include participants, or actors, of the use-case with brief descriptions of their roles in Table J-1.

For the sake of brevity in this example, it is assumed that the scope of the architectural work would not extend beyond the sales arena, and that the proposed solutions fit within the financial and time constraints imposed by XYZ.

The assumptions made by the XYZ architect during Phase B are:

- To support the mission of the business, it is desired to have the sales person and the customer meet and interact face to face at the customer location.
- The model for such a customer visit should be two persons interacting using a shared PC
- The sales person needs to be able to close the deal on site, and then synchronize with information held at the sales base.
- There is a clearly defined product set subject of the sales process, e.g. there is a car to buy.

The relevant business process in scope of this example in the XYZ company is the customer facing portion of the sales process and the supporting systems. This sales process consists of the following steps:

1. initiate the sales process with the customer

    1.1 sales person

1.2 customer

2. discuss the customer requirements

2.1 customer
2.2 sales person

3. work with the customer to create a product configuration

3.1 sales person
3.2 sales person's laptop
3.3 sales person's local (LIPR) and central (CIPR) information process resources
3.4 product configurator
3.5 customer

4. verify that the desired configuration can be delivered

4.1 sales person
4.2 sales person's laptop
4.3 inventory control system
4.4 scheduling system
4.5 customer accepts or rejects

5. determine the price of the requested configuration

5.1 sales person
5.2 sales person's laptop
5.3 pricing system

6. confirm the desire to purchase with the customer

6.1 sales person
6.2 customer

7. place an order

7.1 sales person
7.2 sales person's laptop with printer (for fax)
7.3 order system
7.4 customer

8. customer acceptance

8.1 sales person
8.2 customer

The following use-case table represents participants (sometimes referred to as "actors" in use-case) in the rows, steps of the business process in the columns and roles in the cells. Note that this is an example, and it is not intended to be accurate, but rather demonstrative. Constructing a use-case table is a comparatively small effort that will ultimately enhance the speed and quality of the resulting architecture.

The meanings of the various acronyms used in the table, and in subsequent figures, are listed below:

CIPR	Central Information Processing Resource
ICSys	Inventory Control System
LIPR	Local Information Processing Resource
OrdSys	Order Processing / Information System
ProdConfig	Product Configurator System
ProdSys	Product Information System
SchSys	Scheduling System

	1. initiate	***2.discuss reqs***	***3. create config***	***4. verify config***	***5. price***	***6. confirm***	***7. order***	***8. accept***
sales person	greets customer	listens	represents options with different capabilities	accesses ICSys and SchSys and presents availability to customer	accesses price system and presents price to customer	presents offer	accesses order system	presents contract
customer	accepts sales person	discusses problems /desires	listens and decides on options based on capabilities	accepts or rejects		accepts or rejects		signs or rejects
sales person's laptop			interacts with configurator	interacts with ICSys and SchSys	interacts with Price System		interacts with order system and receives fax response	
sales person's CIPR			provides central information processing					
sales person's LIPR			provides local information processing					
ProdConfig			presents configs to sales person per needs, providing capabilities					
ICSys				provides availability				
SchSys				provides delivery date				
$Sys					provides price information on a config			
OrderSys							processes order and sends fax of order to sales person's laptop	

Table J-1: Use-Case Table of Sales Process

Steps 1, 2, 6 and 8 are not within scope of the architecture work since the only participants involved are humans. The other steps are considered within scope since there are computing components involved in supporting the sales process. Note the computing participants are the first set of identified *candidate building blocks - Business Process Driven List*.

During Step A, the business goals were developed into more detailed business requirements, and these were:

- to improve on the current turnaround time of 48 hours for order processing
- to reduce the number of errors in orders by a factor of 3

A very simplified view of the candidate building blocks required to support the business process with an idea of location is provided below. This model was built from elements of the above table.

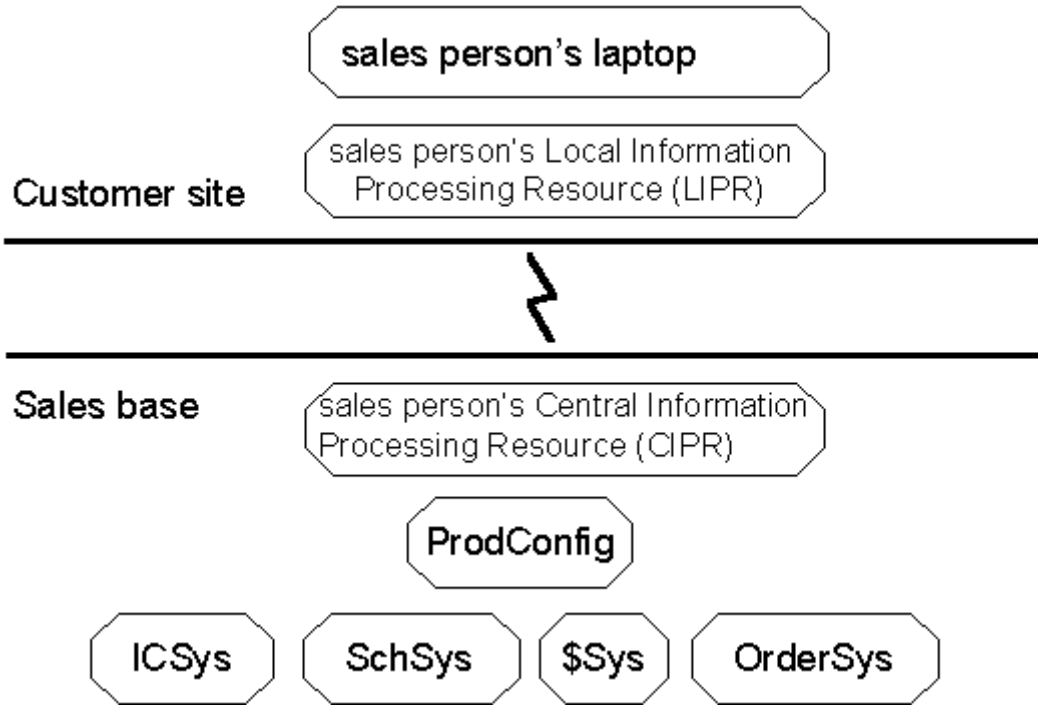


Figure J-2: Model of the Candidate Building Blocks - "Business Process Driven List"

The Technical Functionality and Constraints level

Phases B, C, and D - the Technical Functionality and Constraints Level

The objective of the first step in Phases B, C, and D of the ADM is to build a high level description of the characteristics of the current system, reusable building blocks from the current system, the technical functionality needed to address the business problem, and to identify additional constraints. This is necessary as the description documents the starting point for architectural development and lists the interoperability issues that the final architecture will have to take into account. Potential reusable building blocks may be contained in the existing environment. They are identified in this step.

The best approach is to describe the system in terms already used within the organization. A reliable picture can be built up of the business functions served and the platforms which support those functions. Gather and analyze only that information that allows informed decisions to be made regarding the target architecture.

The inputs to this step are:

- descriptions of the current system
- information on the existing architecture
- model of candidate building blocks

The essential outputs from this activity are:

- a clear description of the current system and its functions
- a statement of the constraints imposed by the internal organization
- a statement of the constraints imposed by the business or external environments
- architecture principles embodied in the current system
- assumptions of required technical functionality
- candidate building blocks - Baseline Driven List
- model of candidate building blocks (see Figure J-5)

The key input to this step is the existing architecture. In this example a depiction of an existing architecture is shown in the following figure. Additionally depicted in this architecture model are pointers to existing problems with the existing architecture. These pointers are used by the architect to determine where existing components are failing, and where existing systems can be re-used.

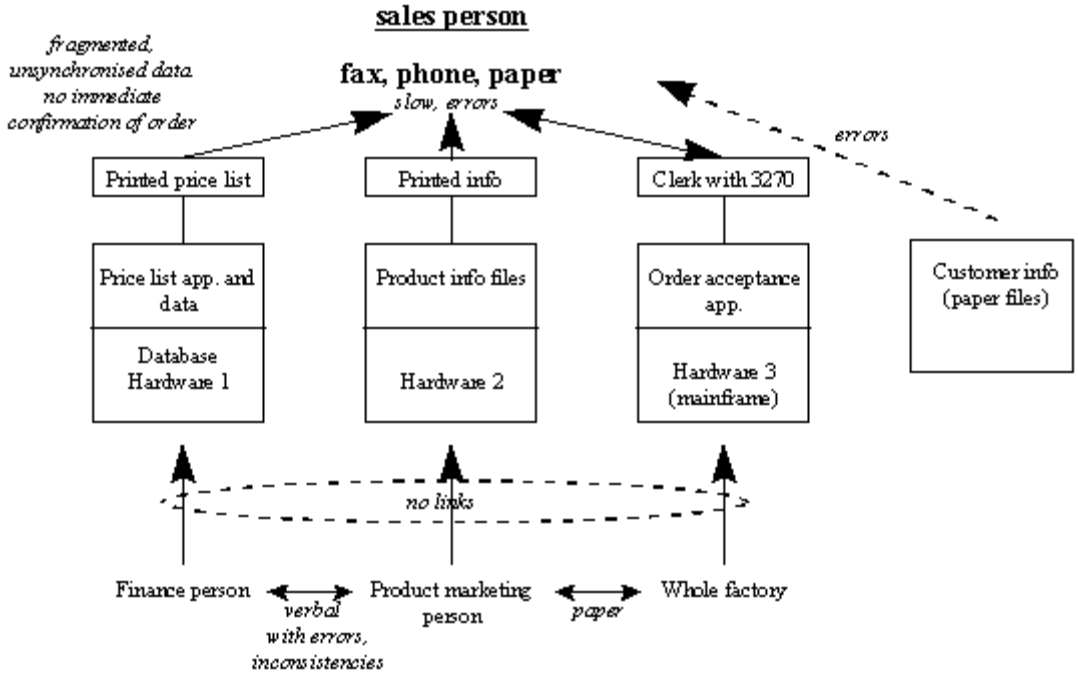


Figure J-3: XYZ's Existing Architecture

It is necessary to record existing strategic decisions about the existing architectural and technological issues such as:

- the existing architecture is founded on the mainframe
- databases are tied to application logic
- security is embedded in the application

The next step consists of restating the business process, considering what functionality will be required and deciding what constraints apply. Decisions at this stage are not definitive, but act as input for the following steps and iterations.

The architects of XYZ identified the following pieces of technical functionality as necessary to support the business processes. This list was produced using standard brainstorming techniques.

Assumptions of required technical functionality

- access to central functions
- application support for simultaneous access by multiple salespersons through multiple connections
- execution of local functions at the point of sale
- access to product information
- entering and checking the required product configuration
- access to customer information
- access to price information
- order entry
- order acceptance
- delivery of confirmation of order to the customer
- the process must be secured

Also in the brainstorming session, some assumptions were made and therefore must be documented as they should be used throughout the process:

- that initiation of the sales process and determination/agreement of the customer requirements were outside the scope of the current work
- that functionality could be distributed between the point of sale and a central base
- that closure of the order should take place at the central location
- that the price list and product information could be made available electronically
- that access could be provided to the acceptance and confirmation of order systems
- that the ordering, product information and price information systems could be linked together

One constraint was put on the development because XYZ already had systems in place to support the sales process:

- existing systems should be used to support product information, order placement and customer information

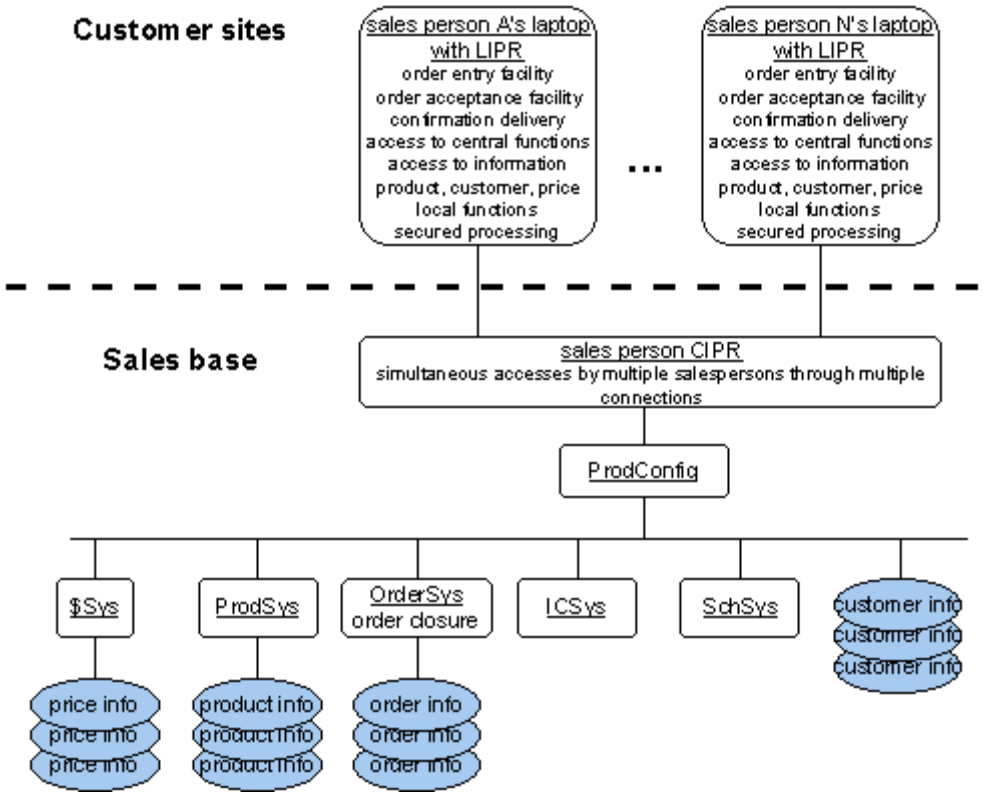


Figure J-4: Model of Candidate Building Blocks Augmented with Technical Functionality

The above model is scrutinized and questions are asked about the functionality that could be provided by the existing system. The following diagram depicts the set of candidate building blocks from the existing system, resulting from this question.

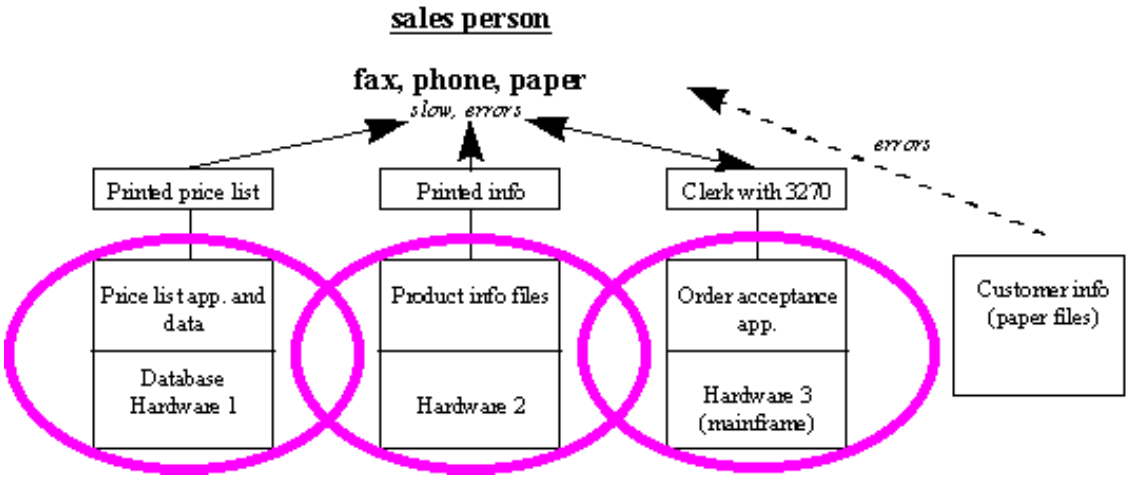


Figure J-5: Candidate Building Blocks from the "Baseline Driven List"

The Architectural Model level.

Phases B, C, and D - the Architectural Model Level

In Phases B, C, and D a number of different architectural views are considered and used to build up a model of the new architecture. At XYZ, the Architectural Model level was developed in the following steps:

1. a baseline description in the TOGAF format
2. consider different architectural views
3. create an architectural model of building blocks
4. select the services portfolio required per building block
5. confirm that the business goals and objectives are met
6. determine criteria for specification selection
7. complete the architecture definition
8. conduct a gap analysis

In executing **step 1** the existing architecture was assessed:

- describing the architectural principles of the existing architecture
- to describe the existing architecture in TOGAF terms
- to identify new requirements, inhibitors and opportunities

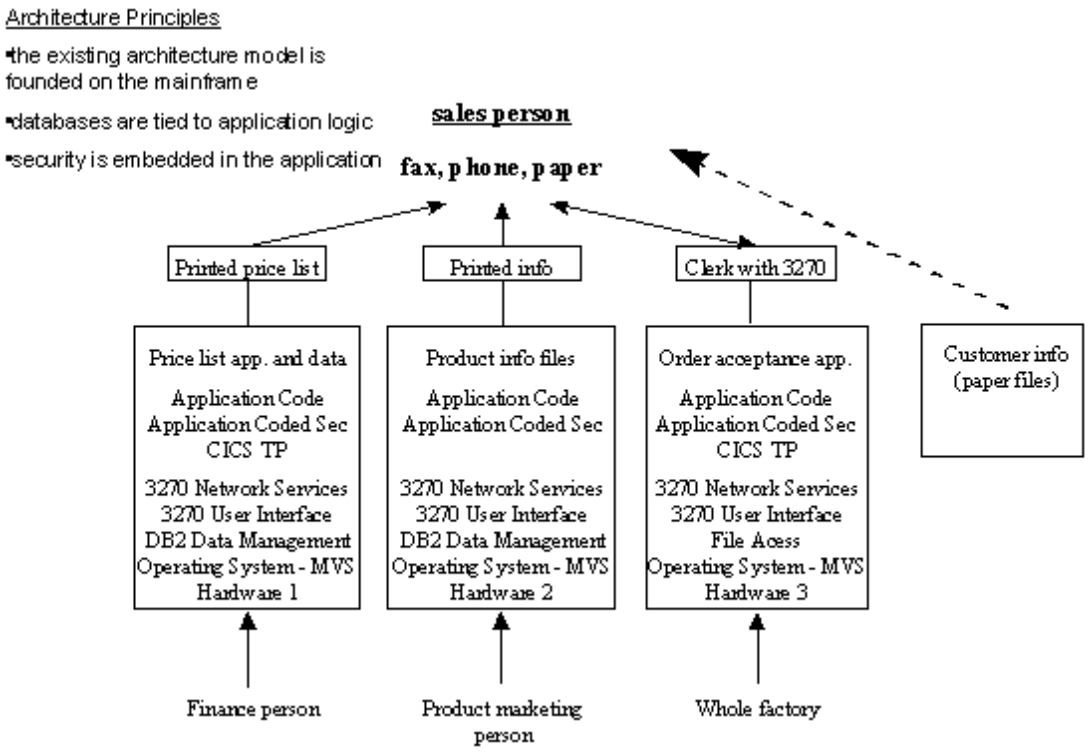


Figure J-6: Existing Architecture in TOGAF Terms

Notice how in Figures J-6 and J-7 the legacy systems supporting the Price list, Product Info and Order Acceptance applications are easy to handle as monolithic building blocks. Figures J-8, J-9 and J-10 show how they can be connected to new building blocks using adapters.

In **step 2** the function view was examined based upon what the system was intended to do, and how it should behave. The function view was depicted in the following figure. Note that the Inventory Control and Scheduling System are not covered.

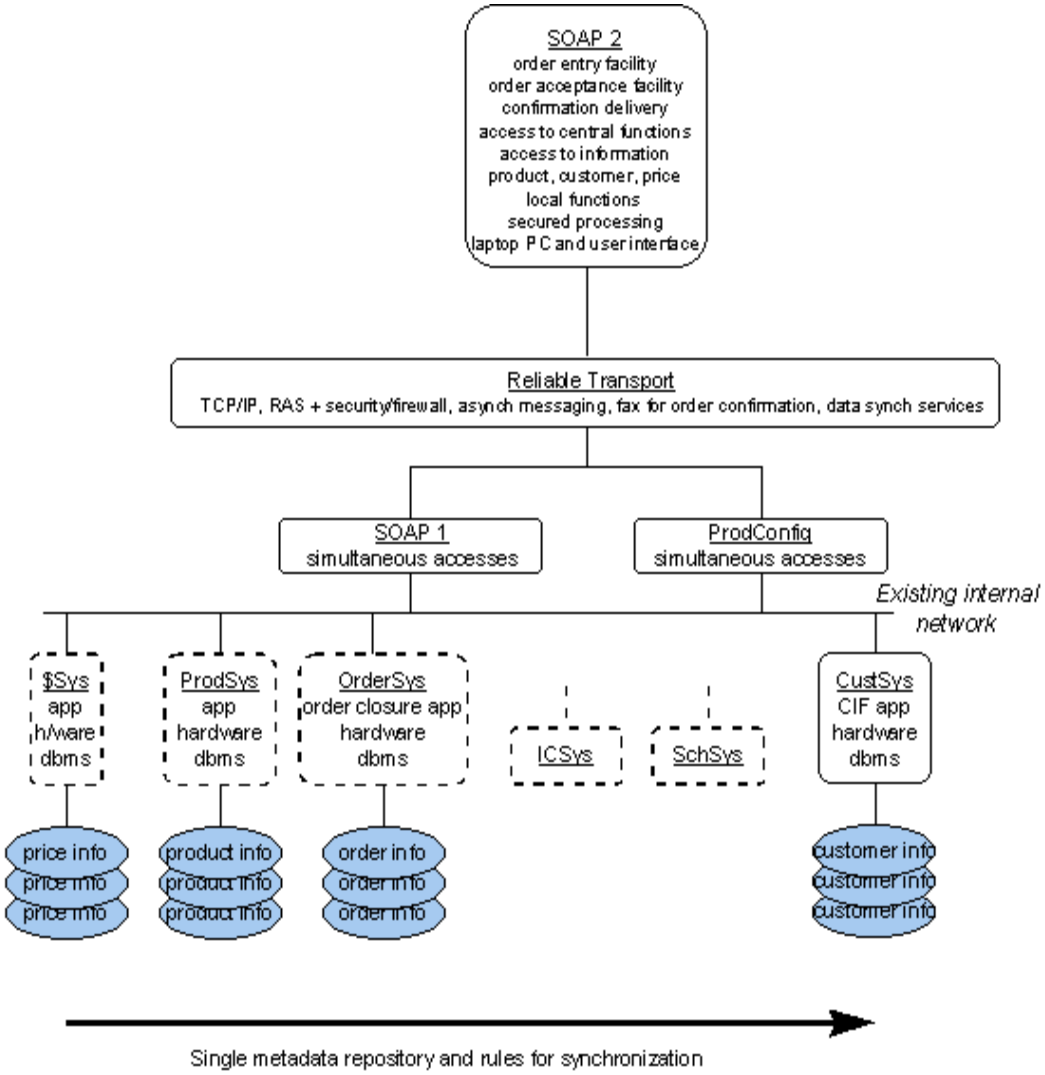


Figure J-7: Future Architecture of Functions

In executing step 2 a view of the future architecture was created by processing the technical functionality that must be provided and by:

- identifying obvious additions
- identifying what would be carried forward from the old system
- determining the return on investment for various options, allowing them to be ranked
- assessing the risk of the various changes
- checking the coverage of the technical functionality
- adding technical functionality required for completeness, checking against the TOGAF Technical Reference Model
- updating and clarifying the business requirements and technical functionality
- iteratively adding precision and detail to the future architecture
- for each architectural decision, completely following through the impact of it
- noting the rationale for each decision whether the answer was yes or no, so as to avoid reopening old issues later

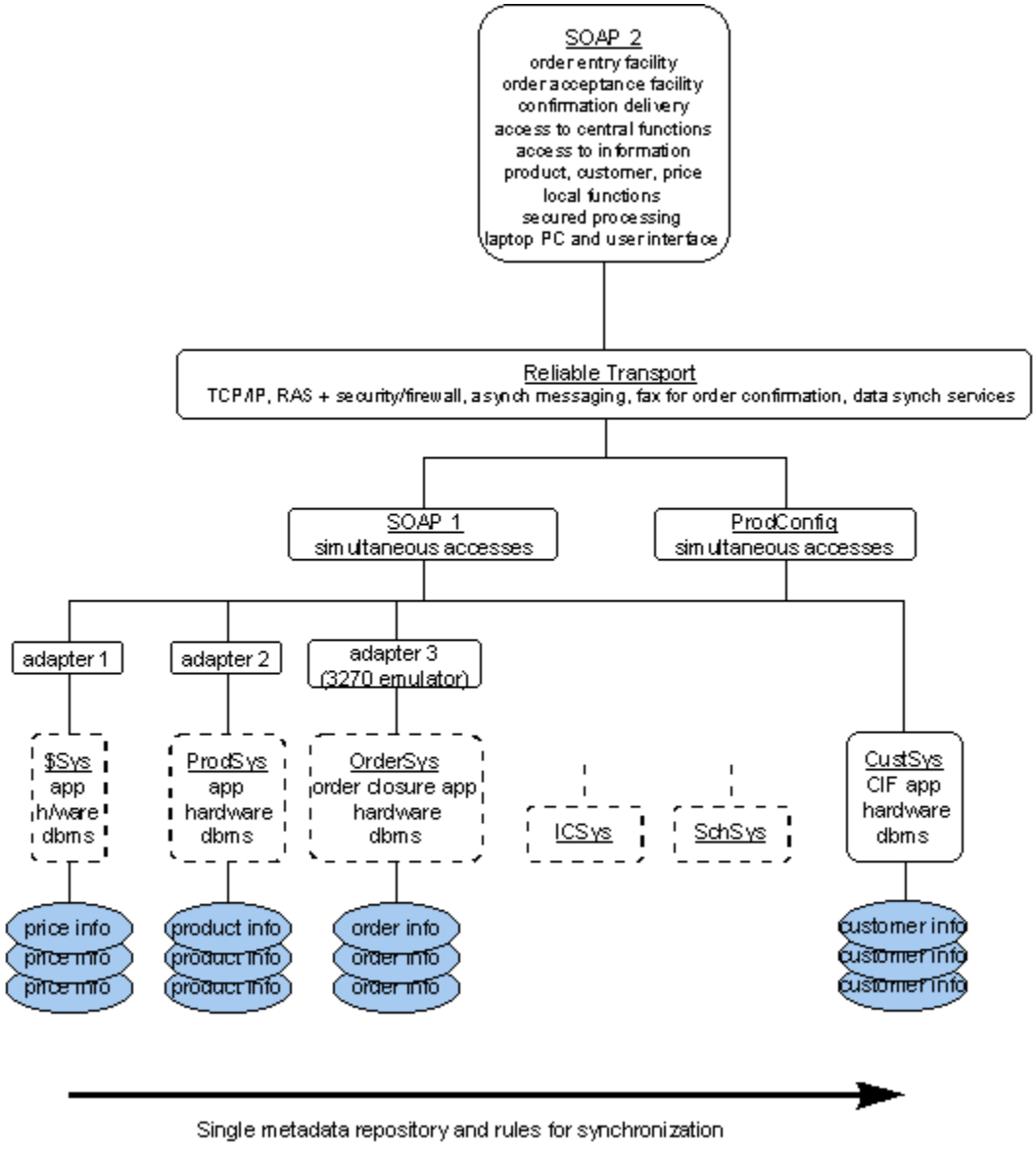


Figure J-8: Augmented Future Architecture of Functions

The future architecture diagram (Fig J-8) shows how the constraints identified in the earlier technical functionality and constraints work have been incorporated. It was necessary to retain the existing systems for order handling and product information. The initial constraint list also included retaining the existing system for customer information, but this was overridden by the need to improve the quality of the sales process and a new system is proposed to deal with this. Return on investment is the driving force behind the decision to retain the existing system for price data. Quality problems with the price system highlighted in Figure J-3 will be resolved through a single metadata definition and rules for synchronization as shown in Figure J-7. These legacy systems are integrated into the new SOAP (Sales Order Application) by developing adapter software. The following describes the SOAP application.

- SOAP consists of two parts. SOAP1 runs at the central site, and SOAP2 on a portable system carried by the sales person. Communication between the two is carried by a reliable transport (TCP/IP and RAS), and includes security provided by a firewall. Asynchronous messaging is also provided. Fax services are required at the central site so as to provide the customer with written confirmation of acceptance of the order. Data synchronization services are needed to keep the sales person's portable and the central systems up to date with each other. Iteration of the architecture development process to validate the results against the business requirements is helped by considering detailed "use-cases". For instance, consider the activity of verifying an entered configuration.
- Entry is handled by SOAP2, running on the sales person's portable system. SOAP2 must deal with:
    - establishment of the link to SOAP1
        - physical link
        - protocols
    - security check
    - direct information request to the proper database
- Then, at the central site:

    - SOAP1 contacts the configurator
    - the configurator:
        - reacts to the named request
        - gets information from the price, product information, customer information and production systems
        - determines the yes or no result
        - returns to SOAP1
    - SOAP1 returns the result to SOAP2

- All of the separate elements in the use-case must be supplied by the solution architecture. Another way of refining the developing architectural model is to use the architecture views:
    - the **computing view** is often the default
    - the **data management view** is often useful
    - the **security** and **management views** are of growing importance
    - performance is an important consideration both on the existing architecture to discover the underlying assumptions and on the future architecture to document the assumptions and provide a basis for change in performance limits. Performance should be addressed in a number of views, including the **computing**, **communications** and **builder's** views.

To ensure that building blocks are as reusable as possible, detailed information is needed about the building block. For this reason it is helpful to take views of individual building blocks and not just of the complete system. For the maximum benefit, it may be necessary to take views of both ABBs and SBBs.

It is the responsibility of the architect to foresee the integration of any application with the rest of the enterprise regardless of the isolated position of the application today. This future integration is facilitated by complete definition of building blocks. It is the responsibility of the business unit to implement in accordance with the rules of the architecture.

Step 3 consists of creating an architecture model of building blocks. The figure above depicts a future architecture model of functions, but does not express the relationships and interfaces between the elements in the architecture model. As the architectural development process continues, it becomes important to define a manageable granularity for building blocks and to fully define their linkages. Without this work there is no guarantee of interoperability between the various building blocks chosen.

We have identified two lists of candidate building blocks in the above steps. Prior to building a model of building blocks these lists are processed and some candidates become recommended building blocks.

Candidate Building Blocks - Business Process Driven List	**Candidate Building Blocks - Baseline Driven List**
sales person's laptop	price list application, data and platform
sales person's CIPR	product information and platform
sales person's LIPR	order acceptance application, data and platform
ProdConfig	
ICSys	
SchSys	
$Sys	
OrderSys	

Table J-2: Candidate Building Blocks - Lists

The process of identifying building blocks includes looking for collections of functions which require integration to draw them together or make them different.

First it is recommended that the candidate building blocks from list B be selected as building blocks because they are reusable legacy items. With these a building block containing all the adapters is identified given the affinity of similar logic, e.g. providing the network adapter functionality on behalf of all the legacy applications.

Next a network building block appears to be required as it is a new network that must be built or purchased and is independent of the applications implemented. It itself can be a reusable building block for other applications.

The laptop with the SOAP 2 application is identified as a building block because it is a modular pack of functionality specially built with applications and data tightly integrated for the mobile sales force. However a RAS-capable firewall was also identified as a separable building block.

The new customer information system is also identified as a reusable building block given its applicability across applications past, new and future. The SOAP 1 and configuration systems were identified as two additional building blocks.

We depict the architecture building blocks at a high level in the following diagram.

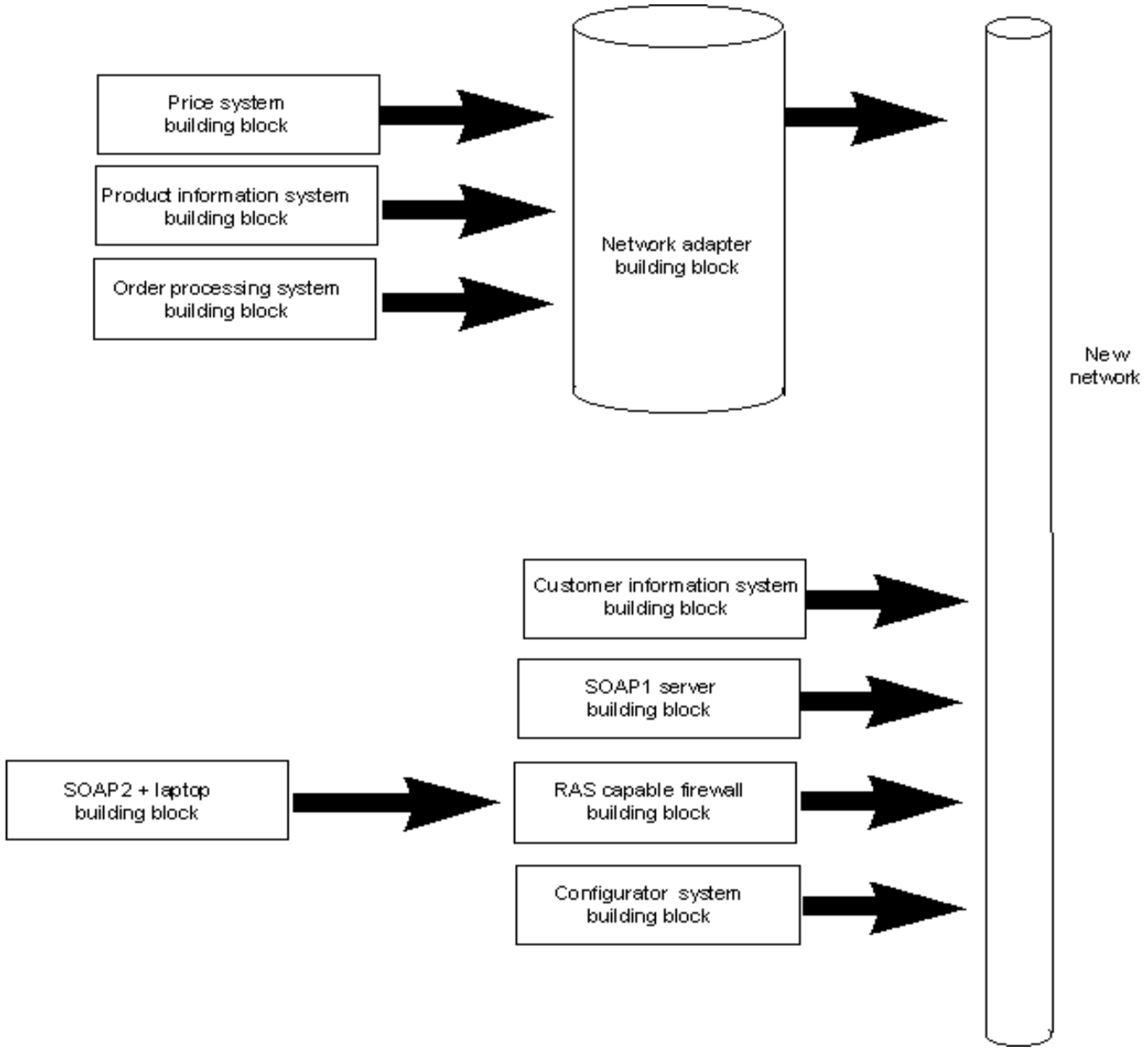


Figure J-9: Representation of XYZ SOAP system

Figure J-9 presents a relationship view of the system. Compare this with Figure J-8, a functional view, to see how different diagrammatic views of the same system can be used to show different things.

In executing step 3 the future architecture was created by processing the technical functionality that must be provided and:

- diagrams of larger systems drawn with this notation quickly show what interfaces are needed between building blocks and which ones need to be identical to realize interoperability benefits.
- Figure J-9 clearly shows where and how glue software is required to bind the legacy systems to the new network.

Step 4 is to select the services portfolio required per building block. The following depicts the services mapped to component in the architecture model.

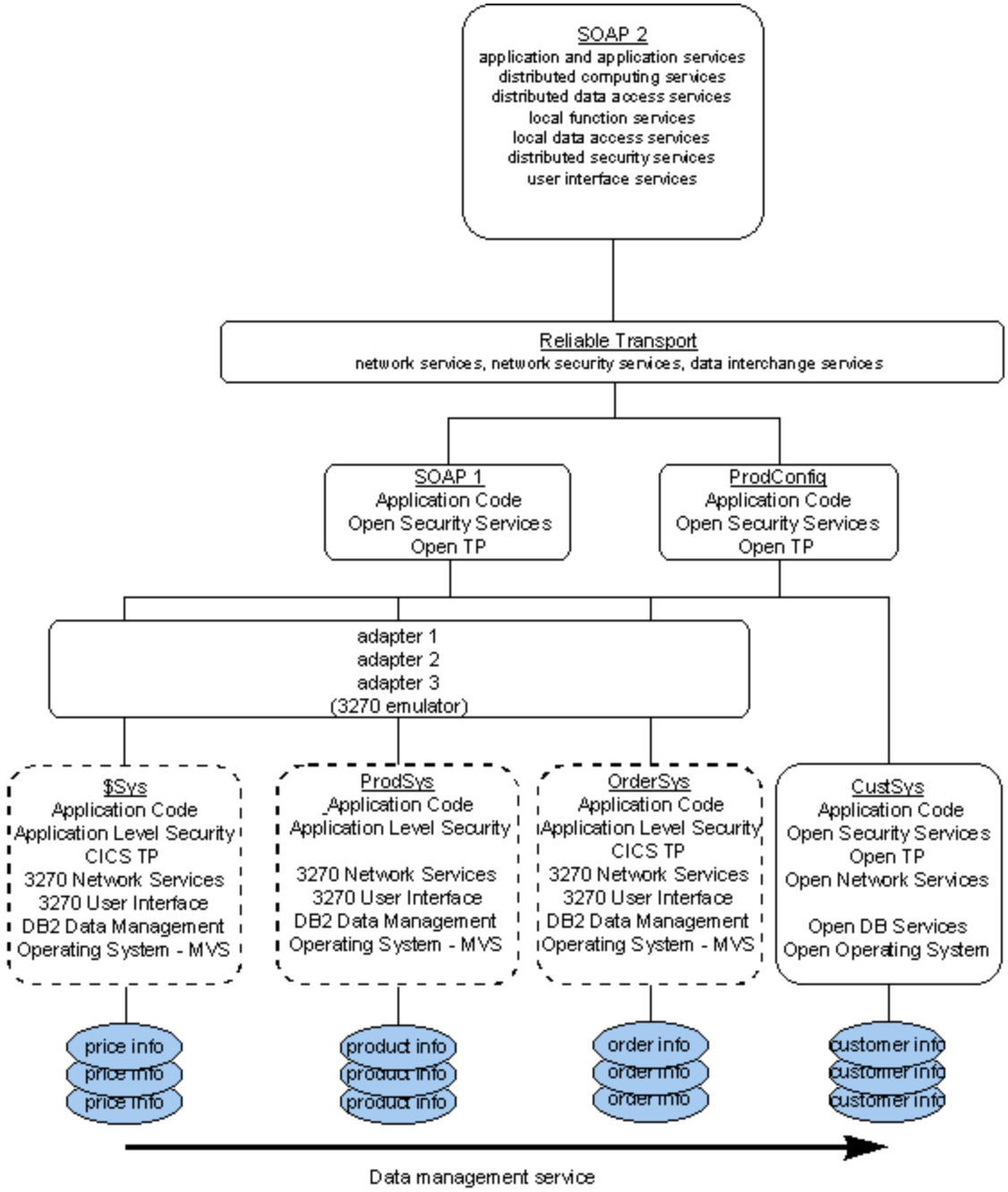


Figure J-10: Services Map

Step 5 in the process is to confirm that the architecture supports the business goals and objectives. This is a relatively subjective task of answering the questions developed in step 1. In this example we did not establish a set of questions that would be used to test the architecture, but one could easily envision such questions and how to pose those questions in light of the architecture. For example one question could be "does the architecture prohibit the immediate processing of an order by a customer?" which would be answered "no" in our case above.

The use-cases developed earlier are a handy tool to test the completeness and applicability of the architecture and its building blocks.

Building block specifications should be recorded in detail. Here is an example of a building block specification document.

Where an enterprise architecture exists or is being developed, it may be valuable at this point to review the new set of building blocks. Anything of benefit to the wider enterprise should be abstracted back to an architectural level and then fed back into the enterprise architecture development process.

Step 8 is to conduct a gap analysis, and is not covered in this example.

The Opportunity Identification level

Phase E - Opportunity Identification

This is the step where projects are identified, ranked and selected.

The steps illustrated above have laid the foundation for this analysis. Figure J-8, for instance, shows the SOAP applications, the reliable transport, the adapters, and the new customer information system as potential projects.

Building Block Reuse level

Phases F to G - Reuse of Building Blocks in Other Projects

In ADM steps F to G, the choice of building blocks may be affected by outside events, such as a change in the availability of products. They can also affect and be affected by issues such as the cost of retraining users during migration from one product to another. Perhaps the most important impact though is the effect that building block choice can have on other work in progress within an organization. This section shows how a diagrammatic representation of the building blocks in a system can be used to identify or prioritize future projects.

An important benefit of defining the building blocks and their linkages is that it becomes possible to pick out reusable components in the architecture. The best way to do this is to draw up a matrix of the building blocks used in an architecture and the applications that use them. Such a matrix for a simple subset of the XYZ case is shown in Figure J-11:

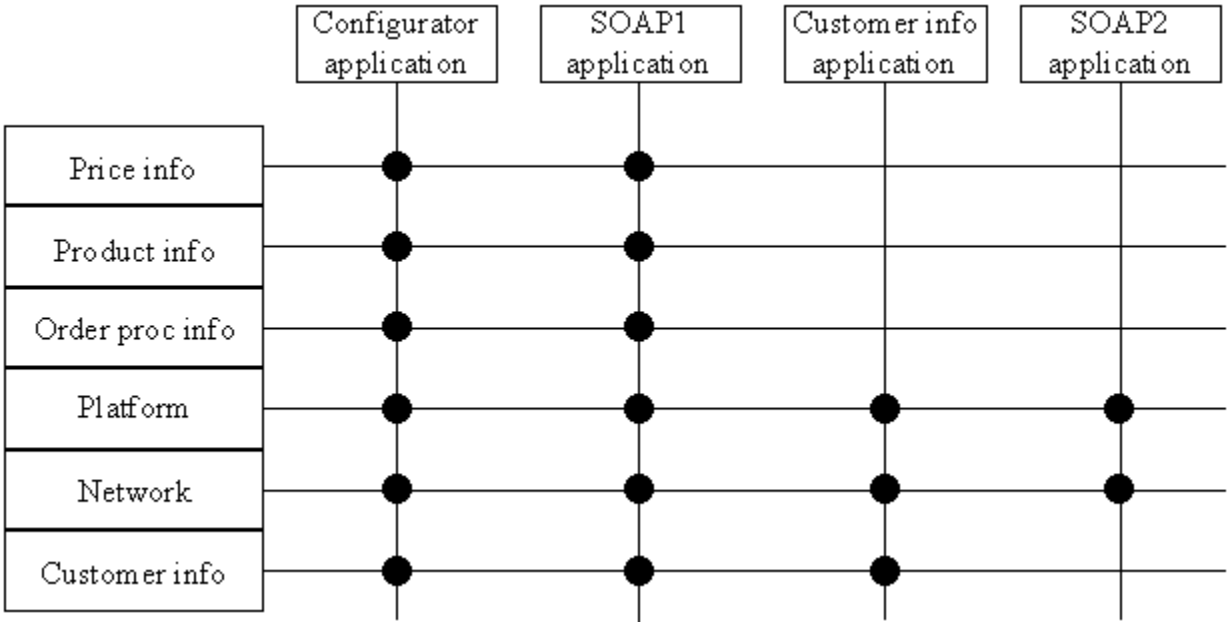


Figure J-11: Simple Component/Application Matrix

Careful ordering of the building blocks in the left hand column allows the architect to identify subsets of functionality common to a number of applications. Figure J-12 shows such a subset. In this case the subset of platform, network and customer information database gives a strong indication that the configurator, SOAP1 and customer information applications should be hosted on the same platform.

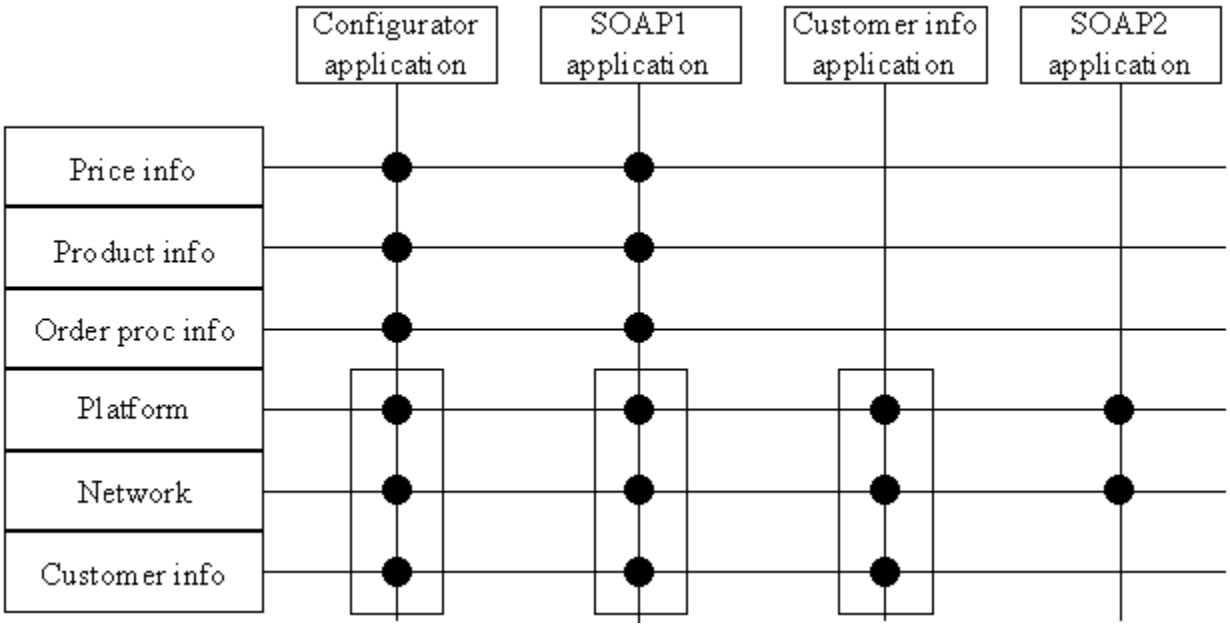


Figure J-12: Identifying Common Functionality

Such identifiable subsets of building blocks also serve another purpose, which is that they can draw attention to opportunities for component reuse. If in the future XYZ decide to implement a customer care system, adding that into the matrix reveals that there would be significant advantages to building the customer care system on the same building blocks used for the configurator, SOAP1 and customer information applications:

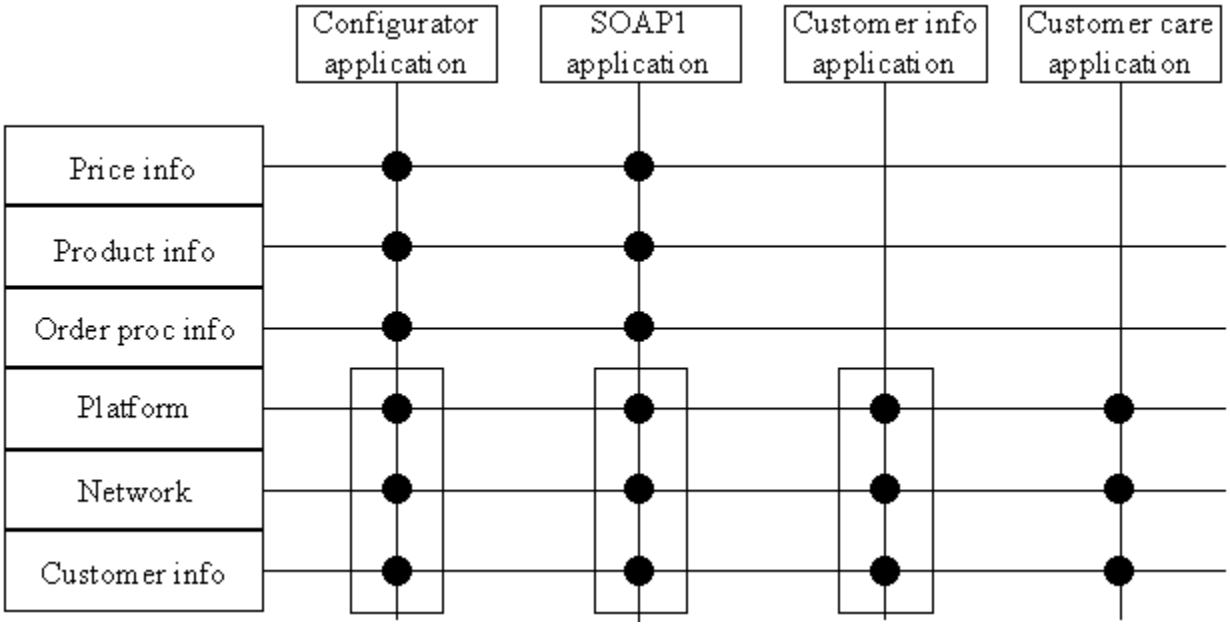


Figure J-13: Using the Matrix to Steer Future Procurement

The key to success in working with building blocks is to establish a useful level of granularity. Too much detail in the chosen set of building blocks will make the architecture unworkable, while too coarse a level of detail will make the work valueless.

Business Process Domain Views

Introduction Role

Introduction

A *Business Process Domain* is a logical grouping of business systems dedicated to a common purpose. Such systems may be geographically co-located, thus emphasising their purpose; or they may grouped by some other constraint, such as a common systems availability target.

In order to demonstrate the responsiveness of the Enterprise Architecture to the business needs of the organization, among the various architecture views that are developed, a *Business Process Domain View* may be used. This describes the Enterprise Architecture from the perspective of the enterprise's key business process domains.

Role

A business process domain view is a set of function views, aligned with the business process structure of the enterprise.

Business Process Domain Views are used during architecture development as a means of verifying, and demonstrating, that the architecture being developed is addressing the business requirements.

Thorough and detailed application of the business scenario technique in the early stages of architecture development should provide an excellent foundation for a set of Business Process Domain views.

Each *Business Process Domain* view addresses components, interfaces, and allocation of services critical to the view. A typical structure is shown below.

Introduction	A description of the domain, its key applications and attributes that characterise the applications within the domain.
Business Problem Statement	A short description of the important business problems relating to the domain, and how they are addressed in the target architecture.
Applications Deployed within the Business Process Domain	A table listing currently deployed applications
Assumptions, Constraints and Guidelines	General guidelines for developers, implementers and system suppliers
Domain Structure	Target Architecture / Business Process Domain Mapping: A table highlighting the Services and Building Blocks applicable to this domain. Application Topology: A figure showing the relationships between the major application elements within this domain
Domain Service Qualities	A description of the service qualities (see Application Platform Taxonomy) that are important for each business process domain, and how they are achieved in the target architecture.

Deployment Guidance Strategy	Lessons learnt with respect to deployment. The migration strategy for implementing the target architecture, as it applies to this business domain. Also any guidance for the implementation team responsible for deployment.
Future Directions	Any important directions identified for systems within the domain
References	Pointers to reference material

Table 1: Business Process Domain Views - Description Structure

Business Scenarios

Introduction

A key factor in the success of an enterprise architecture is the extent to which it is linked to business requirements, and demonstrably supporting and enabling the enterprise to achieve its business objectives.

Business scenarios are an important technique that may be used at various stages of the enterprise architecture, principally the Architecture Vision and the Business Architecture, but in other architecture domains as well, if required, to derive the characteristics of the Architecture directly from the high-level requirements of the business. They are used to help identify and understand business needs, and thereby to derive the business requirements that the architecture development has to address.

A Business Scenario describes:

- a business process, application, or set of applications, that can be enabled by the architecture
- the business and technology environment
- the people and computing components (called "actors") who execute the scenario
- the desired outcome of proper execution

A good Business Scenario is representative of a significant business need or problem, and enables vendors to understand the value to the customer organization of a developed solution.

A good Business Scenario is also "SMART":

- **S**pecific, by defining what needs to be done in the business
- **M**easurable, through clear metrics for success
- **A**ctionable, by
  - clearly segmenting the problem, and
  - providing the basis for determining elements and plans for the solution
- **R**ealistic, in that the problem can be solved within the bounds of physical reality, time and cost constraints
- **T**ime-bound, in that there is a clear statement of when the solution opportunity expires

The section below entitled “Guidelines on Goals” provides detailed examples on objectives that could be considered. Whatever objectives you use the idea is to make those objectives SMART.

Benefits of Business Scenarios

A business scenario is essentially a complete description of a business problem, both in business and in architectural terms, which enables individual requirements to be viewed in relation to one another in the context of the overall problem. Without such a complete description to serve as context:

- There is a danger of the architecture being based on an incomplete set of requirements that do not add up to a whole problem description, and that can therefore misguide architecture work.
- The business value of solving the problem is unclear
- The relevance of potential solutions is unclear

Also, because the technique requires the involvement of business line management and other stakeholders at an early stage in the architecture project, it also plays an important role in gaining the buy-in of these key personnel to the overall project and its end-product - the enterprise architecture.

An additional advantage of business scenarios is in communication with vendors. Most architecture nowadays is implemented by making maximum use of commercial off-the-shelf software (COTS) solutions, often from multiple vendors, procured in the open market. The use of business scenarios by an IT customer can be an important aid to IT vendors in delivering appropriate solutions. Vendors need to ensure that their solution components add value to an open solution and are marketable. Business scenarios provide a language with which the vendor community can link customer problems and technical solutions. Besides making obvious what is needed, and why, they allow vendors to solve problems optimally, using open standards and leveraging each other's skills.

Creating the Business Scenario

The Overall Process

Creating a business scenario involves the following, as illustrated in Figure 1:

1 - Identifying, documenting and ranking the problem driving the scenario

2 - Identify the business and technical environment of the scenario and documenting it in scenario models

3 - Identifying and documenting desired objectives (the results of handling the problems successfully) - get "SMART"

4 - Identifying the human actors (participants) and their place in the business model

5 - Identifying computer actors (computing elements) and their place in the technology model

6 - Identifying and documenting roles, responsibilities and measures of success per actor; documenting the required scripts per actor, and the results of handling the situation

7 - Checking for "fitness for purpose" and refining only if necessary.

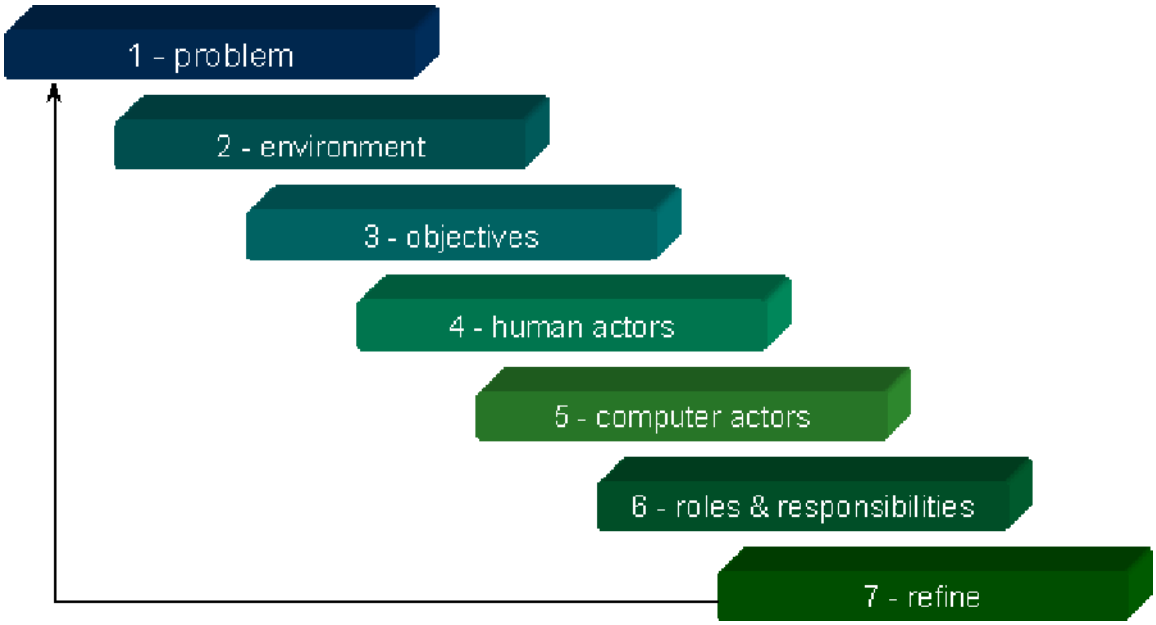


Figure 1: Creating a Business Scenario

A business scenario is developed over a number of iterative phases of Gathering, Analyzing, and Reviewing the information in the Business Scenario.

In each Phase, each of the areas above is successively improved. The refinement step involves deciding whether to consider the scenario complete and go to the next phase, or whether further refinement is necessary. This is accomplished by asking whether the current state of the business scenario is fit for the purpose of carrying requirements downstream in the

architecture process.

The three phases of developing a business scenario are described in detail below, and depicted in Figure 2.

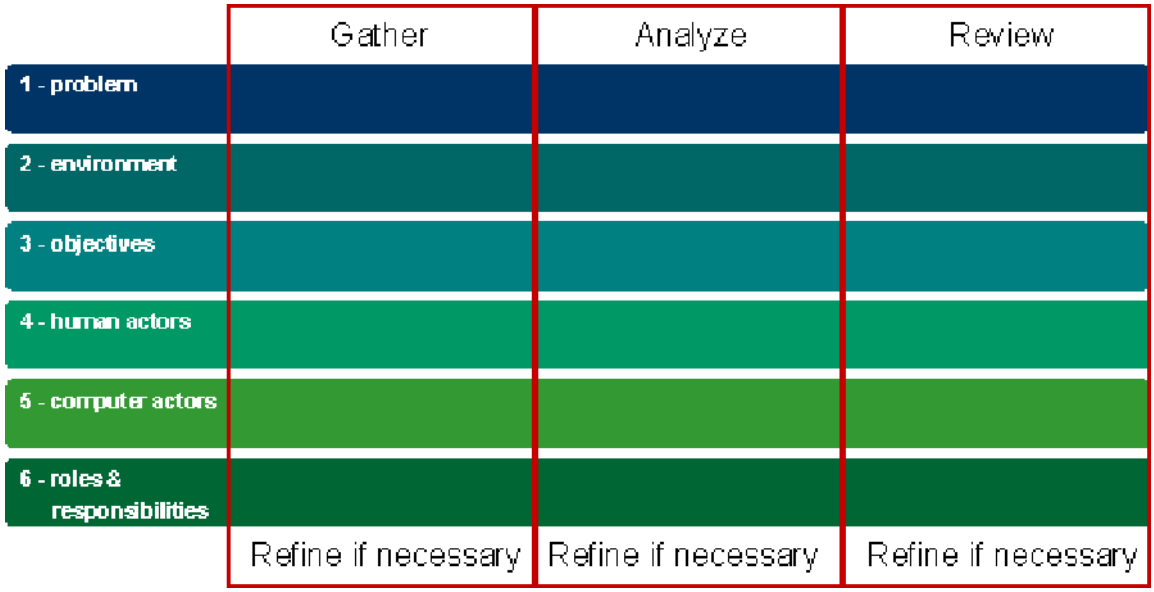


Figure 2: Phases of Developing Business Scenarios

Gathering

The Gathering phase is where information is collected on each of the areas in Figure 1. If information gathering procedures and practices are already in place in an organization - for example, to gather information for strategic planning - they should be used as appropriate, either during business scenario workshops or in place of business scenario workshops.

Multiple techniques may be used in this phase, such as information research, qualitative analysis, quantitative analysis, surveys, requests for information, etc. As much information as possible should be gathered and preprocessed “off-line” prior to any face-to-face workshops (described below). For example, a request for information may include a request for strategic and operational plans. Such documents typically provide great insights, but the information that they contain usually requires significant preprocessing. The information may be used to generate an initial draft of the business scenario prior to the workshop, if possible. This will increase the understanding and confidence of the architect, and the value of the workshop to its participants.

A very useful way to gather information is to hold **business scenario workshops**, whereby a business scenario consultant leads a select and small group of business representatives through a number of questions to elicit the information surrounding the problem being addressed by the architecture effort. The workshop attendees must be carefully selected from high levels in the business and technical sides of the organization. It is important to get people that can and will provide information openly and honestly. Where a draft of the Business Scenario already exists - for example, as a result of preprocessing information gathered during this phase, as described above - the workshop may also be used to review the state of the Business Scenario draft.

Sometimes it is necessary to have multiple workshops: in some cases, to separate the gathering of information on the business side from the gathering of information on the technical side; and in other cases simply to get more information from more people.

When gathering information, the architect can greatly strengthen the business scenario by obtaining “real world examples” - case studies to which the reader can easily relate. When citing real world examples, it is important to maintain a level of anonymity of the parties involved, to avoid blame!!

Analyzing

The Analyzing phase is where a great deal of real business architecture work is actually done. This is where the information that is gathered is processed, and documented, and where the models are created to represent that information, typically visually.

The analysis phase takes advantage of the knowledge and experience of the business scenario consultant using past work and experience to develop the models necessary to depict the information captured. Note that the models and documentation produced are not necessarily reproduced verbatim from interviews, but rather filtered and translated according to the real underlying needs.

In the analysis phase it is important to maintain linkages between the key elements of the business scenario. One technique that assists in maintaining such linkages is the creation of matrices that are used to relate business processes to each of:

- Constituencies
- Human actors
- Computer actors
- Issues
- Objectives

In this way, the business process becomes the binding focal point – which makes a great deal of sense, since in most cases it is business process improvement that is being sought.

Reviewing

The Reviewing phase is where the results are fed back to the sponsors of the project to ensure that there is a shared understanding of the full scope of the problem, and the potential depth of the technical impact.

Multiple Business Scenario workshops or 'readout' meetings with the sponsors and involved parties are recommended. The meetings should be set up to be open and interactive. It is recommended to have exercises built into meeting agendas, in order to test attendees' understanding and interest levels, as well as to test the architect's own assumptions and results.

This phase is extremely important, as the absence of shared expectations is in many cases the root cause of project failures.

Contents of a Business Scenario

The documentation of a Business Scenario should contain all the important details about the scenario. It should capture, and sequence, the critical steps and interactions between actors that address the situation. It should also declare all the relevant information about all actors, specifically: the different responsibilities of the actors; the key pre-conditions that have to be met prior to proper system functionality; and the technical requirements for the service to be of acceptable quality.

There are two main types on content: graphics (models), and descriptive text. Both have a part to play.

Business Scenario Models capture business and technology views in a graphical form, to aid comprehension. Specifically, they relate actors and interactions, and give a starting point to confirm specific requirements.

Business Scenario Descriptions capture details in a textual form. A typical contents list for a business scenario is given below.

Table of Contents

BUSINESS ENVIRONMENT
    Constituencies
PROCESS DESCRIPTIONS
    Process “a”
    …etc.
TECHNICAL ENVIRONMENT
    Technical environment “a”
    …etc.
ACTORS AND THEIR ROLES AND RESPONSIBILITIES
COMPUTER ACTORS AND ROLES
RELATIONSHIP OF COMPONENTS AND PROCESSES
HUMAN ACTORS AND ROLES
RELATIONSHIP OF HUMANS AND PROCESSES
INFORMATION FLOW ANALYSIS
PRINCIPLES AND CONSTRAINTS
    IT Principles
    Constraints
REQUIREMENTS

BUSINESS SCENARIO ANALYSIS

PROBLEM SUMMARY
    Issues
    Objectives

SUMMARY

APPENDIXES

APPENDIX A: BUSINESS SCENARIOS – ADDITIONAL INFORMATION

APPENDIX B-n: BUSINESS SCENARIO WORKSHOP NOTES

Contributions to the Business Scenario

It is important to realize that the creation of a business scenario is not solely the province of the Architect. As mentioned previously, business line management and other stakeholders in the enterprise are involved, to ensure that the business goals are accurately captured. In addition, depending on the relationship that an organization has with its IT vendors, the latter also may be involved, to ensure that the roles of technical solutions are also accurately captured, and to ensure communication with the vendors.

Typically, the involvement of the business management is greatest in the early stages, while the business problems are being explored and captured, while the involvement of the architect is greatest in the later stages, when architectural solutions are being described. Similarly, if vendors are involved in the business scenario process, the involvement of the customer side (business management plus enterprise architects) is greatest in the early stages, while that of the vendors is greatest in the later stages, when the role of specific technical solutions is being explored and captured.This concept is illustrated in Figure 3.

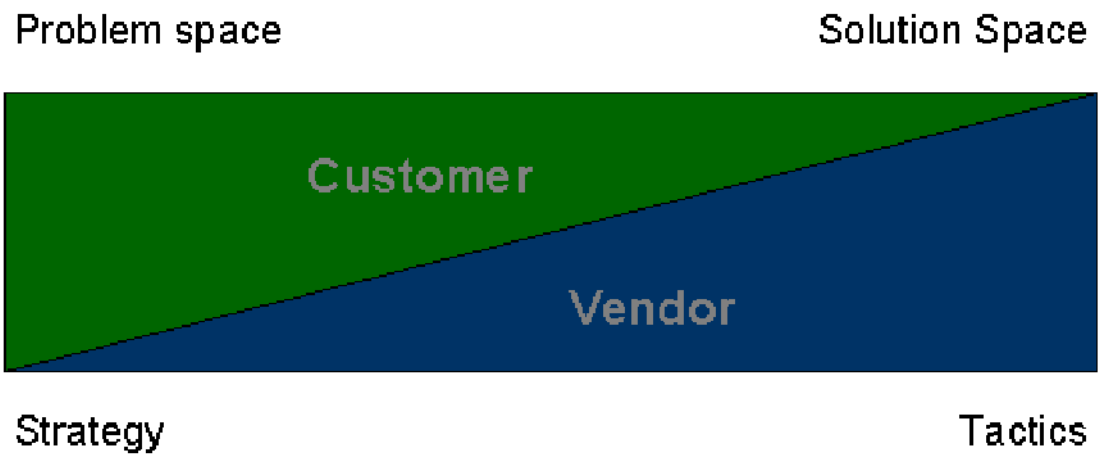


Figure 3: Relative Contributions to a Business Scenario

Vendor IT architects might be able to assist enterprise IT architects with integration of the vendors' products into the enterprise architecture. This assistance most probably falls in the middle of the timeline in Figure 3.

Business Scenarios and the TOGAF ADM

Business Scenarios figure most prominently in the initial phase of the ADM, Architecture Vision, when they are used to define relevant business requirements, and to build consensus with business management and other stakeholders.

However, the business requirements are referred to throughout all phases of the ADM life cycle, as illustrated in Figure 4.

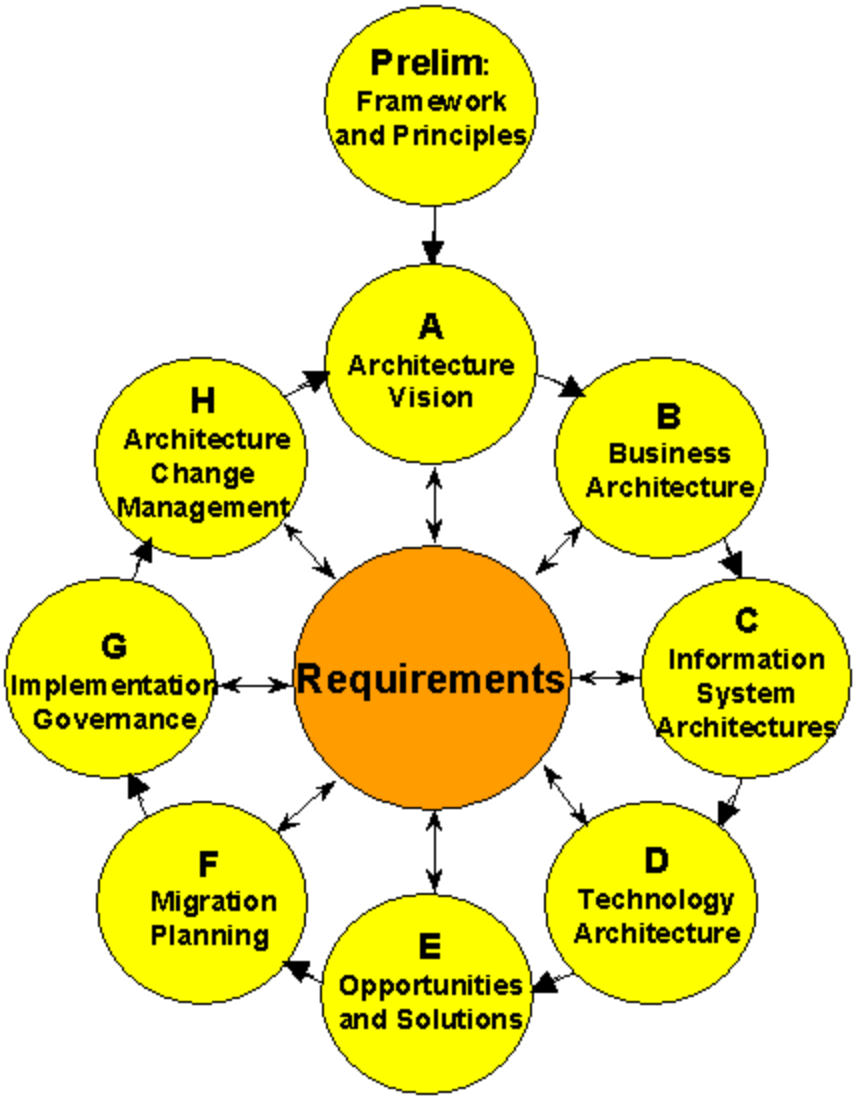


Figure 4: Relevance of Requirements Throughout the ADM

Because business requirements are important throughout all phases of the ADM life cycle, the Business Scenario technique has an important role to play in the TOGAF ADM, by ensuring that the business requirements themselves are complete and correct.

Guidelines on Developing Business Scenarios

General Guidelines

The stakeholders (e.g., business managers, end-users) will tell you what they want, but as an architect you must still gain an understanding of the business, so you must know the most important actors in the system. If the stakeholders do not know what they want:

- Take time, observe and record how they are working today
- Structure information in such a way that it can be used later
- Uncover critical business rules from domain experts
- Stay focused on what needs to be accomplished, how it is to be accomplished.

This effort provides the anchor for a chain of reason from business requirements through to technical solutions. It will pay off later to be diligent and critical at the start.

Questions to Ask for Each Area

The business scenario workshops mentioned above in the Gathering phase are really structured interviews. While there is no single set of appropriate questions to ask in all situations, the following provides some guidance to help business scenario consultants in asking questions.

Identifying, documenting and ranking the problem

Is the problem described as a statement of WHAT needs to be accomplished, like steps in a process, and not HOW (with technology "push")?

If the problem is too specific or a “how”:

- Raise a red flag
- Ask "why do you need to do it that way?" questions

If the problem is too vague or unactionable:

- Raise a red flag
- Ask "what is it you need to do, or will be able to do if this problem is solved?" questions

Ask questions that help identify where and when the problem exists:

- Where are you experiencing this particular problem? In what business process?
- When do encounter these issues? During the beginning of the process, the middle, the end?

Ask questions that help identify the costs of the problem:

- Do you account for the costs associated with this problem? If so, what are they?
- Are there hidden costs? If so, what are they?
- Is the cost of this problem covered in the cost of something else? If so, what and how much?
- Is the problem manifested in terms of poor quality or a perception of an ineffective organization?

Identifying the business and technical environment, and documenting in models

Questions to ask about the business environment:

- What key process suffers from the issues? What are the major steps that need to be processed?
- Location/scale of internal business departments?
- Location/scale of external business partners?
- Any specific business rules and regulations related to the situation?

Questions to ask about the current technology environment:

- What technology components are already presupposed to be related to this problem?
- Are there any technology constraints?
- Are there any technology principles that apply?

Identifying and documenting objectives

Is the "what" sufficiently backed up with the rationale for "why"? If not, ask for measurable rationale in the following areas:

- Return on investment
- Scalability
- Performance needs
- Compliance to standards
- Ease of use measures

Identifying human actors and their place in the business model

An actor represents anything that interacts with or within the system. This can be a human, or a machine, or a computer program. Actors initiate activity with the system, e.g.:

- Computer user with the computer
- Phone user with the telephone
- Payroll clerk with the Payroll System
- Internet subscriber with the web browser

An actor represents a role that a user plays: i.e., a **user** is someone playing a **role** while using the system (e.g., John (user) is a dispatcher (actor)). Each actor uses the system in different ways (otherwise they should be the same actor). Ask about the humans that will be involved, from different view points, such as:

- Developer
- Maintainer
- Operator
- Administrator
- User

Identify computer actors and their place in the technology model

Ask about the computer components likely to be involved, again from different points of view. What must they do?

Documenting roles, responsibilities, measures of success, required scripts

When defining roles, ask questions like:

- What are the main tasks of the actor?

- Will the actor have to read/write/change any information?
- Will the actor have to inform the system about outside changes?
- Does the actor wish to be informed about unexpected changes?

Checking for "fitness for purpose", refining if necessary

Is there enough information to identify who/what could fulfil the requirement? If not, probe more deeply.

Is there a description of when, and how often, the requirement needs to be addressed? If not, ask about timing.

Guidelines on Business Scenario Documentation

Textual documentation

Effective Business Scenario documentation requires a balance between ensuring that the detail is accessible, and preventing it from overshadowing the results and overwhelming the reader. To this end, the Business Scenario document should have the main findings in the body of the document and the details in appendices.

In the appendices:

- Capture all the important details about a Business Scenario:
  - Situation description and rationale
  - All measurements
  - All actor roles and sub-measurements
  - All services required
- Capture the critical steps between actors that address the situation, and sequence the interactions
- Declare relevant information about all actors
  - Partition the responsibility of the actors
  - List pre-conditions that have to be met prior to proper system functionality
  - Provide technical requirements for the service to be of acceptable quality

In the main body of the Business Scenario:

- Generalize all the relevant data from the detail in the appendices.

Business Scenario Models

- Remember the purpose of using models:
  - Capture business and technology views in a graphical form
  - Help comprehension
  - Give a starting point to confirm requirements
  - Relate actors and interactions
- Keep drawings clear and neat
  - Do not put too much into one diagram
  - Simpler diagrams are easier to understand
- Number diagrams for easy reference
  - Maintain a catalog of the numbers to avoid duplicates

Guidelines on Goals and Objectives

The Importance of Goals

One of the first steps in the development of an architecture is to define the overall goals and objectives for the development. The objectives should be derived from the business goals of the organization, and the way in which Information Technology is seen to contribute to meeting those goals.

Every organization behaves differently in this respect, some seeing IT as the driving force for the enterprise and others seeing IT in a supporting role, simply automating the business processes which already exist. The essential thing is that the architectural objectives should be very closely aligned with the business goals and objectives of the organization.

The Importance of SMART Objectives

Not only must goals be stated in general terms, but also specific measures need to be attached to them to make them SMART, as described above.

The amount of effort spent in doing this will lead to greater clarity for the sponsors of the architecture evolution cycle. It will pay back by driving proposed solutions much more closely toward the goals at each step of the cycle. It is extremely helpful for the different stakeholders inside the organization, as well as for suppliers and consultants, to have a clear yardstick for measuring fitness for purpose. If done well, the architecture development method can be used to trace specific decisions back to criteria, and thus yield their justification.

The goals below have been adapted from those given in previous Versions of TOGAF. These are categories of goals, each with a list of possible objectives. Each of these objectives should be made SMART with specific measures and metrics for the task. However, since the actual work to be done will be specific to the architecture project concerned, it is not possible to provide a list of generic SMART objectives that will relate to any project.

Instead, we provide here some example SMART objectives.

Example of Making Objectives "SMART"

Under the general goal heading “improve user productivity” below, there is an objective to provide a “consistent user interface” and it is described as follows:

> *“A consistent user interface will ensure that all user accessible functions and services will appear and behave in a similar, predictable fashion regardless of application or site. This will lead to better efficiency and fewer user errors, which in turn may result in lower recovery costs.”*

To make this objective SMART, one asks whether the objective is specific, measurable, actionable, realistic, and time-bound, and then augments the objective appropriately.

The following captures an analysis of these criteria for the stated objective.

- **Specific** – the objective of providing “a consistent user interface that will ensure all user accessible functions and services will appear and behave in a similar, predictable fashion regardless of application or site” is pretty specific. However the measures listed in the second sentence could be more specific....
- **Measurable** – as stated above, the objective is measurable, but could be more specific. The second sentence could be amended to read (for example): “This will lead to 10% greater user efficiency and 20% fewer order entry user errors, which in turn may result in 5% lower order entry costs.”
- **Actionable** – the objective does appear to be actionable. It seems clear that consistency of the user interface must be provided, and that could be handled by whoever is responsible for providing the user interface to the user device.
- **Realistic** - the objective of providing “a consistent user interface that will ensure all user accessible functions and services will appear and behave in a similar, predictable fashion regardless of application or site” might not be realistic. Considering the use today of PDAs at the user end might lead one to augment this objective to assure that the downstream developers don’t unduly create designs that hinder the use of new technologies. The objective could be re-stated as “a consistent user interface, *across user interface devices that provide similar functionality*, that will ensure ..." etc.
- **Time-bound** – the objective as stated is not time-bound. To be time-bound the objective could be re-stated as “By the end of quarter 3 provide a consistent….”

The above results in a SMART objective that looks more like this (again remember this is an example):

> *“By the end of quarter 3 provide a consistent user interface across user interface devices that provide similar functionality to ensure all user accessible functions and services appear and behave in a similar when using those devices in a predictable fashion regardless of application or site. This will lead to 10% greater user*

efficiency and 20% fewer order entry user errors, which in turn may result in 5% lower order entry costs."

Categories of Goals and Objectives

Although every organization will have its own set of goals, some examples may help in the development of an organization-specific list. The goals given below are categories of goals, each with a list of possible objectives, which have been adapted from the Goals given in previous Versions of TOGAF.

Each of the objectives given below should be made SMART with specific measures and metrics for the task involved, as illustrated in the example above. However, the actual work to be done will be specific to the architecture project concerned, and it is not possible to provide a list of generic SMART objectives that will relate to any project.

Goal: Improve Business Process Performance

Business process improvements can be realized through the following **objectives**:

- Increased process throughput
- Consistent output quality
- Predictable process costs
- Increase re-use of existing processes
- Reduce time of sending business information from one process to another process

Goal: Decrease Costs

Cost improvements can be realized through the following **objectives**:

- Lower levels of redundancy and duplication in assets throughout the enterprise,
- Decreased the reliance on external IT service providers for integration and customization
- Lower costs of maintenance

Goal: Improve Business Operations

Business operations improvements can be realized through the following **objectives**:

- Increased budget available to new business features
- Decreased costs of running the business
- Decreased time to market for products or services
- Increase the quality of services to customers
- Improved quality of business information

Goal: Improve Management Efficacy

Management efficacy improvements can be realized through the following **objectives**:

- Increased flexibility of business
- Shorter time to make decisions
- Higher quality decision

Goal: Reduce Risk

Risk improvements can be realized through the following **objectives**:

- Ease of implementing new processes
- Decreased errors introduced into business processes through complex and faulty systems
- Decreased real world safety hazards (including hazards that cause loss of life)

Goal: Improve Effectiveness of IT Organization

IT organization effectiveness can be realized through the following **objectives**:

- Increase in rollout of new projects
- Decreased time to rollout new projects
- Lower cost in rolling out new projects
- Decreased loss of service continuity when rolling out new projects
- Common Development. Applications that are common to multiple business areas will be developed or acquired once and re-used rather than separately developed by each business area.
- Open Systems Environment. A standards-based common operating environment, which accommodates the injection of new standards, technologies and applications on an organization-wide basis, will be established. This standards-based environment will provide the basis for development of common applications and facilitate software reuse.
- Use of Products. As far as possible, hardware-independent, off-the-shelf items should be used to satisfy requirements in order to reduce dependence on custom developments and to reduce development and maintenance costs.
- Software Reuse. For those applications that must be custom developed, development of portable applications will reduce the amount of software developed and add to the inventory of software suitable for reuse by other systems.
- Resource Sharing. Data processing resources (hardware, software and data) will be shared by all users requiring the services of those resources. Resource sharing will be accomplished in the context of security and operational considerations.

Goal: Improve User Productivity

User productivity improvements can be realized through the following **objectives**:

- Consistent User Interface. A consistent user interface will ensure that all user accessible functions and services will appear and behave in a similar, predictable fashion regardless of application or site. This will lead to better efficiency and fewer user errors, which in turn may result in lower recovery costs.
- Integrated Applications. Applications available to the user will behave in a logically consistent manner across user environments, which will lead to the same benefits as a consistent user interface.
- Data Sharing. Databases will be shared across the organization in the context of security and operational considerations, leading to increased ease of access to required data.
  Goal: Improve Portability and Scalability
  The portability and scalability of applications will be through the following objectives:
- Portability. Applications that adhere to Open Systems standards will be portable, leading to increased ease of movement across heterogeneous computing platforms. Portable applications can allow sites to upgrade their platforms as technological improvements occur, with minimal impact on operations.
- Scalability. Applications that conform to the model will be configurable, allowing operation on the full spectrum of platforms required.

Goal: Improve Interoperability

Interoperability improvements across applications and business areas can be realized through the following **objectives**:

- Common Infrastructure. The architecture should promote a communications and computing infrastructure based on open systems and systems transparency including, but not limited to, operating systems, database management, data interchange, network services, network management and user interfaces.
- Standardization. By implementing standards-based platforms, applications will be provided with and will be able to use a common set of services that improve the opportunities for interoperability.
  Goal: Increase Vendor Independence
  Vendor independence will be increased through the following objectives:
- Interchangeable Components. Only hardware and software that has standards-based interfaces will be selected, so that upgrades or the insertion of new products will result in minimal disruption to the user's environment.
- Non-proprietary Specifications. Capabilities will be defined in terms of non-proprietary specifications that support full and open competition and are available to any vendor for use in developing commercial products.

Goal: Reduce Life-Cycle Costs

Life-cycle costs can be reduced through most of the objectives discussed above. In addition, the following **objectives** directly address reduction of life-cycle costs:

- Reduced Duplication. Replacement of isolated systems and islands of automation with interconnected open systems will lead to reductions in overlapping functionality, data duplication and unneeded redundancy, because open systems can share data and other resources.
- Reduced Software Maintenance Costs. Reductions in the quantity and variety of software used in the organization will lead to reductions in the amount and cost of software maintenance. Use of standard off-the-shelf software will lead to further reductions in costs since vendors of such software distribute their product maintenance costs across a much larger user base.

- Incremental replacement. Common interfaces to shared infrastructure components allow for phased replacement or upgrade with minimal operational disturbance.
- Reduced Training Costs. Common systems and consistent human computer interfaces will lead to reduced training costs.

Goal: Improve Security

Security can be improved in the organization's information through the following **objectives**:

- Consistent Security Interfaces for Applications. Consistent security interfaces and procedures will lead to fewer errors when developing applications and increased application portability. Not all applications will need the same suite of security features, but any features used will be consistent across applications.
- Consistent Security Interfaces for Users. A common user interface to security features will lead to reduced learning time when moving from system to system.
- Security Independence. Application deployment can use the security policy and mechanisms appropriate to the particular environment if there is good layering in the architecture.
- A 25% reduction in calls to the Help desk relating to security issues
- A 20% reduction in "false positives" detected in the network (a "false positive" is an event that appears to be an actionable security event, but in fact is a false alarm)

Goal: Improve Manageability

Management improvement can be realized through the following objectives:

- Consistent Management Interface. Consistent management practices and procedures will facilitate management across all applications and their underlying support structures. A consistent interface can simplify the management burden, leading to increased user efficiency.
- Reduced Operation, Administration and Maintenance Costs. Operation, administration and maintenance costs may be reduced through the availability of improved management products and increased standardization of the objects being managed.

Summary

Business Scenarios help address one of the most common issues facing IT executives: aligning Information Technology with the business.

The success of any major IT project is measured by the extent to which it is linked to business requirements, and demonstrably supports and enables the enterprise to achieve its business objectives. Business Scenarios are an important technique that may be used at various stages of defining enterprise architecture, or any other major IT project, to derive the characteristics of the architecture directly from the high-level requirements of the business. Business Scenarios are used to help identify and understand business needs, and thereby to derive the business requirements that the architecture development, and ultimately the information technology, has to address.

However, it is important to remember that Business Scenarios are just a tool, not the objective. They are a part of, and enable, the larger process of architecture development. The architect should use them, but not get lost it them. The key is to stay focused - watch out for "feature creep", and address the most important issues that tend to return the greatest value.

Case Studies

Role Summary

The Role of Case Studies

One of the goals of The Open Group's Architecture Forum is to provide a forum within which both customer and vendor organizations can exchange feedback and experience in the use of TOGAF.

All of this feedback is considered in the on-going evolution of TOGAF within the Architecture Forum, and indeed much of it has already been incorporated in this current version of TOGAF.

It is important to emphasise that no case study can provide a complete blueprint for how another organization should go about using TOGAF. All organizations are different, and each organization should understand what TOGAF has to offer, and adopt and adapt the parts that are useful for its needs.

Nevertheless, over the years of its existence as an architectural framework, TOGAF has been used by a variety organizations around the world in major architecture developments, both in its entirety, and in an adapted form.

It is hoped that the case studies presented here will provide useful guidance to organizations intending to use TOGAF or considering using it to develop an enterprise IT architecture.

The formats of the case studies vary, between normal descriptive text, through presentations, and in some cases video.

Summary of Case Studies

The following case studies show TOGAF in use in a variety of situations.

- Dairy Farm Group (Hong Kong)
- Department of Social Security (UK)
- Litton PRC (US)
- Ministry of Defence (UK)
- NATO (Belgium)
- Police IT Organization (UK)
- QA Consulting
- Statskonsult (Norway)
- Westpac (Australia)

Dairy Farm Group (Hong Kong)

The Dairy Farm Group case study illustrates extensive use of TOGAF as the basis of an enterprise-wide IT architecture to integrate many disparate business units.

The Dairy Farm Group (DFG) is a holding company in the Retail sector. It has a very strong presence in the Asia / Pacific region, and is the 71st largest retailing company world wide.

DFG has a corporate goal to be the largest, most successful retailer in Asia / Pacific in its chosen markets. To support this goal, DFG has restructured from a federation to a unified group of companies, with a single corporate purpose business focus, and a single IT infrastructure.

The DFG Technical Program Architecture Group (TAPG) was chartered to develop a Technical Architecture for DFG, and chose TOGAF and its supporting methodology as the basis. Using TOGAF, the TAPG was able, in a very short period of time, from July through October 1998, to develop a world class technical architecture.

It was particularly helpful to be able to point key suppliers to a published version of TOGAF, so that they could see an explanation of the methodology and refer to the TOGAF Standards Information Base.

Formats:

- This case study has its own web site. Specific formats:
  - A set of video clips in MPEG format
  - Presentation (PDF)
  - Descriptive summary text

An more recent update (2001) is also available.

Department of Social Security (UK)

The UK Department of Social Security (DSS) is responsible for the development, maintenance and delivery of the United Kingdom's social security programme. It utilises the largest civilian computer operation in Europe.

The DSS case study illustrates the use of TOGAF, both as the basis of a new architectural framework, and as a key tool in managing the outsourcing of service delivery, in that vendors and integrators were required to use TOGAF as the basis for their tenders, and to use it subsequently in the on-going management of the IT architecture.

- The use of TOGAF is explained in the DSS case study.

Litton PRC (US)

Litton PRC chose the TOGAF Architecture Development Method as a basis for its revamped internal Architecture Design Process. The Litton PRC case study (PDF) reviews the use of TOGAF within Litton PRC, and explains why TOGAF was chosen.

Also, the JEDMICS case study illustrates the early stages of the Architecture Development Method in use on a specific project.

Ministry of Defence (UK)

The UK Ministry of Defence in its case study shows how the Architecture Development Methodology can be used to develop an organization-specific architectural framework, allowing independent operating divisions to develop their own architectures while ensuring that they share a common core operating environment.

NATO (Belgium)

NATO's example illustrates the development of target and transitional architectures in the context of overall enterprise architecture.

Police IT Organization (UK)

The UK Police IT Organization (PITO) used TOGAF as the basis for the Technical Architecture for its National Strategy for Police Information Systems (NSPIS).

The NSPIS case study includes a comprehensive approach to architecture views, and a methodology for ensuring interoperability between the building blocks of the final architecture.

QA Consulting

QA Consulting are a UK consultancy and training company whose mission is to improves IT effectiveness within large organisations. The company's focus is on enhancing the skills of people - optimising human capital investment, and providing IT expertise that complements in-house skills. The company is the UK's No.1 IT Trainer, with 400+ Courses, both Instructor-lead and Internet-based, covering both technical and business skills.

The QA case study explains the company's apporach to using TOGAF, and gives an example of a recent client engagement in the Travel industry using TOGAF.

Statskonsult (Norway)

Statskonsult is the Department of IT Planning & Co-ordination, in the Directorate for Public Management, within the Norwegian Government. It used parts of the TOGAF Architecture Development Method to develop an architecture for an IT infrastructure for the public sector in Norway.

Formats:

- The case study explains the general background to the project.
- The use of the TOGAF Architecture Development Methodology is explained in a separate presentation (PDF).

Westpac (Australia)

Westpac is a major Australian bank who have used TOGAF in collaboration with IBM, in much the same way as the UK Department of Social Security - i.e., as the basis of managing the technology components of a major outsourcing relationship.

- The Westpac Case Study gives an overview of the approach used and reactions.

Glossary

The TOGAF Glossary is intended to define terms essential to the understanding of TOGAF. It is not intended as a general purpose open systems glossary and does not contain terms considered to be in common use.

AC
Access Control

ACSE
Association Control Service Element

Ada
High-level computer programming language developed by the US Department of Defense (DoD). Ada is used as the standard programming language for DoD. It is used for real-time processing, is modular in nature, and includes object-oriented features.

ANSI
American National Standards Institute

API
See Application Program Interface

APP
See Application Portability Profile

Application
A classification of computer programs designed to perform specific tasks, such as word processing, database management, or graphics.

Application Platform
The collection of hardware and software components that provide the services used by support and mission-mission-specific software applications.

Application Portability Profile (APP)
The NIST APP is the structure that integrates US Federal, national, international, and other specifications to provide the functionality necessary to accommodate the broad range of US Federal information technology requirements.

Application Program Interface (API)
(1) The interface, or set of functions, between the application software and the application platform.
(2) The most common means by which an software programmer invokes other software functions.

Application Software
Software entities which have a specific business purpose.

APSE
Ada Programming Support Environment

Architectural Framework
A tool for assisting in the production of organization-specific architectures. An architectural framework consists of a technical reference model, a method for architecture development and a list of component standards, specifications, products and their interrelationships which can be used to build up architectures.

Architecture
Architecture has two meanings depending upon its contextual usage.
(1) A formal description of a system, or a detailed plan of the system at component level to guide its implementation.
(2) The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time.

Architecture, Baseline
The existing system architecture before entering a cycle of architecture review and redesign.

Architecture Continuum
A part of the Enterprise Continuum. The Architecture Continuum provides a repository of architectural elements with increasing detail and specialization. This Continuum begins with foundational definitions like reference models, core strategies, and basic building blocks. From there it spans to industry architectures and all the way to an organization's specific architecture.

Architecture, Database
The logical view of the data models, data standards, and data structure. It includes a definition of the physical databases for the information system, their performance requirements, and their geographical distribution.

Architecture, Target
Depicts the configuration of the target information system.

Architecture View
A perspective from which an architecture may be viewed in order to ensure that a specific topic is considered in a coherent manner - e.g. Security.

ASN
Abstract Syntax Notation

Availability
The probability that system functional capabilities are ready for use by a user at any time, where all time is considered, including operations, repair, administration, and logistic time. Availability is further defined by system category for both routine and priority operations.

Top

Base-level functions
Initial or basic functions.

Baseline
A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development and that can be changed only through formal change control procedures or a type of procedure such as configuration management.

Batch processing
Processing data or the accomplishment of jobs accumulated in advance in such a manner that each accumulation thus formed is processed or accomplished in the same computer run.

Business system
Hardware, software, policy statements, procedures and people which together implement a business function.

Top

CCITT
Consultative Committee on International Telegraph and Telephone

Client
An application component which requests services from a server.

CMIS
Common Management Information Service

CMIP
Common Management Information Protocol

COBOL
Acronym for Common Business-Oriented Language. COBOL is a computer programming language used extensively in mainframes and minicomputers for business applications.

Communications mechanism
Hardware and software functions which allow Application Platforms to exchange information.

Communications network
A set of products, concepts, and services, that enable the connection of computer systems for the purpose of transmitting data and other forms (e.g., voice and video) between the systems.

Communications node
A node that is either internal to the communications network (e.g., routers, bridges, or repeaters) or located between the end device and the communications network to operate as a gateway.

Communications system
A set of assets (transmission media, switching nodes, interfaces, and control devices), that will establish linkage between users and devices.

Configuration management
A discipline applying technical and administrative direction and surveillance to:
(a) identify and document the functional and physical characteristics of a configuration item,
(b) control changes to those characteristics and,
(c) record and report changes to processing and implementation status.

Connectivity Service
A service area of the External Environment entity of the Technical Reference Model that provides end-to-end connectivity for communications through three transport levels (global, regional, and local). It provides general and applications-applications-specific services to platform end devices.

CORBA
Common Object Request Broker Architecture

Top

Data Dictionary
A specialized type of database containing metadata, which is managed by a data dictionary system; a repository of information describing the characteristics of data used to design, monitor, document, protect, and control data in information systems and databases; an application of data dictionary systems.

Data Element
A basic unit of information having a meaning and that may have subcategories (data items) of distinct units and values.

Database
Structured or organized collection of information, which may be accessed by the computer.

Database management system
Computer application program that accesses or manipulates the database.

Data Interchange Service
A service of the Platform entity of the Technical Reference Model that provides specialized support for the interchange of data between applications on the same or different platforms.

Data Management Service
A service of the Platform entity of the Technical Reference Model that provides support for the management, storage, access, and manipulation of data in a database.

DBMS
Database Management System

DCE
Distributed Computing Environment

DDL
Data Definition Language

Default
Command which is automatically executed if none is specifically indicated.

Directory Service
Part of the network services of the Application Platform entity of the Technical Reference Model that provides locator services that are restricted to finding the location of a service, location of data, or translation of a common name into a network specific address. It is analogous to telephone books and supports distributed directory implementations.

DISA
US Department of Defense Information Systems Agency

Distributed Database
(1) A database that is not stored in a central location but is dispersed over a network of interconnected computers.
(2) A database under the overall control of a central database management system but whose storage devices are not all attached to the same processor.
(3) A database that is physically located in two or more distinct locations.

DMF
Data Management Facility

Top

ECMA
European Computer Manufacturers Association

EDI
Electronic Data Interchange

EEI
External Environment Interface

End user
Person who ultimately uses the computer application or output.

Enterprise
The highest level in an organization ----includes all missions and functions.

Enterprise Continuum
Comprises two complementary concepts: the Architecture Continuum and the Solutions Continuum. Together these are a range of definitions with increasing specificity, from foundational definitions and agreed enterprise strategies all the way to architectures and implementations in specific organizations. Such coexistence of abstraction and concreteness in an enterprise can be a real source of confusion. The Enterprise Continuum also doubles as a powerful tool to turn confusion and resulting conflicts into progress.

Enterprise Model
A high level model of an organization's mission, function, and information architecture. The model consists of a function model and a data model.

ES
End system
Expand
Ability to resize objects to produce better organization of on-screen material, usually a graphic or a window.
External Environment Interface (EEI)
The interface that supports information transfer between the application platform and the external environment.

Top

File
Any specifically identified collection of information stored in the computer.
FIPS
Federal Information Processing Standard.
FORTRAN
Acronym for FORmula TRANslator, which is a high level computer language used extensively in scientific and engineering applications.
FTAM
File Transfer, Access, and Management
Function
A useful capability provided by one or more components of a system.

Top

GNMP
Government Network Management Profile
GOSIP
Government Open System Interconnection Profile
GSS
General Security Service
GUI
Graphical User Interface

Top

Hardware
(1) Physical equipment, as opposed to programs, procedures, rules, and associated documentation.
(2) Contrast with software.
Human-Computer Interface (HCI)
Human Computer Interface Hardware and software allowing information exchange between the user and the computer.

Top

IEC
The International Electrotechnical Commission, the international standards body which is responsible for electrical standards.
IEEE
Institute of Electrical and Electronic Engineers.
Information
Any communication or representation of knowledge such as facts, data, or opinions, in any medium or form, including textual, numerical, graphic, cartographic, narrative, or audio-visual forms.
Information Domain

A set of commonly and unambiguously labeled information objects with a common security policy that defines the protections to be afforded the objects by authorized users and information management systems.

Information System

The computer-based portion of a business system.

Information Technology (IT)

The technology included in hardware and software used for information, regardless of the technology involved, whether computers, communications, micro graphics, or others.

Interface

Interconnection and interrelationships between two devices, two applications, or the user and an application or device.

Interoperability

(1) The ability of two or more systems or components to exchange and use information.
(2) The ability of systems to provide and receive services from other systems and to use the services so interchanged to enable them to operate effectively together.

IS

Information System

ISA

Information System Architecture

ISO

International Standards Organization

IT

Information Technology

Top

JTC1

A Joint Technical Committee established by ISO and IEC to take responsibility for their shared interests in IT standardization.

Top

LAN

Local Area Network

Life cycle

The period of time that begins when a system is conceived and ends when the system is no longer available for use.

Top

Metaview (also known as a *viewpoint*)

A specification of the conventions for constructing and using a view. A metaview acts as a pattern or template of the view, from which to develop individual views. A metaview establishes the purposes and audience for a view, the ways in which the view is documented (e.g., for visual modeling), and the ways in which it is used (e.g., for analysis).

MIS

Management Information Systems

MLS

Multilevel Security

MTA

Message Transfer Agent

Multimedia service

A service of the Technical Reference Model that provides the capability to manipulate and manage information products consisting of text, graphics, images, video, and audio.

Top

NIST
US National Institute of Standards and Technology
NLSP
Network Layer Security Protocol

Top

ODA
Office Document Architecture
ODIF
Office Document Interchange Format
OIW
OSI Implementors' Workshop
OODBMS
Object-Oriented Database Management System
Open specifications
Public specifications that are maintained by an open, public consensus process to accommodate new technologies over time and that are consistent with international standards.
Open system
A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software: (a) to be ported with minimal changes across a wide range of systems, (b) to interoperate with other applications on local and remote systems, and (c) to interact with users in a style that facilitates user portability.
Open systems environment (OSE)
The comprehensive set of interfaces, services, and supporting formats, plus user aspects for interoperability or for portability of applications, data, or people, as specified by information technology standards and profiles.
Operating system service
A core service of the Application Platform entity of the Technical Reference Model that is needed to operate and administer the application platform and provide an interface between the application software and the platform (e.g., file management, input/output, print spoolers).
ORB
Object Request Broker
OS
Operating System
OSE
Open System Environment
OSI
Open Systems Interconnection

Top

PEX
PHIGS Extension to X Windows
PHIGS
Programmer's Hierarchical Interactive Graphics System
Platform
See Application Platform.
Portability
(1) The ease with which a system or component can be transferred from one hardware or software environment to another.
(2) A quality metric that can be used to measure the relative effort to transport the software for use in another environment or to convert software for use in another operating environment, hardware configuration, or software system environment.
(3) The ease with which a system, component, data, or user can be transferred from one hardware or software environment to another.
POSIX
Portable Operating System Interface (for Computer Environments)
Profile

A set of one or more base standards, and, where applicable, the identification of those classes, subsets, options, and parameters of those base standards, necessary for accomplishing a particular function.

Profiling
Selecting standards for a particular application.

Top

RDA
Remote Database Access

RDBMS
Relational Database Management System

Repository
A system that manages all of the data of an enterprise, including data and process models and other enterprise information. Hence, the data in a repository is much more extensive than that in a Data Dictionary, which generally defines only the data making up a database.

RM
Reference Model

RPC
Remote Procedure Call

Top

Scalability
The ability to use the same application software on many different classes of hardware/software platforms from personal computers to super computers (extends the portability concept). The capability to grow to accommodate increased work loads.

Security
Services which protect data, ensuring its confidentiality, availability and integrity.

Server
An application component which responds to requests from a client.

SGML
Standard Generalized Markup Language

SMAP
Security Management Application Process

SMTP
Simple Mail Transfer Protocol

SNA
System Network Architecture

SNMP
Simple Network Management Protocol

Solutions Continuum
A part of the Enterprise Continuum. The Solutions Continuum contains implementations of the corresponding definitions in the Architecture Continuum. In this way it becomes a repository of re-useable solutions for future implementation efforts.

SQL
Structured Query Language

SWG
Special Working Group

System
A collection of components organized to accomplish a specific function or set of functions (taken from Draft Recommended Practice for Architectural Description IEEE P1471/D5.2).

System and network management service
A cross-category service of the Application Platform Entity of the Technical Reference Model that provides for the administration of the overall information system. These services include the management of information, processors, networks, configurations, accounting, and performance.

System stakeholder
An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system (taken from ANSI/IEEE Std 1471-2000).

Top

TAFIM
Technical Architecture Framework for Information Management

Taxonomy of architecture views
the organized collection of all views pertinent to an architecture.

TCP/IP
Transmission Control Protocol/Internet Protocol

TCSEC
Trusted Computer System Evaluation Criteria

Technical Reference Model
A structure which allows the components of an information system to be described in a consistent manner.

TFA
Transparent File Access

TLSP
Transport Layer Security Protocol

TNI
Trusted Network Interpretation

TP
Transaction processing

Transaction
Interaction between a user and a computer in which the user inputs a command to receive a specific result from the computer.

Transaction sequence
Order of transactions required to accomplish the desired results.

TRM
Technical Reference Model

TSIG
Trusted Systems Interoperability Group

Top

UIDL
User Interface Definition Language

UIMS
User Interface Management System

UISRM
User Interface System Reference Model

User
(1) Any person, organization, or functional unit that uses the services of an information processing system.
(2) In a conceptual schema language, any person or any thing that may issue or receive commands and messages to or from the information system.

User Interface Service
A service of the Application Platform entity of the Technical Reference Model that supports direct human-machine interaction by controlling the environment in which users interact with applications.

Top

View
A representation of a whole system from the perspective of a related set of concerns.

Viewpoint (also known as a *metaview*)
A specification of the conventions for constructing and using a view. A metaview acts as a pattern or template of the view, from which to develop individual views. A metaview establishes the purposes and audience for a view, the ways in which the view is documented (e.g., for visual modeling), and the ways in which it is used (e.g., for analysis).

Top

WAN
Wide Area Network

Top

Other Architectures and Architectural Frameworks

Introduction

TOGAF is one of a number of architectures and architectural frameworks in use today. Many of the other architectural initiatives have a good deal in common with TOGAF. In the documents linked in the following list, these initiatives are briefly described and their relationships to the TOGAF elements are explored.

To go to a specific architectural framework, follow the hyperlink from the list below.

Alternatively, to browse a number of architectures or frameworks, you may find it more convenient to use the hyperlinks in the Contents frame opposite.

- The C4ISR Architecture Framework gives comprehensive architectural guidance for the various Commands, Services, and Agencies within the U.S. Department of Defense, in order to ensure interoperable and cost effective military systems. It is a successor to the TAFIM, which was officially withdrawn in January 2000.
- The Object Management Group's Common Object Request Broker Architecture (CORBA) is an object-oriented architecture designed to support distributed computing and application integration.
- The Practical Guide to Federal Enterprise Architecture provide guidance to U.S. federal agencies in initiating, developing, using, and maintaining their enterprise architectures.
- The Federal Enterprise Architecture Framework (FEAF) provides direction and guidance to U.S. federal agencies for structuring an enterprise architecture.
- ISO/IEC 14252 (IEEE Std 1003.0) is an architectural framework built on open systems standards.
- The NCR Enterprise Architecture Framework (opens in new window) is based on NCR's architecture practice Global Information Technology Planning (GITP), and NCR's architecture model Open Cooperative Computing Architecture (OCCA 6). The NCR Enterprise Architecture Framework was created to guide the development of systems, industry, and customer specific architectures.
- The ISO Reference Model for Open Distributed Processing (RM-ODP) is a co-ordinating framework for the standardization of Open Distributed Processing. It creates an architecture within which support of distribution, interworking and portability can be integrated.
- The SPIRIT Platform Blueprint is a specification developed by the Network Management Forum, which aims to provide a common, agreed set of specifications for a general purpose computing platform.
- The US Department of Defense Technical Architecture Framework for Information Management (TAFIM) is based on IEEE 1003.0, greatly extended to include a methodology for architecture development.

C4ISR Architecture Framework

Overview

The acronym **C4ISR** stands for **C**ommand, **C**ontrol, **C**omputers, **C**ommunications (**C4**), **I**ntelligence, **S**urveillance, and **R**econnaissance (**ISR**). The C4ISR Architecture Framework Version 2.0 is a framework giving comprehensive architectural guidance for all of these related DoD domains, in order to ensure interoperable and cost effective military systems. It has emerged in recent years as a successor to TAFIM, which was officially withdrawn in January 2000.

The Framework is under revision to generalize it to apply to all functional areas of the Department of Defense. It is already being used in government areas beyond the Defense sector.

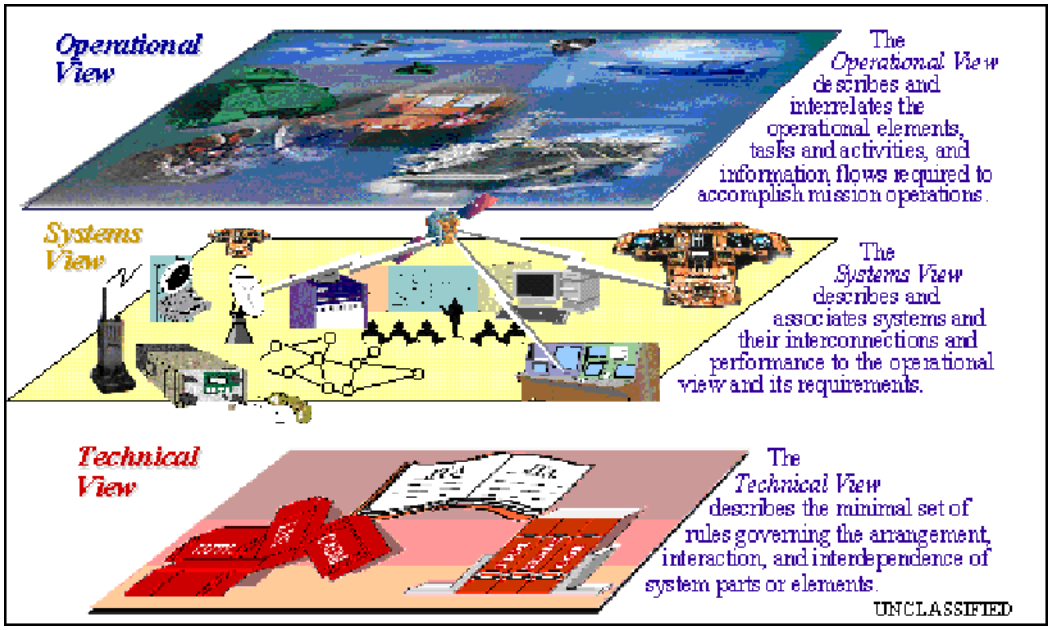


Figure 1: C4ISR Architecture Framework - Three Views of an Architecture

The impetus for the Framework was the realization within the US Department of Defense that DoD organizations across the world were developing architectures representing specific contributions and relationships with respect to overall DoD operations, but that significant differences in content and formats were inhibiting the ability to rationalize or compare different architecture descriptions. In turn, disparate and unrelatable architecture products were leading to non-integrated, non-interoperable, and non-cost effective capabilities in the field.

The C4ISR Architecture Framework is intended to ensure that the architecture descriptions developed by the various Commands, Services, and Agencies within DoD are interrelatable between and among each organization's operational, systems, and technical architecture views, and are comparable and integratable across Joint and multi-national organizational boundaries.

In particular, the Framework:

- Assures that architectures are integratable across the Defense community
- Establishes linkages or threads that tie together the operational, systems, and technical views of an architecture
- Provides the basis for an audit trail that relates current and postulated systems to measures of effectiveness for mission operations

Relationship to TOGAF

Whereas its Architecture Development Method forms a core part of TOGAF, guidance in the C4ISR Framework concerning the process of describing an architecture, i.e., what steps to perform and in what order, has intentionally been kept to a minimum. There are several reasons for this:

- The decision as to which products to build, beyond the essential ones, depends on the purpose of the architecture description
- The sequence in which the products are built likewise depends on the purpose of the architecture description, and on other programmatic factors as well
- Many DoD organizations use their own tailored processes, which the Framework can complement

The most critical aspects of the guidance is that the purpose for building the architecture description should be clearly understood and articulated up front. This purpose will influence the choice of what information to gather, what products to build, and what kinds of analysis to apply.

A major structural concept in the C4ISR Architecture Framework is the three sets of "Views" (Operational, System, Technical). The use of the term "view" in the C4ISR Architecture Framework is different from the use of the term in TOGAF, although there is a rough correspondence:

- between the Operational View in the C4ISR Architecture Framework, and the Business Architecture in TOGAF

("Operations" in the military sphere is the "business" being undertaken by the military "enterprise"); and
- between the System View in the C4ISR Architecture Framework, and the Information System Architectures (Data and Applications Architecture) and Technology Architecture in TOGAF.
- between the Technical View in the C4ISR Architecture Framework, and the Standards View of the Technology Architecture in TOGAF.

A number of the products (deliverables / artifacts) defined in the C4ISR Architecture Framework have no equivalent in the TOGAF ADM because the ADM does not go down to the level of detail that these particular C4ISR products address. These C4ISR products at the level of system design rather than architecture. Specific organizations tailoring the ADM to their own needs might decide to go this level of detail, however (typically, by including them in the Target Architecture step in Phases B, C, and D).

CORBA

Overview

The Object Management Group's Object Management Architecture (OMA), often loosely referred to as the CORBA architecture, is an object-oriented application architecture centered on the concept of an Object Request Broker (ORB). The Object Request Broker acts as a switching center, locating objects, storing interface definitions and object implementations, and relaying messages between objects in a distributed heterogeneous environment.

CORBAservices are a low-level set of common object services available to all objects, covering functions like object creation and deletion, naming, security services and many others.

Horizontal CORBAfacilities are higher-level functions such as distributed documents or printing, suitable for use in a wide variety of market sectors.

Domain CORBAfacilities are vertical market-specific interfaces which will provide common facilities for applications within a particular market sector.

Relationship to TOGAF

The CORBA architecture, or OMA, is an application-level architecture which focuses exclusively on issues affecting distributed object-oriented systems. It is entirely consistent with TOGAF, and depends on the presence of lower-level facilities such as those described by TOGAF for operating system support, communications and so on.

Object-based service categories in TOGAF are called out in the section Object-Oriented Provision of Services.

Enterprise Architecture Planning (EAP)

Steven Spewak's Enterprise Architecture Planning (EAP) is a set of methods for planning the development of information, applications, and technology architectures (the recommended approach being to develop them in that order), and for aligning the three types of architecture with respect to each other. The goal is to ensure that such architectures form the blueprints for sound, implementable systems that solve real business problems.

The overall EAP methodology involves the following "steps":

1. **Planning Initiation**: Defining scope, objectives, roles and responsibilities, and deciding which methodology to use, who should be involved, and what toolset to use. This leads to producing a workplan for the Enterprise Architecture Planning activity and securing management commitment to go through all of the following phases.
2. **Principles**: Developing the core principles to support the effective governance of information and technology. These principles form the basis for making architectural decisions, accepting the results, and managing the migration. They are based on industry best practice and the enterprise's purpose, vision and values, and are implemented through policies, procedures, and standards.
3. **Business Modeling**: Modeling the current business activities and the information used, and identifying business process improvement opportunities.

4. **Current Systems & Technology**: Defining what is in place today for application systems and supporting technology platforms. This is a summary-level inventory of application systems, data, and technology platforms to provide a baseline for long-range migration plans.
5. **Data Architecture**: Developing the data architecture, including defining the major business activities and data objects needed to support the business.
6. **Applications Architecture**: Defining the major kinds of applications needed to manage the data and support the business functions.
7. **Technology Architecture**: Defines the platforms needed to provide a technological infrastructure for the applications that manage the data and support the business functions.
8. **Implementation/Migration Plans**: Defines the sequence for implementing applications, a schedule for implementation, a cost/benefit analysis, and a clear step-by-step path for migration. Executive-level recommendations are made for implementing the plan, and a plan is developed for the transition period after following the Enterprise Architecture Planning activity.
9. **Planning Conclusion**: Final report and presentation of the results to management.

The EAP methodology thus positions the four types of "architecture" in the sequence: Business Architecture, Data Architecture, Applications Architecture, and IT (or Technology) Architecture as the recommended sequence.

Comparison with TOGAF

Spewak's EAP methodology is analogous to TOGAF's Architecture Development Method:

- Steps 1, 2 and 3 map to the development of a "Business Architecture" and the underlying business and technical requirements in Phase B of the TOGAF ADM;
- Step 4 maps to the first step, Baseline Description, in ADM Phases C and D;
- Steps 5, 6 and 7 map directly to the Target Architecture related steps of ADM Phases C and D;
- Step 8 maps to ADM Phase E Opportunities and Solutions and Phase F Migration Planning.

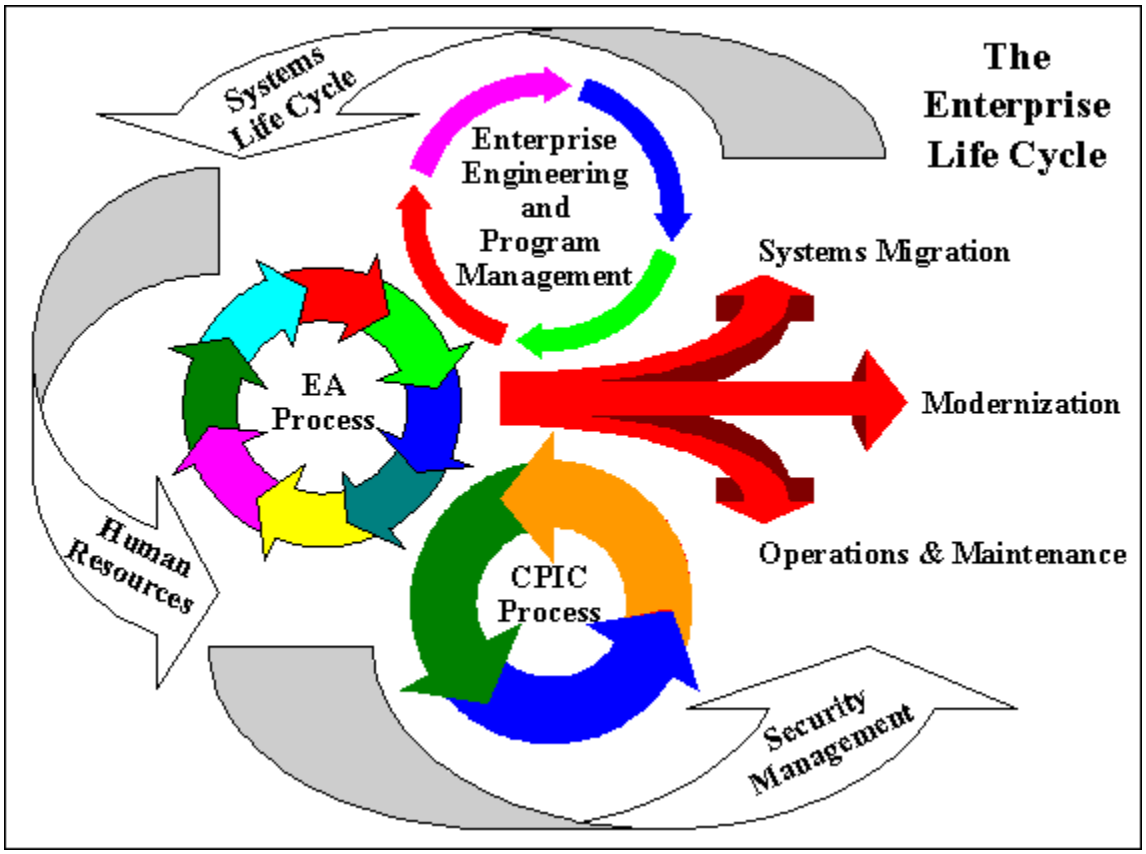
EAP does not have a taxonomy of the various viewpoints and views, or a Foundation Architecture (Technical Reference Model and Standards Information Base).

Federal Enterprise Architecture - Practical Guide

Overview

The US Federal CIO Council published "*A Practical Guide to Federal Enterprise Architecture*", Version 1.0, in February 2001, in a cooperative venture with the General Accounting Office (GAO) and the Office of Management and Budget (OMB). The purpose of this document is to provide guidance to U.S. federal agencies in initiating, developing, using, and maintaining their enterprise architectures (EA). This guide offers an end-to-end process to initiate, implement, and sustain an enterprise architecture program, and describes the necessary roles and responsibilities for a successful enterprise architecture program. The guidance presented in the practical guide should be tailored by each federal agency according to its needs.

This guide focuses on EA processes, products, and roles and responsibilities. The guide addresses how enterprise architecture processes fit within an overall enterprise life cycle: namely, by describing in detail how the enterprise architecture processes relate to enterprise engineering, program management, and Capital Planning and Investment Control (CPIC) processes, as summarized in the following diagram:



At the initiation of its enterprise architecture program, each Agency should establish the scope of its enterprise architecture and formulate a strategy that includes the definition of a vision, objectives, and principles. The following figure summarizes the enterprise architecture processes.

Executive buy-in and support should be established and an architectural team formed within the organization. The team defines an approach and process tailored to agency needs. The architecture team implements the process to build both the baseline and target enterprise architectures. The architecture team also prepares a sequencing plan for transitioning systems, applications, and associated business practices, based on gap analyses and business drivers. Projects are selected and controlled in the CPIC and the enterprise engineering and program management processes and are guided by, and compliant with the enterprise architecture. Lastly, the architecture is maintained through a continuous modification to reflect the Agency's current baseline and target business practices, organizational goals, visions, technology, and infrastructure.

Relationship to TOGAF

The Federal Practical Guide's enterprise architecture processes closely align with the life cycle phases of the TOGAF ADM. In addition, the Practical Guide adds steps such as establishing an Enterprise Architecture Policy and Principles.

Federal Enterprise Architecture Framework (FEAF)

Overview

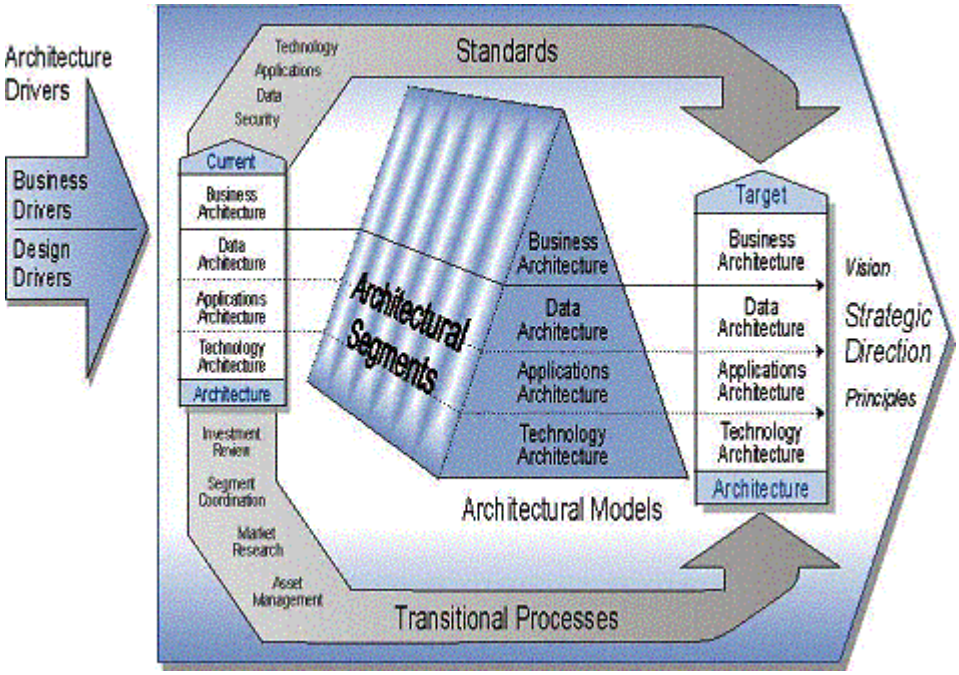
The US Federal CIO Council published the Federal Enterprise Architecture Framework (FEAF) [PDF, HTML], Version 1.1, in September 1999. The FEAF promotes shared development for U.S. federal processes, interoperability, and sharing of information among U.S. federal agencies and other governmental entities. The FEAF provides direction and guidance to Federal agencies for structuring an enterprise architecture. The FEAF describes eight components of an enterprise architecture:

- Architecture Drivers
- Strategic Direction
- Current Architecture
- Target Architecture

- Transitional Processes
- Architectural Segments
- Architectural Models
- Standards

The FEAF also provides direction for establishing "Federal segments", which are cross-agency business areas (such as international trade, grants, common patient records) that transcend federal agency boundaries. These federal architectural segments collectively constitute the Federal Enterprise Architecture.

The FEAF partitions a given architecture into business, data, applications, and technology architectures, as shown in the following figure. The FEAF currently includes the first three columns of the Zachman Framework and the Spewak Enterprise Architecture Planning (EAP) methodology.



The following figure shows the FEAF Architecture Matrix, which names the enterprise architecture products to be developed for each cell. Version 1.1 of the FEAF does not prescribe the contents or approach for developing these work products.

	Data Architecture	Application Architecture	Technology Architecture
Planner Perspective	List of Business Objects	List of Business Processes	List of Business Locations
Owner Perspective	Semantic Model	Business Process Model	Business Logistics System
Designer Perspective	Logical Data Model	Application Architecture	System Geographic Deployment Architecture
Builder Perspective	Physical Data Model	Systems Design	Technology Architecture
Subcontractor Perspective	Data Dictionary	Programs	Network Architecture

Relationship to TOGAF

The FEAF contains guidance analogous to the TOGAF Foundation Architecture and architecture viewpoints and views. The rows of the FEAF matrix (which correspond to the Zachman Framework) do not directly map to the TOGAF structure, although the ADM mapping to the Zachman Framework supports a close correlation between the FEAF and TOGAF. The columns of the FEAF matrix correspond directly to three of the four architecture domains covered by TOGAF (TOGAF also covers Business Architecture).

ISO/IEC 14252 (IEEE Std 1003.0)

Overview

ISO/IEC Technical Report (TR) 14252:1996, Guide to the POSIX Open System Environment, is a direct line ancestor of TOGAF. TOGAF was originally based on the US Department of Defense Technical Architecture Framework for Information Management, or TAFIM, which was itself a development of ISO/IEC TR 14252.

Relationship to TOGAF

At the topmost level, ISO/IEC 14252, TAFIM and TOGAF share a very similar high-level reference model. ISO/IEC TR 14252 does not include a diagram giving a detailed breakdown of the application software, application platform or external environment entities, but the remainder of the document breaks down the application platform into a number of similar, though not identical, areas.

TOGAF includes a considerable amount of material on architecture development which is lacking from ISO/IEC TR 14252, while ISO/IEC TR 14252 includes a considerable amount of detail in service category definition and in the recommended lists of standards and specifications.

The architectural approach taken in ISO/IEC TR 14252 is broadly similar to that of TOGAF, in that the architecture reference model diagram implies no structure among the service categories, leaving that to individual system architectures and real-world implementations.

NCR Enterprise Architecture Framework

The description of the NCR Enterprise Architecture Framework, including a comparison with TOGAF, is hosted on NCR's own web site.

NCR Enterprise Architecture Framework is based on NCR's architecture practice Global Information Technology Planning (GITP), and NCR's architecture model Open Cooperative Computing Architecture (OCCA 6). The Framework was created to guide the development of systems, industry, and customer specific architectures.

ISO Reference Model for Open Distributed Processing (ISO RM-ODP)

Overview

The Reference Model of Open Distributed Processing, ITU-T Rec. X.901 | ISO/IEC 10746-1 to ITU-T Rec. X.904 | ISO/IEC 10746-4, commonly referred to as *RM-ODP*, provides a framework to support the development of standards that will support distributed processing in heterogeneous environments. It is based, as far as possible, on the use of formal description techniques for specification of the architecture.

RM-ODP uses an object modelling approach to describe distributed systems. Two structuring approaches are used to simplify the problems of design in large complex systems: five 'viewpoints' provide different ways of describing the system; and eight 'transparencies' identify specific problems unique to distributed systems which distributed system standards may wish to address. Each viewpoint is associated with a language which can be used to describe systems from that viewpoint.

The five viewpoints described by RM-ODP are:

1. The **enterprise** viewpoint, which examines the system and its environment in the context of the business requirements on the system, its purpose, scope and policies. It deals with aspects of the enterprise such as its organizational structure, which affect the system.

2. The **information** viewpoint, which focuses on the information in the system. How the information is structured, how it

changes, information flows, and the logical divisions between independent functions within the system are all dealt with in the information viewpoint.

3. The **computational** viewpoint, which focuses on functional decomposition of the system into objects which interact at interfaces.

4. The **engineering** viewpoint, which focuses on how distributed interaction between system objects is supported.

5. The **technology** viewpoint, which concentrates on the individual hardware and software components which make up the system.

Relationship to TOGAF

RM-ODP is very tightly focused on problems relating to interactions between the objects making up distributed information processing systems, while TOGAF embraces the full spectrum of systems, whether distributed or not. As such, TOGAF coverage is a superset of that provided by RM-ODP. However, the relationship between the viewpoints described by RM-ODP and the TOGAF views is not an obvious one, and there is some danger of confusing the two.

In fact, each RM-ODP viewpoint is an examination of a distributed system at a different level of detail, while TOGAF adopts the ANSI/IEEE Std 1471-2000 approach to views as representations of a whole system from the perspective of a related set of concerns. This makes it impractical to try to compare the set of TOGAF views with RM-ODP viewpoints.

A better comparison can be made between the RM-ODP viewpoints and the different levels of detail used by TOGAF in the examination of building blocks. In *Building Blocks and the Architecture Development Method*, Figure 2 shows how the iterative process of building block definition can be divided into the Business Process level, the Technical Functionality and Constraints level, the Architectural Model level, and the Solution Model level.

The mapping between the Building Block Example and ODP viewpoints is as follows

Business Process level	**< = = >**	Enterprise and Information viewpoints and Technical Functionality and Constraints level
Architectural model level	**< = = >**	Computational viewpoint
Solution level	**< = = >**	Technology and Engineering viewpoints

---

SPIRIT Platform Blueprint Issue 3.0

Overview

The **S**ervice **P**roviders' **I**ntegrated **R**equirements for **I**nformation **T**echnology (SPIRIT) Platform Blueprint is a specification that was developed within the Network Management Forum, now known as the TeleMangement Forum (TMF). The SPIRIT Platform Blueprint Issue 3.0 is published by The Open Group.

SPIRIT is a joint effort between telecommunication service providers, computer system vendors and independent software vendors, with the goal of producing a common, agreed set of specifications for a general-purpose computing platform. The objective is to provide a core set of specifications for use in each company's purchasing of software components for general-purpose computing platforms. The SPIRIT specifications are based predominantly on widely accepted industry standards.

Relationship to TOGAF

SPIRIT defines a practical, tested selection of specifications, most of which are referenced within the TOGAF Standards Information Base, that achieves portability and interoperability for large-scale systems. The focus of SPIRIT is on ensuring that the SPIRIT selections are agreed upon by the vendor side for implementability and on the user side for procurability.

Technical Architecture Framework for Information Management (TAFIM)

Overview

The US Department of Defense Technical Architecture Framework for Information Management was developed from ISO/IEC Technical Report (TR) 14252 - 1995, *Guide to the POSIX Open System Environment* (IEEE Std 1003.0), and was used as the basis of TOGAF Version 1. As a result there is a strong resemblance between the three.

As of 7th January, 2000, TAFIM has been officially cancelled by the DoD. The TAFIM website has been archived at http://www-library.itsi.disa.mil/tafim.html, where it will continue to be available for information and reference purposes only. A number of new documents have been developed that supersede specific TAFIM subject areas. These are:

- C4ISR Architecture Framework, Version 2.0, 18 Dec 1997
- JTA, Version 3.0, 15 Nov 1999
- DoD TRM, Version 1.0, 5 Nov 1999

Relationship to TOGAF

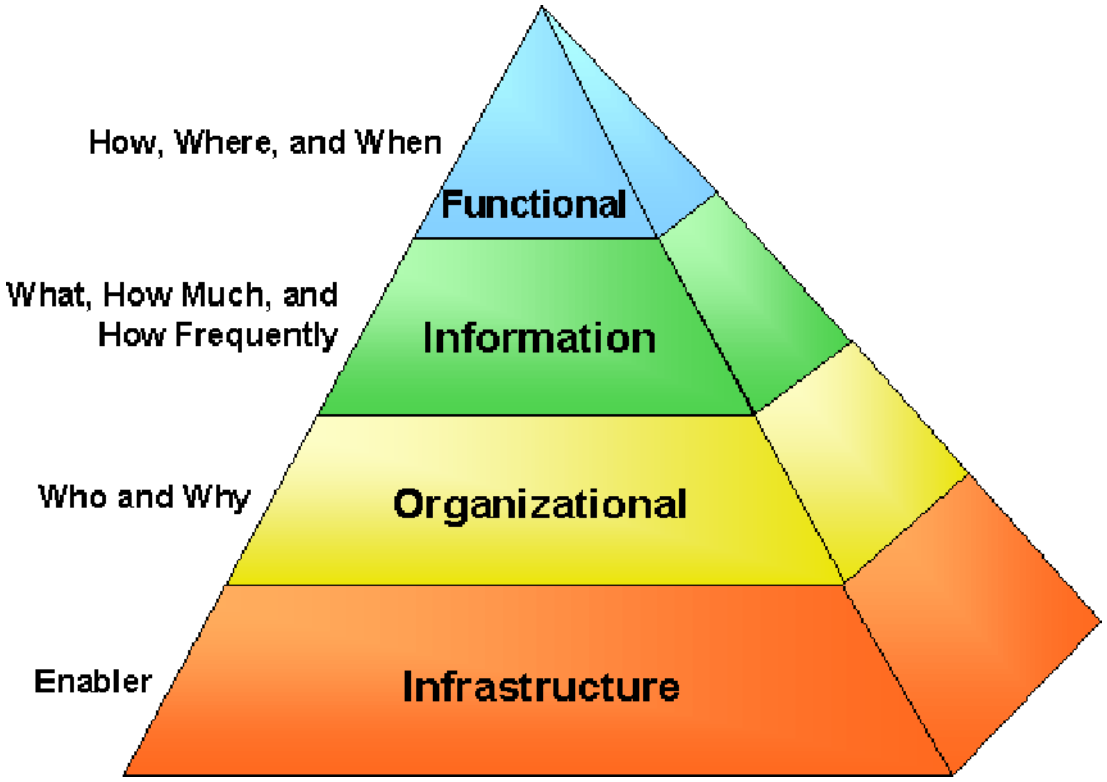
TAFIM was the parent of TOGAF, and the US Defense Information Systems Agency contributed extensively to the development of TOGAF, with the result that the two architectural frameworks have much in common. The TOGAF Technical Reference Model was largely derived from TAFIM, and the TOGAF Architecture Development Method was originally based on parts of TAFIM.

Treasury Enterprise Architecture Framework (TEAF)

Overview

In July 2000, the Department of the Treasury published the Treasury Enterprise Architecture Framework. The TEAF provides (1) guidance to Treasury bureaus concerning the development and evolution of information systems architecture; (2) a unifying concept, common principles, technologies, and standards for information systems; and (3) a template for the development of the enterprise architecture.

The TEAF describes an architectural framework that supports Treasury's business processes in terms of work products. This framework guides the development and redesign of the business processes for various bureaus in order to meet the requirements of recent legislation in a rapidly changing technology environment. The TEAF prescribes architectural views and a set of essential and supporting work products to portray these views. The following figure illustrates the TEAF framework.



The TEAF's functional, information and organizational architecture views collectively model the organization's processes, procedures, and business operations. By grounding the architecture in the business of the organization, the TEAF defines the core business procedures and enterprise processes. Through its explicit models, a TEAF-based architecture enables the identification and reasoning of enterprise- and system-level concerns and investment decisions.

The TEAF separates enterprise architecture information into Enterprise Architecture Direction (drivers, policies, program roadmap), Enterprise Architecture Description, and Enterprise Architecture Accomplishment (transition strategy, technical forecasts and insertion). The Enterprise Architecture Description is a matrix, with columns being views ((functional, information, organizational, and infrastructure) and rows being perspectives (planner, owner, designer, builder). Many of the TEAF work products are also in the C4ISR Architecture Framework. However, TEAF introduces the Information Assurance Trust Model, Information Assurance Risk Assessment, and the Enterprise Architecture Roadmap. The Enterprise Architecture Roadmap summarizes the scope of the bureau's enterprise architecture, the drivers, governance plan, approach and methodology, and program plan.

Relationship to TOGAF

Many of the observations made in describing the relationship of the C4ISR Architecture Framework to TOGAF apply also to the TEAF. Namely, the TEAF is prescriptive as to what work products should be produced, and the term view/perspective is used differently.

Zachman Framework

The Zachman Framework is a framework providing a view of the subjects and models needed to develop a complete Enterprise architecture. This framework is described in detail at the ZIFA web site.

An alternative source is the University of Nebraska at Omaha, which gives a very detailed and highly readable description of the Zachman Framework.

The Zachman Framework is a widely used approach for developing and/or documenting an enterprise-wide information systems architecture. Zachman based his framework on practices in traditional architecture and engineering. This resulted in an approach which on the vertical axis provides multiple perspectives of the overall architecture, and on the horizontal axis a classification of the various artifacts of the architecture.

The purpose of the framework is to provide a basic structure which supports the organization, access, integration, interpretation, development, management, and changing of a set of architectural representations of the organization's information systems. Such objects or descriptions of architectural representations are usually referred to as artifacts.

The framework, then, can contain global plans as well as technical details, lists and charts as well as natural language statements. Any appropriate approach, standard, role, method, technique, or tool may be placed in it. In fact, the framework can be viewed as a tool to organize any form of metadata for the enterprise.

Comparison with TOGAF

The ADM mapping to the Zachman Framework supports a close correlation between the Zachman Framework and the TOGAF ADM.

The Zachman Framework provides a a very comprehensive and well-established taxonomy of the various viewpoints, models and other artifacts that an enterprise may want to consider developing as part of an enterprise architecture. (The recommendation of the Zachman Framework itself is that all the cells be covered.)

The current recommended set of viewpoints in TOGAF does not cover all 30 cells of the Zachman Framework. However, with TOGAF it is possible to develop viewpoints and views to cover other aspects of the Zachman Framework as necessary.

TOGAF recommends some viewpoints that are not included in the Zachman Framework: for example, the Security and Manageability Viewpoints. The selection of viewpoints needs to be determined by the purpose of the architecture, and the TOGAF ADM defines a process for driving that selection.

The vertical axis of the Zachman Framework provides a source of potential viewpoints for the architect to consider. The horizontal axis could be regarded as providing a generic taxonomy of concerns.

The Zachman Framework says nothing about the processes for developing viewpoints or conformant views, or the order in which they should be developed. It does not provide a method such as TOGAF's ADM, or a Foundation Architecture (Technical Reference Model and Standards Information Base).

Tools for Architecture Development

Overview Issues Evaluation Criteria

Overview

As an enterprise architecture framework, TOGAF provides a basis for developing architectures in a uniform and consistent manner. Its purpose in this respect is to ensure that the various architecture descriptions developed within an enterprise, perhaps by different architects or architecture teams, support the comparison and integration of architectures within and across architecture domains (business, data, applications, technology), and relating to different business area scopes within the enterprise.

To support this goal, TOGAF defines numerous deliverables in the form of architectures, represented as architecture models, architecture views of those models, and other artifacts. Over time, these artifacts become a resource that needs to be managed and controlled, particularly with a view to reuse. This concept is referred to in TOGAF as the *Enterprise Continuum*.

Architecture models and views are discussed in detail separately in Part IV. This section discusses considerations in choosing automated tools in order to generate such architecture models and views, and to maintain them over time.

Issues in Tool "Standardization"

In the current state of the tools market, many enterprises developing enterprise architectures struggle with the issue of standardizing on tools, whether they seek a single, "one size fits all" tool or a multi-tool suite for modeling architectures and generating the different architecture views required.

There are ostensible advantages associated with selecting a single tool. Organizations following such a policy can hope to realize benefits such as reduced training, shared licenses, quantity discounts, maintenance, and easier data interchange.

However, there are also reasons for refusing to identify a single mandated tool, including reasons of principle (endorsing a single architecture tool would not encourage competitive commercial innovation or the development of advanced tool capability); and the fact that a single tool would not accommodate a variety of architecture development "maturity levels" and specific needs across an enterprise. Successful enterprise architecture teams are often those that harmonize their architecture tools with their architecture maturity level, team/organizational capabilitities, and objectives or focus. If different organizations within an enterprise are at different architecture maturity levels and have different objectives or focus (e.g., enterprise versus business versus technology architecture), it becomes very difficult for one tool to satisfy all organizations' needs.

Evaluation Criteria and Guidelines

TOGAF does not require or recommend any specific tool. However, in recognition of the problems that enterprise architects currently face in this area, this section provides a set of proposed evaluation criteria for selecting architecture tools to develop the various architecture models and views that are required.

Individual enterprises may wish to adapt these generic evaluation criteria to their particular circumstances and requirements. In particular, such an exercise would typically produce weightings of the various criteria that can be used to produce a "score" for the specific tools evaluated.

Tool Criteria

Functionality

Key Features and Functions

- Does it support the framework that your organization has chosen to use?

    - Does it support production of the deliverables required?
    - If not, does it support some of the known frameworks e.g. TOGAF or Zachman Framework out of the box?
- Glossary
    - Glossary extendable to become a taxonomy?
    - Active Glossary to enforce a taxonomy?
- Ability to represent architecture models and views in a way meaningful to non-technical architecture stakeholders
- Does it support meta-models e.g. ability to configure and tailor models
- Does it support enterprise use e.g. multi-user collaboration support?
- Does it allow drill down? e.g. conceptual, logical, physical, etc.
- Does it provide a mechanism for linking requirements to the resulting enterprise architecture? I.e. requirements traceability
- Security features:
    - Does it facilitate access control e.g. different permissions for different roles?
    - Does its security design support corporate security policies?
- Does it natively support report generation?
- Does it support a common language and notation?

Intuitiveness / Ease-of-use Factors

- An easy to follow "process map" guiding use of the tool
- On-line help
- Relevant out-of-the box various architecture constructs, be it business, data, application, or technology?
- Relevant out-of-the-box templates or patterns for constructs, which can be used to help organizations "jump start"
- Support for visualization modeling - e.g. drag and drop and lines that equate to links
- Can it be extended or customized and does it provide utilities to do that?
- Does it track and audit changes?
- Does it provide a way for consistently naming and organizing those artifacts?
- Can those artifacts/components be easily viewed, used, and reused?
- What requirements are there for use of programmatic languages?

Organizational Compatibility Factors

- Internationalization / Localization Capability
    - Can the tool be used in all the geographic locations and/or language domains in which architecture work is done?

Tool Capacity / Scalability Constraints

- Does the tool have capacity constraints?
    - Size of data
    - Number of files
    - Number of data entries/records?
- What are the tool's design "sweet spots" (i.e., optimal design configuration parameters), and how scalable is it around those optima?
    - Is there an upgrade path beyond the capacity constraints of the tool?

Architecture of the Tool

- Repository distributed or central?
- Dynamic repository?
- Does the tool function with multiple industry standard data stores (e.g. Oracle, Sybase), or is storage proprietary?
- Backwards compatibility with prior releases of the tool
- Does it allow integration and consolidation of data into a central repository?
- Does it include version control?
- Is it accessible through a web client?
- What platforms (hardware, OS, DBMS, network) does it run on?

Full life cycle support

- Does it provide full life-cycle support?
- Does it support various relevant views out-of-the-box? E.g. business process, data, application, technology.
- Does it support the creation of custom views?
- Does it use modeling methods and techniques relevant to this enterprise's architecture practice?
- Does is support simulation?

- Is the model that it produces executable?

Interoperability Factors

- Import/Export
  - Can it create an artifact inside the tool and export it to other commonly used tools, and have the users of those tools use the artifact intact?
  - Can it import an artifact created in another tool, and use the artifact intact?
- Does it integrate with other tools?
- Does it provide and support industry standard APIs?
- Use of relevant industry standards, e.g. XML, HTML, produce hypertext, UML, other industry standard?

Financial Considerations

- What is the acquisition cost?
- What is the total cost of ownership?
  - Maintenance
  - Equipment costs
  - Support costs
  - Number of resources required to keep it up to date
  - Administration responsibilities / time constraints
  - Will there be any impacts of introducing the tool into your environment? E.g. does it require some unique infrastructure?
  - Training
  - Licensing Models

Vendor Factors

- Will vendor remain viable?
- How long has vendor existed in this arena?
- Do they have large customers?
- Do they have professional services?
- Third party support?
- Does tool have history at the organization and if so what is its reputation?
- Training Factors
  - Availability
  - Costs
  - Amount required to become productive
  - Style of learning (CBT, classroom)

General Pointers

- Value of the tool is dependent upon the architecture maturity of the organization.
- Need to match the tool to the capability of your organization i.e. where it is architecturally?
- Trade-off between tactical considerations (competency, familiarity, …) and strategical considerations (overall organization's standards and directions)
- Teaming can positively or negatively affect a tool's success.

Mapping the TOGAF ADM to the Zachman Framework

Introduction

A number of architecture frameworks exist, each of which has its particular advantages and disadvantages, and relevance, for enterprise architecture. Several are discussed in Other Architectures and Frameworks.

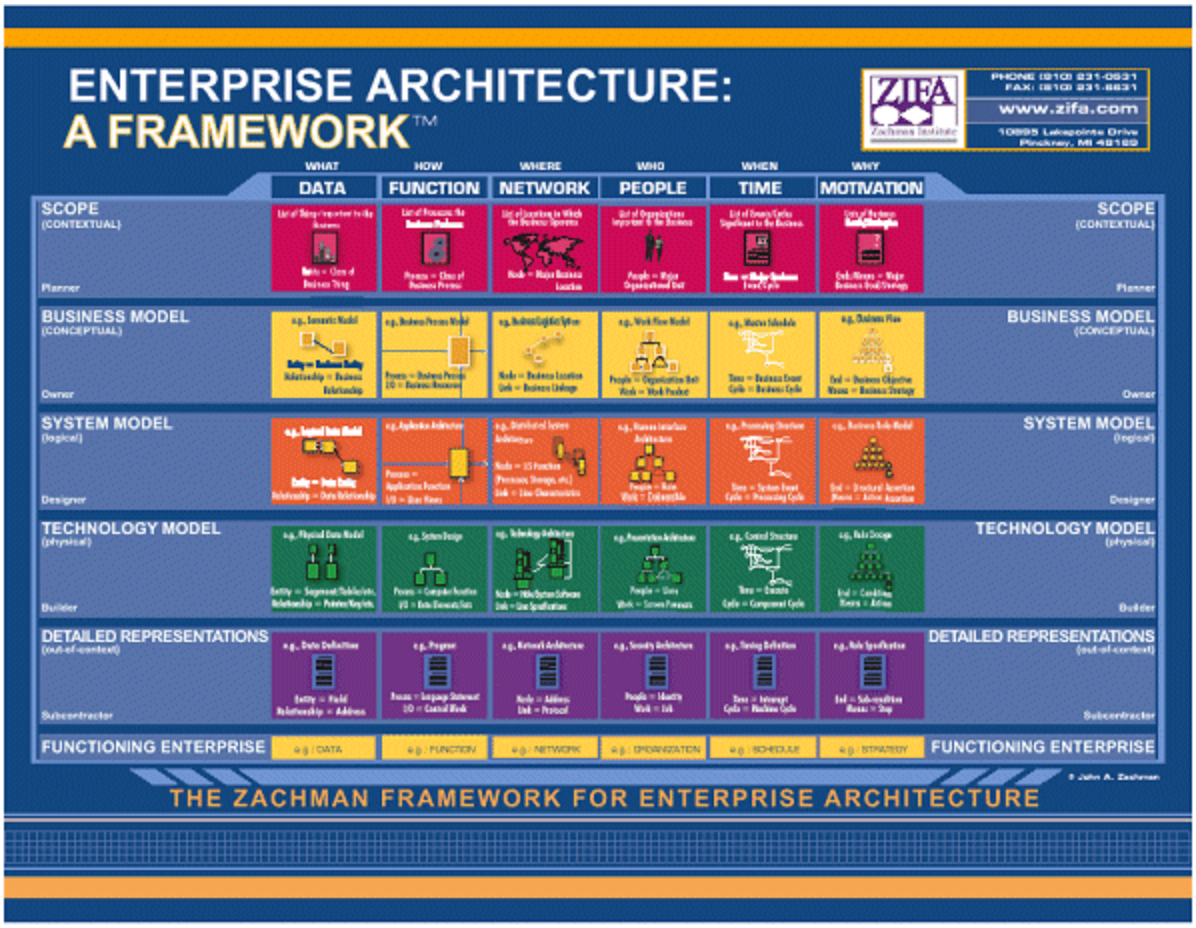
However, there is no accepted industry standard *method* for developing an enterprise architecture. The Open Group's goal with TOGAF is to work towards making the TOGAF Architecture Development Method just such an industry standard method, which can be used for developing the products associated with any recognized enterprise framework that the architect feels is appropriate for a particular architecture. The Open Group's vision for TOGAF is as a vehicle and repository for practical, experience-based information on how to go about the process of enterprise architecture, providing a generic method with which specific sets of deliverables, specific reference models, and other relevant architectural assets, can be integrated.

To illustrate the concept, this section provides a mapping of the various Phases of the TOGAF ADM to the cells of the well known Zachman Framework.

The Zachman Framework

The Zachman Framework for Enterprise Architecture, sometimes simply referred to as "The Zachman Framework", has become a de facto standard for classifying the artifacts developed in enterprise architecture. It is a logical structure for classifying and organizing the design artifacts of an enterprise that are significant to its management. It draws on a classification scheme found in the more mature disciplines of Architecture/Construction and Engineering/Manufacturing, used for classifying and organizing the design artefacts relating to complex physical products such as a building or an aircraft. Zachman adopts this classification scheme to the design and construction of information systems.

The Zachman Framework comprises a 6x6 matrix.



The columns represent various aspects of the enterprise that can be described, or modeled; and the rows represent various viewpoints from which the aspects can be described. Thus each cell formed by the intersection of a column and a row represents an aspect of the enterprise modeled from a particular viewpoint. The architect selects and models the cells that are appropriate to the immediate purpose, with the ultimate objective of modeling all the cells.

The six viewpoints are:

1. The Scope (Contextual) Viewpoint - aimed at the *Planner*
2. The Business Model (Conceptual) Viewpoint - aimed at the *Owner*
3. The System (Logical) Viewpoint - aimed at the *Designer*
4. The Technology (Physical) Viewpoint - aimed at the *Builder*
5. The Detailed Representations (Out-of-Context) Viewpoint - aimed at the *Subcontractor*
6. The Functioning Enterprise Viewpoint

The six aspects - and the interrogatives to which they correspond - are:

1. The Data aspect - What?
2. The Function aspect - How?
3. The Network aspect - Where?
4. The People aspect - Who?
5. The Time aspect - When?
6. The Motivation aspect - Why?

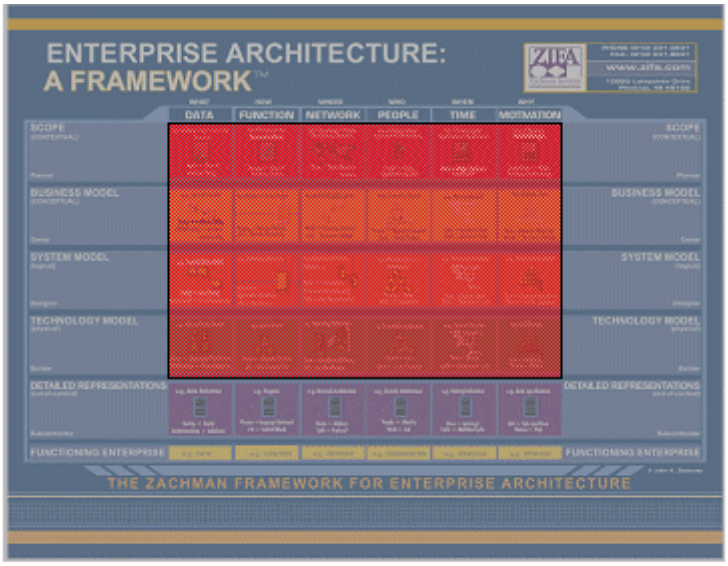
Although the Zachman Framework applies to enterprises, the Framework itself is generic. It is a comprehensive, logical structure for the descriptive representations (i.e. models, or design artefacts) of any complex object, and it does not prescribe or describe any particular method, representation technique, or automated tool.

The strength of the framework is that it provides a way of thinking about an enterprise in an organized way, so that it can be described and analyzed. It also enables the individuals involved in producing enterprise information systems to focus on selected aspects of the system without losing sight of the overall enterprise context. In designing and building complex systems, such as enterprise systems, there are simply too many details and relationships to consider simultaneously. At the same time, isolating single variables and making design decisions out of context results in sub-optimization, with all the

attendant costs and risks. The challenge is the same whether the system is physical, like an aircraft, or conceptual, like an enterprise system. How do you design and build it, piece by piece, and step by step, such that it achieves its purpose without losing its value and raising its cost by optimizing the pieces and sub-optimizing the overall?

Mapping TOGAF to the Zachman Framework

The scope of the four architecture domains of TOGAF aligns very well with the first four rows of the Zachman Framework, as shown in the following mapping of these domains:



Scope of TOGAF ADM

Several domains overlap in the above diagram: the earliest domain to address a cell has precedence in the coloring scheme.

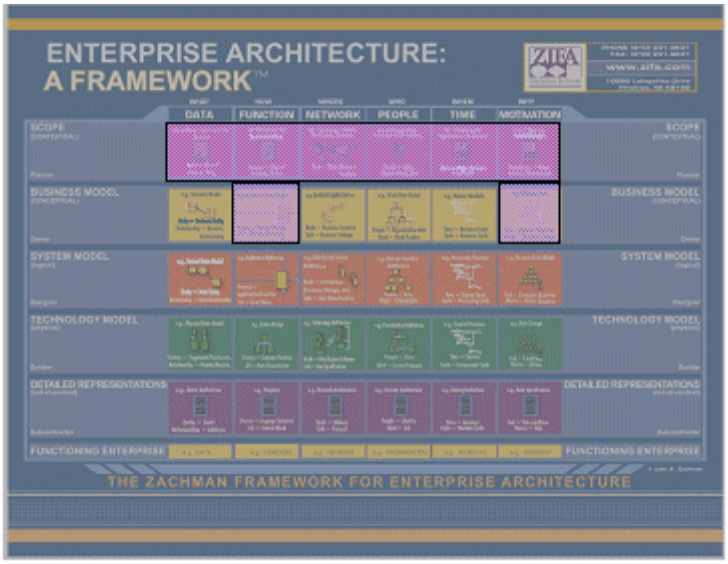
The mappings of the individual phases of the ADM are shown in detail below.

It should be noted that, in addition to the mappings to specific cells given below, the Detailed Representations and Functioning Enterprise viewpoints (the lowest two rows) of the Zachman Model are also addressed and represented in TOGAF, through the Architecture Governance Framework, and through ADM deliverables such as the various Architecture Contracts. These ensure the validity and viability of the delivered solutions to meet the business needs.

Preliminary Phase: Framework and Principles

The outputs of this phase are:

- Framework Definition
  - ZF: Business/Function (model of the architecture development process) [R2,C2]
- Architecture Principles
  - ZF: Scope/Data, Scope/Function, Scope/Network, Scope/People, Scope/Time, Scope/Motivation [R1,C1; R1,C2; R1,C3; R1,C4; R1,C5; R1,C6]
- Restatement of, or reference to, Business Principles, Business Goals and Business Drivers
  - ZF: Composite of: Scope/Motivation, Business/Motivation [R1,C6; R2,C6]

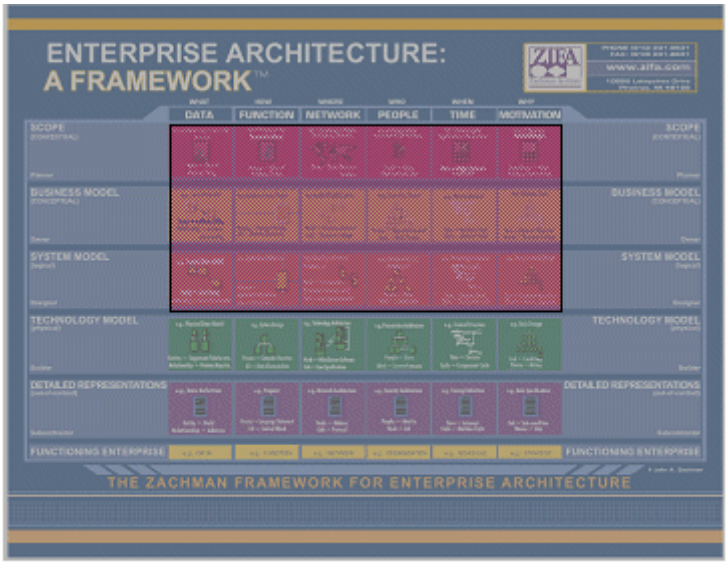


Scope of Preliminary Phase: Framework and Principles

Phase A: Architecture Vision

The outputs of this phase are:

- Approved Statement of Architecture Work / Project Definition, including in particular:
  - Scope and constraints
    - ZF: Scope/Data, Scope/Function, Scope/Network, Scope/People, Scope/Time [R1,C1; R1,C2; R1,C3; R1,C4; R1,C5; R1,C6]
    - Note: The Scope/motivation cell is presumed to be addressed by strategic business planning activites outside the scope of the Architecture Vision
  - Plan for the architecture work
- Refined statements of Business Principles, Business Goals and Strategic Drivers
  - ZF: Scope/Data, Scope/Motivation [R1,C1; R1,C6]
- Architecture Principles (if not previously existing)
  - ZF: Scope/Data, Scope/Function, Scope/Network, Scope/People, Scope/Time, Scope/Motivation [R1,C1; R1,C2; R1,C3; R1,C4; R1,C5; R1,C6]
- Architecture Vision / Business Scenario, including:
  - Business Baseline Version 1
    - ZF: Business/Data, Business/Function, Business/Network, Business/People, Business/Time, Business/Motivation [R2,C2; R2,C2; R2,C3; R2,C4; R2,C5; R2,C6]
  - Technical Baseline Version 1
    - ZF: System/Data, System/Function, System/Network, System/People, System/Time, System/Motivation [R3,C2; R3,C2; R3,C3; R3,C4; R3,C5; R3,C6]
  - Business Architecture Version 1
    - ZF: Business/Data, Business/Function, Business/Network, Business/People, Business/Time, Business/Motivation [R2,C2; R2,C2; R2,C3; R2,C4; R2,C5; R2,C6]
  - Technical Architecture Version 1
    - ZF: System/Data, System/Function, System/Network, System/People, System/Time, System/Motivation [R3,C2; R3,C2; R3,C3; R3,C4; R3,C5; R3,C6]

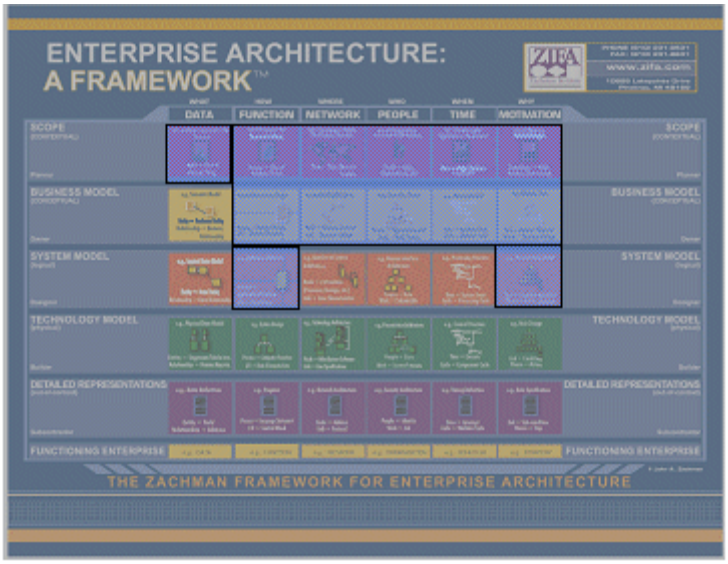


Scope of Phase A: Architecture Vision

Phase B: Business Architecture

The outputs of this phase are:

- Statement of Architecture Work (updated if necessary)
- Validated business principles, business goals, and strategic drivers
    - ZF: Scope/Data, Scope/Function, Scope/Network, Scope/People, Scope/Time [R1,C1; R1,C2; R1,C3; R1,C4; R1,C5; R1,C6]
- Target Business Architecture - Version 2 (detailed)
    - *Organization structure*, identifying business locations and relating them to organizational units.
        - ZF: Scope/Network, Scope/People, Business/Network, Business/People
        - [R1,C3; R1,C4; R2,C3; R2,C4]
    - *Business goals and objectives*, for each organizational unit.
        - ZF: Scope/Network, Scope/Time, Business/Network, Business/Time, Business/Motivation [R1,C3; R1,C5; R2,C3; R2,C5; R2,C6]
            - Note: The Scope/motivation cell is presumed to be addressed by strategic business planning activites outside the scope of the Business Architecture
    - *Business functions*. a detailed, recursive step involving successive decomposition of major functional areas into sub-functions.
        - ZF: Scope/Function, Business/Function [R1,C2; R2,C2]
    - *Business Services* - the services that each enterprise unit provides to its customers, both internally and externally.
        - ZF: Business/Function, System/Function [R2,C2; R3,C2]
    - *Business processes,* including measures and deliverables
        - ZF: Business/Function, Business/Time [R2,C2; R2,C5]
    - *Business roles*, including development and modification of skills requirements.
        - ZF: Scope/People, Business/People [R1,C4; R2,C4]
    - *Correlation of organization and functions*. Relate business functions to organizational units in the form of a matrix report.
        - ZF: Scope/Function, Scope/Network, Scope/People, Business/Function, Business/Network, Business/People [R1,C2; R1,C3; R1,C4; R2,C2; R2,C3; R2,C4]
- Business Baseline - Version 2 (detailed) - if appropriate
- Views corresponding to the selected viewpoints addressing key stakeholder concerns
- Gap analysis results
- Technical requirements (drivers for the Technical Architecture work): identifying, categorising and prioritising the implications for work in the remaining architecture domains; for example, by a dependency/ priority matrix. (For example, guiding trade-off between speed of transaction processing and security.) List the specific models that are expected to be produced (for example, expressed as primitives of the Zachman Framework).
    - ZF: System/Motivation [R3,C6]
- Business Architecture Report
- Updated business requirements



Scope of Phase B: Business Architecture

Note:

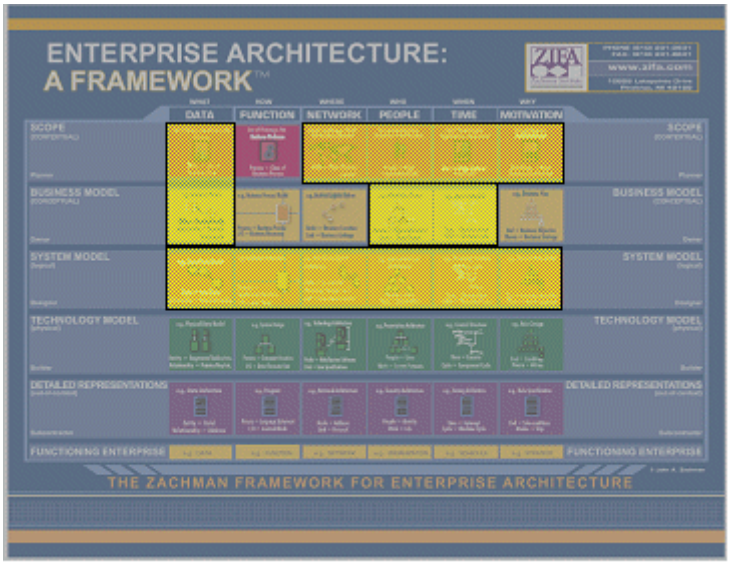
- The Business/Data cell is covered by the Data and Applications Architectures

Phase C: Informations System Architectures: Data Architecture

The outputs of this part of Phase C are:

- Statement of Architecture Work (updated if necessary)
- Data Baseline Description - if appropriate
- Validated Principles, or new Data Principles (if generated here)
    - ZF: Scope/Data, Scope/Network, Scope/People, Scope/Time [R1,C3; R1,C4; R1,C5]
- Target Data Architecture
    - Conceptual data model
        - ZF: Business/Data [R2,C1]
    - Logical data model
        - ZF: System/Data [R3,C1]
    - Data Management Process models
        - ZF: System/Function, System/People [R3,C2; R3,C3]
    - Data entity / business function matrix
        - ZF: Composite of Business/People, System/Data, System/Function, [R2,C4; R3,C1; R3,C2]
    - Data interoperability requirements
        - ZF: Composite of System/Data, System/Function, System/Network, System/People [R3,C1; R3,C2; R3,C3; R3,C4]
- Viewpoints addressing key stakeholder concerns.
- Views corresponding to the selected viewpoints; e.g.:
    - Data dissemination view
        - ZF: Composite of System/Data, System/Function, System/Network, System/People [R3,C1; R3,C2; R3,C3; R3,C4]
    - Data lifecycle view
        - ZF: Composite of System/Data, System/Function, System/Time
    - Data security view
        - ZF: Composite of System/Function, System/Data, System/Network, System/People, System/Time
    - Data model management view
        - ZF: Composite of Business/Data, System/Data, Business/Time, System/Time
- Gap analysis results
- Relevant technical requirements that will apply to this evolution of the architecture development cycle
    - ZF: System/Motivation
- Data Architecture Report, summarizing what was done and the key findings
- Impact Analysis

- Updated business requirements (if appropriate)



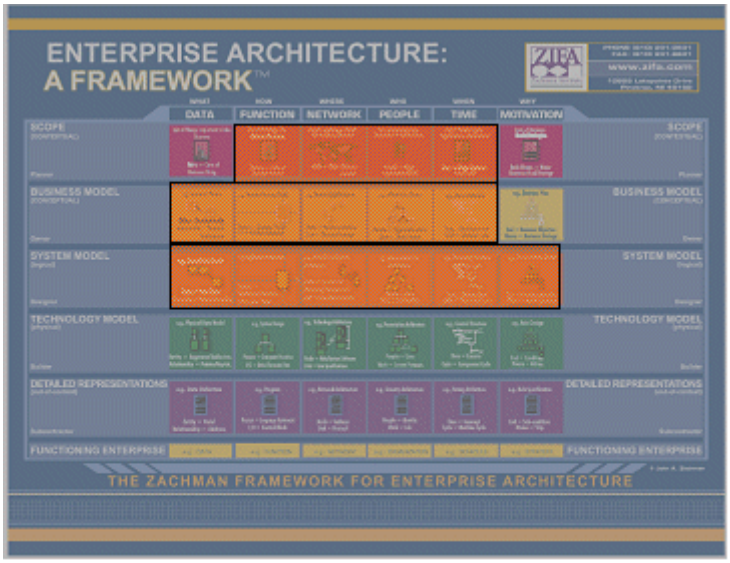
Scope of Phase C: Data Architecture

Phase C: Informations System Architectures: Applications Architecture

The outputs of this part of Phase C are:

- Statement of Architecture Work (updated if necessary)
- Applications Baseline Description - if appropriate
- Validated Applications Principles, or new Applications Principles (if generated here)
    - ZF: Scope/Function, Scope/Network, Scope/People, Scope/Time
- Target Applications Architecture
    - Process Systems Model
        - ZF: System/Function
    - Sytems / Place Model
        - ZF: System/Network
    - People / Systems Model
        - ZF: System/People
    - Systems / Time Model
        - ZF: System/Time
    - Applications interoperability requirements
        - ZF: Composite of System/Data, System/Function, System/Network, System/People, System/Time, System/Motivation
- Viewpoints addressing key stakeholder concerns.
- Views corresponding to the selected viewpoints; e.g.:
    - Common Applications services view
        - ZF: Composite of System/Data, System/Function, System/Network, System/People, System/Time
    - Applications Interoperability view
        - ZF: Composite of System/Data, System/Function, System/Network, System/Time
    - Applications / Information View
        - ZF: Composite of System/Data, System/Function, System/Network, System/Time
    - Applications / User locations View
        - ZF: Composite of System/Network, System/People
- Gap analysis results
    - Areas where the Business Architecture may need to change to cater for changes in the Applications Architecture
        - ZF: Composite of Business/Data, Business/Function, Business/Network, Business/People, Business/Time
    - Identify any areas where the Data Architecture (if generated at this point) may need to change to cater for changes in the Applications Architecture.
        - ZF: Composite of Business/Data, Business/People, Business/Time
    - Identify any constraints on the Technology Architecture about to be designed.
        - ZF: System/Motivation

- Applications Architecture Report, summarizing what was done and the key findings
- Impact Analysis
- Updated business requirements (if appropriate)



Scope of Phase C: Application Architecture

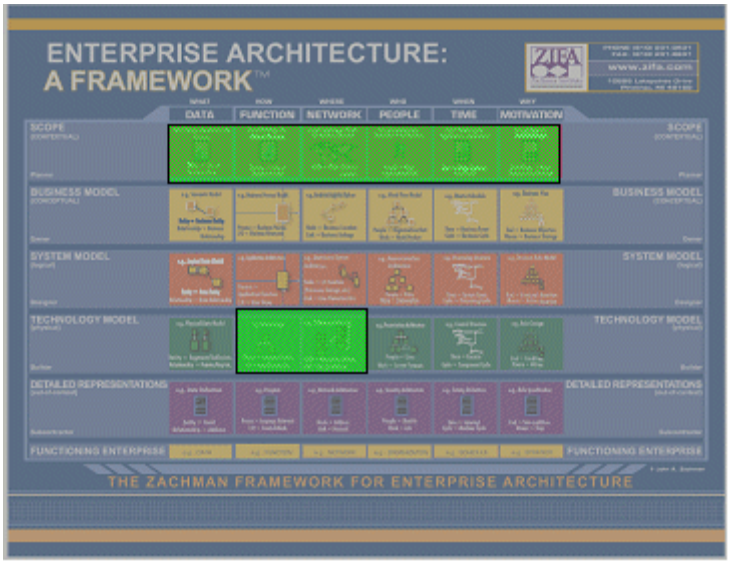
Phase D: Technology Architecture

The outputs of Phase D are given below, first by relevant individual Step, and then as a composite for the whole phase.

Step 1. Create a baseline description in the TOGAF format

The outputs of this step are:

- Technology Architecture principles (if not existing)
  - ZF: Scope/Data, Scope/Function, Scope/Network, Scope/People, Scope/Time, Scope/Motivation
- Technology Architecture Version 0.1:
  - Technology Architecture - Constraints
  - Technology Architecture - Architecture Principles
  - Technology Architecture - Requirements Traceability - key questions list
  - Technology Architecture - Requirements Traceability - criteria for selection of service portfolio
  - Technology Architecture Model - Version 0.1
    - ZF: Technology/Function, Technology/Network

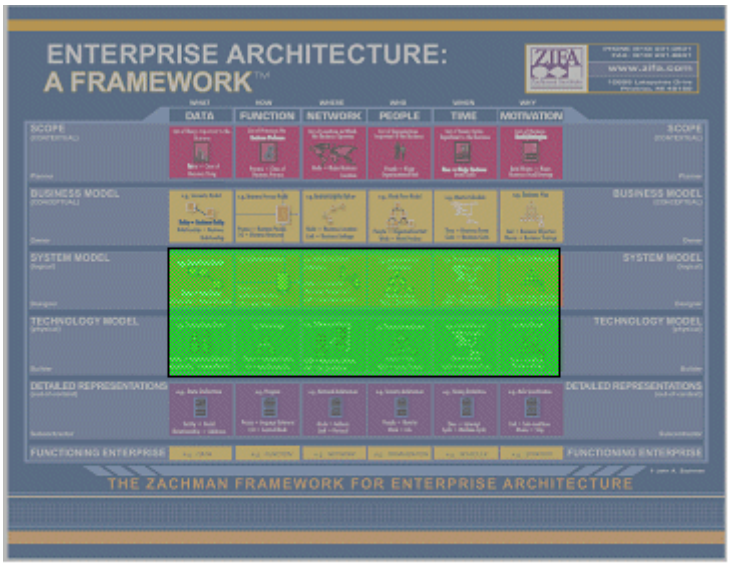


Scope of Phase D: Technology Architecture – Step 1

Step 2. Consider different architecture reference models, viewpoints, and tools

The outputs of this step are:

- Technology Architecture Version 0.2
  - Technology Architecture - Architecture Viewpoints
    - Networked Computing/Hardware View
      - ZF: System/Network, Technology/Network
    - Communications View
      - ZF: Composite of: System/Network, System/People, Technology/Network, Technology/People
    - Processing View
      - ZF: System/Data, System/Function, System/Network, System/People, System/Time, Technology/Data, Technology/Function, Technology/Network, Technology/People, Technology/Time
    - Cost View
      - ZF: Technology / Motivation
    - Standards View
      - ZF: Technology / Motivation
- Technology Architecture - Constraints
  - ZF: System / Motivation

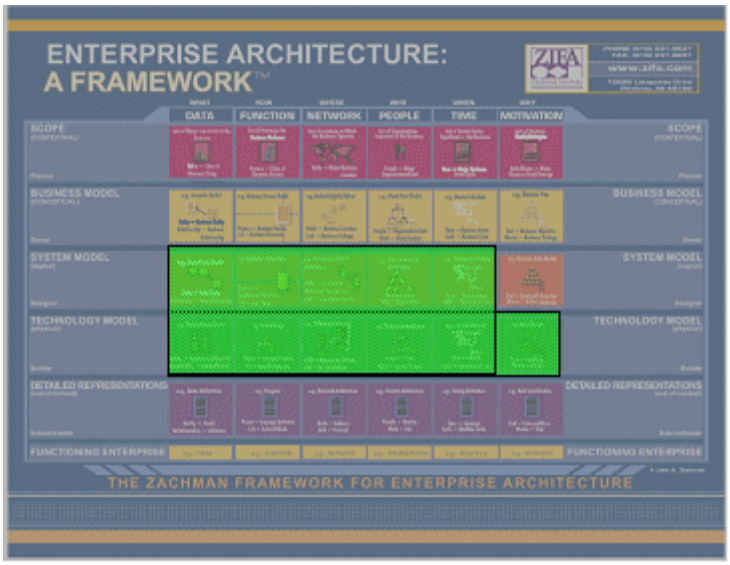


Scope of Phase D: Technology Architecture – Step 2

Step 3. Create an architectural model of building blocks

The outputs of this step are:

- Technology Architecture Version 0.3
  - Technology Architecture Model
    - Networked Computing/ Hardware View
      - ZF: Technology / Network, System / Network
    - Communications View
      - ZF: Composite of: Technology / Network, Technology / People, System / Network, System / People
    - Processing View
      - ZF: Technology / Network, Technology / Time, Technology / People, Technology / Data, Technology/Function, System / Network, System / Time, System / People, System / Data, System/Function
    - Cost View
      - ZF: Technology / Motivation
    - Standards View
      - ZF: Technology / Motivation
  - Technology Architecture - change requests and/or extensions or amendments to be incorporated in an organization-specific architecture continuum.

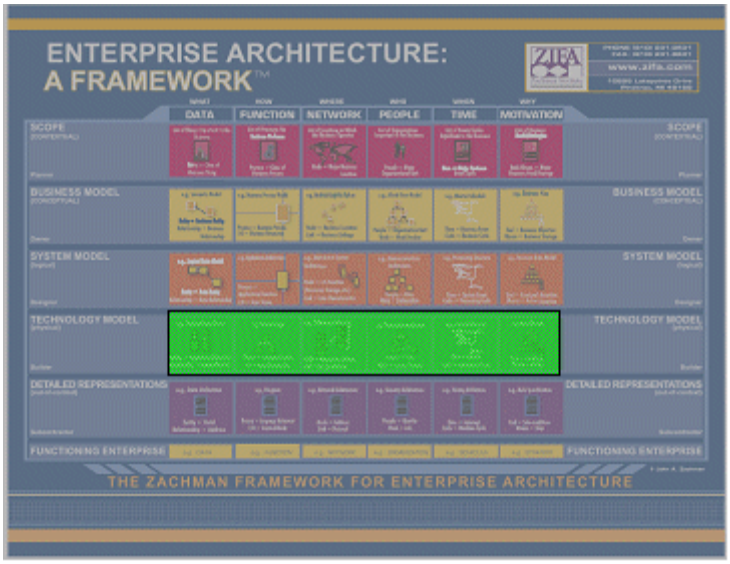


Scope of Phase D: Technology Architecture – Step 3

4. Select the services portfolio required per building block

The outputs of this step are:

- Technology Architecture Version 0.4
  - Technology Architecture - target services (a description of the service portfolios required also known as an Organization Specific Framework)
    - ZF: Technology / Network, Technology / Time, Technology / People, Technology / Data, Technology/Function, Technology/Motivation
  - Technology Architecture - change requests and/or extensions or amendments to be incorporated in an organization-specific architecture continuum

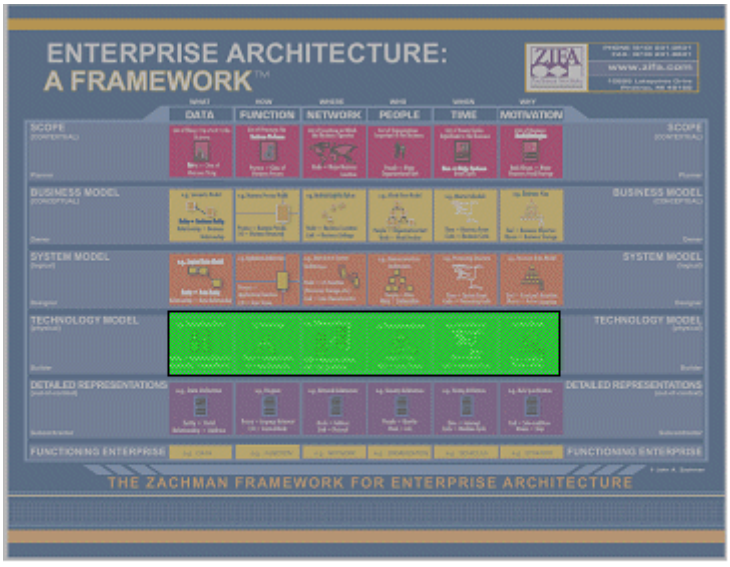


Scope of Phase D: Technology Architecture – Step 4

Step 8. Conduct a gap analysis

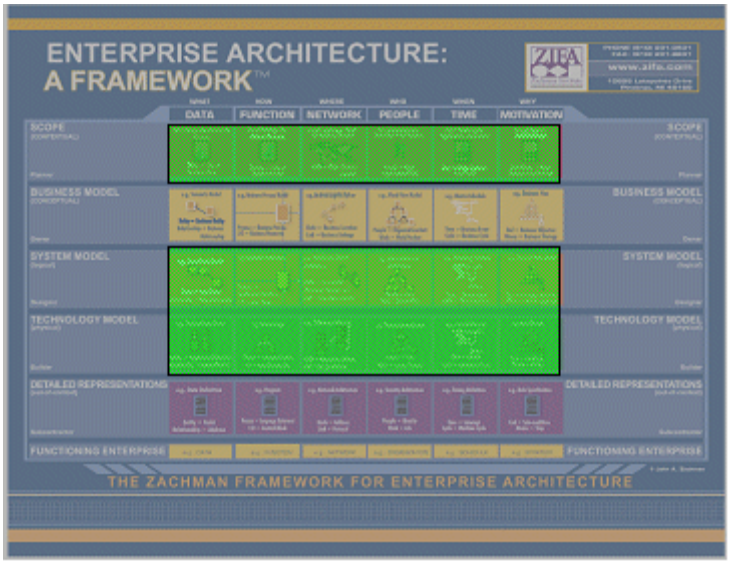
The outputs of this step are:

- Technology Architecture Version 1
  - Technology Architecture - gap report
    - ZF: Composite of Technology/Data, Technology/ Function, Technology/Network, Technology/People, Technology/Time, Technology/Motivation



Scope of Phase D: Technology Architecture – Step 8

Composite Mapping for phase D



Scope of Phase D: Technology Architecture

For more detailed information on the Zachman Framework, refer to any of John Zachman's publications, or the Zachman Institute for Framework Advancement (ZIFA).