Microsoft Architecture Overview

Executive Summary

July 2002

Michael Platt

Microsoft Corporation

Contents

This document is intended for business, software, and infrastructure architects who want to understand Microsoft's approach to enterprise, application, and technology architectures. It covers architectural terminology, patterns, concepts, and definitions as a series of views or levels of architecture.

Enterprise Architecture

The definition of an architecture used in ANSI/IEEE Std 1471-2000is: "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."

An *enterprise architecture* (EA) is a conceptual tool that assists organizations with the understanding of their own structure and the way they work. It provides a map of the enterprise and is a route planner for business and technology change.

Normally an enterprise architecture takes the form of a comprehensive set of cohesive models that describe the structure and the functions of an enterprise. Important uses of it are in systematic IT planning and architecting, and in enhanced decision making.

The individual models in an EA are arranged in a logical manner, and this provides an ever-increasing level of detail about the enterprise, including:

? Its objectives and goals.
? Its processes and organization.
? Its systems and data.

? The technology used.

Microsoft Architectural Perspective

The information in the enterprise architecture can be viewed from many perspectives and it can satisfy many needs. Architectural users include business managers and analysts, system architects and designers, workflow and procedures analysts, logistics specialists, organizational analysts, and so on. These people require high-level summary information, detailed data, and all levels in between. These demands are met through the creation of conceptual views, logical analyses, and physical implementations.

At Microsoft, we find that four general perspectives are important and are commonly used. These are the business, application, information, and technology perspectives.

The business perspective

The *business perspective* describes how a business works. It includes broad business strategies along with plans for moving the organization from its current state to an envisaged future state. It will typically include the following:

? The enterprise's high-level objectives and goals.

? The business processes carried out by the entire enterprise, or a significant portion of the enterprise.

? The business functions performed.

? Major organizational structures.

? The relationships between these elements.

The application perspective

The *application perspective* defines the enterprise's application portfolio and is application-centered. This view will typically include:

? Descriptions of automated services that support the business processes.

? Descriptions of the interaction and interdependencies (interfaces) of the organization's application systems.

? Plans for developing new applications and revising old applications based on the enterprises objectives, goals, and evolving technology platforms.

The application perspective may represent cross-organization services, information, and functionality, linking users of different skills and job functions in order to achieve common business objectives.

The information perspective

The *information perspective* describes what the organization needs to know to run its business processes and operations. It includes:

? Standard data models.

? Data management policies.

? Descriptions of the patterns of information production and consumption in the organization.

The information perspective also describes how data is bound into the work flow, including structured data stores such as databases, and unstructured data stores such as documents, spreadsheets, and presentations that exist throughout the organization.

The technology perspective

The *technology perspective* lays out the hardware and software supporting the organization. It includes, but is not limited to:

? Desktop and server hardware.
? Operating systems.
? Network connectivity components.
? Printers.
? Modems.

The technology perspective provides a logical, vendor-independent description of infrastructure and system components that are necessary to support the application and information perspectives. It defines the set of technology standards and services needed to execute the business mission.

Although there can be many perspectives, there is only one enterprise architecture that the perspectives view. The value of the enterprise architecture is not in any one individual perspective, but in the relationships, interactions, and dependencies among perspectives.

While all the perspectives are key elements of the enterprise architecture, this document will focus on the application and technology perspectives.

Application and Technology Architecture

The *functional* requirements of a software system describe the business value that the software delivers. For a weather service, a functional requirement might be stated as "given a well-formed message A as input, the service will return a message B correct for the time span and geographic location represented in message A."

An *application architecture* is the architecture of any automated services that support and implement such functional requirements, including the interfaces to the business and other applications. It describes the structure of an application and how that structure implements the functional requirements of the organization. Whilst there should ideally be one application architecture in an organization, in practice there are typically many different application architectures.

The *operational* requirements of a software system define the reliability, manageability, performance, security, and interoperability requirements of the software (to list just a few). Common examples might be that the service is only available to authorized subscribers, and that the service be functioning properly 99.999 percent of the time.

A *technology architecture* is the architecture of the hardware and software infrastructure that supports the organization and implements the operational (or non functional) requirements, particularly the application and information architectures of the organization. It describes the structure and inter-relationships of the technologies used, and how those technologies support the operational requirements of the organization.

A good technology architecture can provide security, availability, and reliability, and can support a variety of other operational requirements, but if the application is not designed to take advantage of the characteristics of the technology architecture, it can still perform poorly or be difficult to deploy and operate. Similarly, a well-designed application structure that matches business process requirements precisely—and has been constructed from reusable software components using the latest technology—may map poorly to an actual technology configuration, with servers inappropriately configured to support the application components and network hardware settings unable to support information flow. This shows that there is a relationship between the application architecture and the technology architecture: a good technology architecture is built to support the specific applications vital to the organization; a good application architecture leverages the technology architecture to deliver consistent performance across operational requirements.

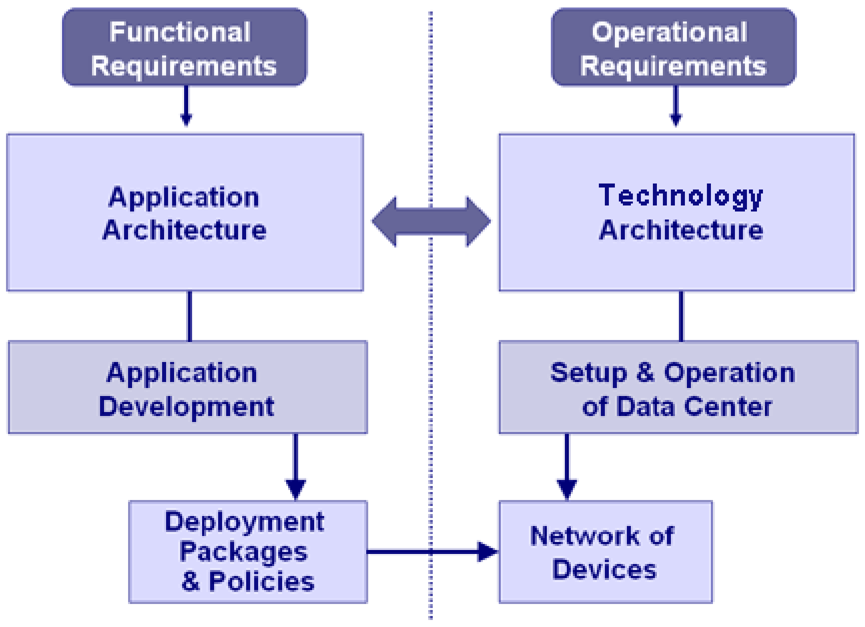

Figure 1. Relationships between architectures

Conceptual, Logical, and Physical Views

For all architectural perspectives there are various views of the architecture that are often classified as conceptual, logical, and physical views. *Conceptual views* are the most abstract and tend to be described in terms that are most familiar to the (non-IT professional) users of the system. The conceptual view is used to define the functional requirements and the business users' view of the application to generate a business model. *Logical views* show the main functional components and their relationships within a system independently of the technical details of how the functionality is implemented. Architects create *application models*, which are logical views of the business model, as they determine how to meet business objectives and requirements. The application models represent the logical view of the architecture for an application. *Physical views* are the least abstract and illustrate the specific implementation components and their relationships. Each of the elements in the physical view is implemented, normally by a design and development process, as a software or hardware system. This *implementation view* is normally owned by the development or operations organizations within an organization and so is outside the scope of this document.

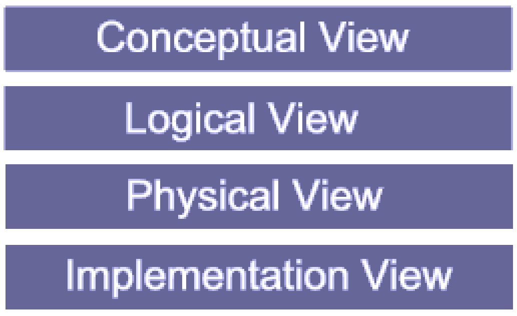

Figure 2. Architectural views

There can be (and indeed, normally are) multiple views at each architectural level; for example, there is normally a logical application architecture view per application.

These views are driven by sets of requirements and in turn generate input into design, development, setup, and operational processes and systems.

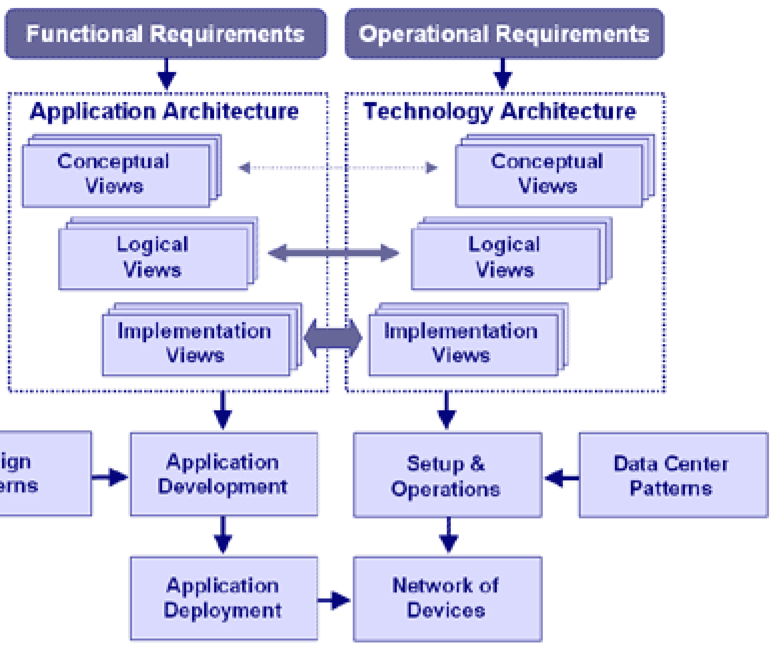

Figure 3. Architectural views and patterns

The remainder of this guide will focus on application and technology architectures, their concepts and key patterns for construction of service-based applications that exploit the emerging technology of Web services. The implementation area, including design, development, setup, deployment, and administration, while of vital importance in the overall system generation, is outside the scope of this document.

Application Architecture

As previously discussed, the application architecture provides three views: the conceptual, the logical, and the physical. These views are used by architects to generate models within organizations that support and meet their business requirements. Ideally there is just one model per view, but in reality there may be multiple models per view—the result of growth and change in organizations and technologies. However, the rationalization of these models to a minimum set is the key to the provision of both efficient and flexible organizations.

Conceptual view

The conceptual view is used to define the business requirements and the business users' view of the application to generate a *business model*. Conceptual modeling techniques, such as use case analysis, activity diagrams, process design, and business entity modeling, help to build a description of the key business processes and the data they use, in way that emphasizes business objectives and requirements, and is free of implementation technology.

Logical view

Architects create *application models* that are logical views of the business model as they determine how to meet business objectives and requirements. The application models represent the logical view of the architecture for an application.

Architects here are concerned with the overall structure of the application. They decide on the mapping of data management and process steps, they design the interactions between parts of the model in terms of logical messages and sequences, and they determine what data and state should be held by the model.

Physical view

Each of the elements in the application model requires mapping to elements of real technologies. In this way application models are realized as *implementation models*. Part of this task is undertaken during conventional development when programmers write detailed business logic as code, but much of the implementation activities are properly classed as framework completion—a technique for development where much of the infrastructure of distributed applications and data management is handled by sophisticated frameworks that are extended by custom application logic and declarative control structures. Framework completion shields developers from the intricacies of, for instance, asynchronous message handling, and allows developers with modest skills to make effective contributions to the project.

Architecting and building these models for an organization at each of the different levels is clearly a considerable amount of work and effort. Additionally, the correct definition of these models is critical for an organization. An incorrect architectural model almost always results in serious design or operational issues such as scalability or reliability problems or, in the worst cases, project non-completion and business impact. Architects are looking for frameworks and roadmaps to assist them in creation and implementation of these models and to minimize the risks associated with the use of incorrect models.

There are two main types of architectural guidance and assistance that can be provided to architects to speed up model generation and minimize risk.

The first of these is a set of architectural concepts that provide:

? A common understanding and communication.

? Guidance as to how and when specific concepts should be used, and information about their attributes.

? An indication as to when these concepts will be realized and available, either in terms of guidance or real technology.

The second is a set of patterns, based on real-world experience harvested from a large number of successful distributed applications, which are composed from these fundamental concepts. These patterns encapsulate important best practices for distributed application design and minimize the risk of project failure by providing known good, tested architectural models.

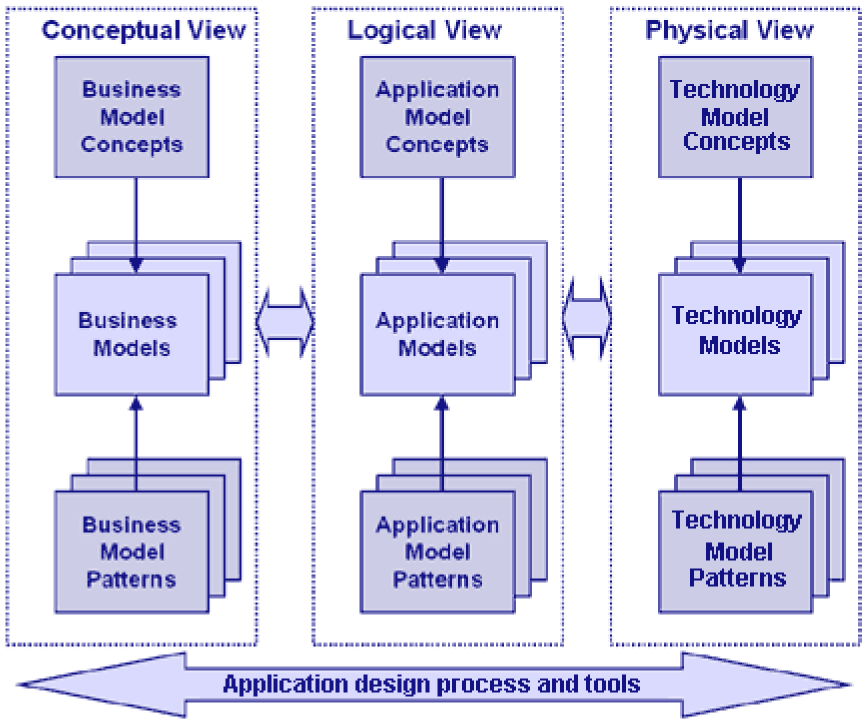
Conceptual View
Business Model Concepts
Business Models
Business Model Patterns
Logical View
Application Model Concepts
Application Models
Application Model Patterns
Physical View
Technology Model Concepts
Technology Models
Technology Model Patterns
Application design process and tools

Figure 4. Views of application architecture

These two sets of guidance; concepts and patterns, are relevant at the conceptual level (where they are business model concepts and patterns), the logical level (where there are application concepts and patterns), and at the technology level. The provision of these concepts and patterns is key to the successful, rapid, and cost-effective implementation of systems and the adoption of technology by organizations.

Application Architecture: Conceptual View

In the past, applications have been built by integrating local system services such as file systems and device drivers. This model was very flexible in providing access to a rich set of development resources and precise control over how the application behaved; however, this was very error-prone, costly, and time-consuming.

Today, complex distributed applications are being constructed that integrate existing applications and services from all over their networks and then add unique value on top by using elements such as business entities, data entities, and façades. This enables developers to focus on delivering unique business value. The result is reduced time-to-market, higher developer productivity, and ultimately, higher-quality software. This has been a powerful architectural model for a number of years; however, it creates "application stovepipes" or "islands of information" that cause significant problems in architectural reuse.

We are entering the next phase of computing—a phase enabled by the Internet together with the concept of Web services, which enables the creation of powerful applications that can be used by anyone, anywhere. It increases the reach of applications and enables the continual delivery of software. In this context, software is a service—to subscribe to and use through a communication network.

.NET facilitates this idea by joining the tightly coupled, highly productive aspects of n-tier computing with the loosely coupled, message-oriented concepts of the Web. This style of computing is called *XML Web services*. It represents the next evolution of application development and is the basis for conceptual application architecture.

Web services are discrete units of application logic that expose message-based interfaces suitable for access across a network. Typically, services provide both the business logic and the state management relevant to the problem they are designed to solve. When designing services, your goal is to effectively encapsulate the logic and data associated with real-world processes, making intelligent choices about what to include and what to implement as separate services.

State manipulation is governed by business rules. Business rules are relatively stable algorithms, such as the method in which an invoice is totaled from an item list, and are typically implemented as application logic.

Services are governed by policy. Policies are less static than business rules and may be regional or customer-specific. Policies are typically driven from lookup tables at run time.

So a more complete definition of services might be, "Services are network-capable units of software that implement logic, manage state, communicate via messages, and are governed by policy."

This conceptual view of applications is covered in more detail in Application Architecture: Conceptual View.

Application Patterns

A pattern is a solution to a problem in a context. A pattern codifies specific knowledge collected from experience in a domain. An application pattern is an architectural-level pattern that defines best practices in architectural design for a specific application environment.

There are a number of different types and taxonomies of patterns that are outside the scope of this document, as is the definition and explanation of specific patterns. Many of the present architectural patterns can be used in Web services-based architectures, but there also are a number of new patterns that are created by the new constructs available in Web services.

Technology Architecture

Like application architecture, technology architecture provides three views: the conceptual, the logical, and the physical. These views are used by architects to generate models within organizations that support and meet their operational requirements. Just as is the case with applications, there should be a single technology architecture, but in reality there almost always are multiple technology architectures caused by the growth and change of organizations and technologies. A key requirement for organizations is the integration of these disparate technology architectures into one all-encompassing architecture to allow the reuse of present-day applications and the rationalization of these technology architectures to a minimum set. The provision of this single, common architecture is vital to the creation of efficient, effective, and flexible organizations.

Conceptual view

A conceptual view of a technology architecture is used to map out areas of technology into a structure and framework. This is used to define, name, and position these areas for a common understanding between the IT supplier and the organizations using the technology and to ensure that all the technology areas required to implement an organization's operational or nonfunctional requirements are defined and available to the organization.

Logical view

The logical view of the technology architecture is where the major functional elements that provide support for enterprise-scale operational requirements and their interrelationships are provided. Enterprise technology elements such as databases, mail systems, transaction support, and reliable messaging are provided in the logical view. The technologies that are provided at this level are normally packaged together as servers by enterprise software vendors.

Physical view

Each of the elements in the technology architecture requires mapping to elements of real technologies for both hardware and software. In this way, technology architectures are realized as complete systems of networks, servers, operating systems, and so on. Actual physical locations, server product names, and connectivity are shown at this level

Architects are looking for technology frameworks and roadmaps from IT vendors to assist them in building systems that meet their organization's operational requirements and to ensure that their organization's technology architectures are aligned with that of the IT vendors.

Technology Architecture: Conceptual View

The technology conceptual architecture is the technology basis for achieving enterprise-scale Web services in an organization. The high-level diagram of the technology conceptual architecture shows a set of generic levels that provide enterprise-based services for Web service generation. These levels contain the common elements that are required by any Web service application or system.

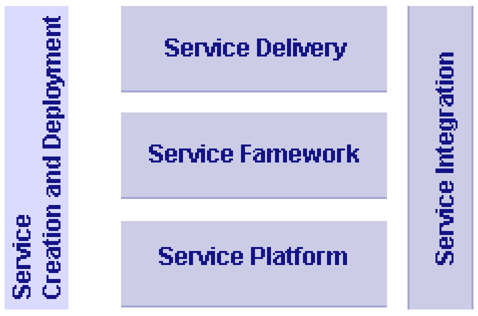

Figure 5. Conceptual view of technology architecture

At the bottom of the diagram is the Service Platform, providing operating system, hardware, storage, networking, and the trust and management services for the whole system.

The Service Framework hosts the process, logic, functions, and state management required by a Web service–based application and is the full Enterprise Application Server with specific support for Web services.

Service Delivery contains the portal and client services that focus on presentation issues and technologies, including support for all types of devices.

Service Integration provides integration and interoperation between services and present-day operational systems: legacy applications, commercial applications, databases, and other Web services. This is commonly called Enterprise Application Integration (EAI).

Finally, Service Creation provides the tools, process, methodologies, and patterns required to design, develop, assemble, manage, deploy, and test Web services.

This conceptual view of technology architecture is covered in more detail in Technology Architecture: Conceptual View.

Technology Patterns

A technology pattern is an architectural-level pattern that defines the best practice architectural design for a specific technology environment. Enterprise architects are looking for guidance and best practice for their organizations in the following critical areas:

Microsoft

? Security, identity, and trust.

? Integration and interoperation, both with future systems and present day operational (legacy) systems.

? Deployment, administration, and management.

? Distributed technology architectures for scalability and performance.

? Technology architecture for reliability and availability.