Chapter 7 – Defining the System Architecture

Table of Contents

Chapter Overview

Chapter 6 introduced the foundation principles of systems design. This chapter begins the detailed discussion of the activities within the fourth core process, *Design System Components*. Specifically, this chapter serves two purposes:

1. Provide a review and summary of technology and architectural concepts most important in modern systems design
2. Describe the two design activities most concerned with technology and architecture: *Describe the environment* and *Design the application components* (see Figure 7-1)

Learning Objectives

After reading this chapter, the student should be able to:

- Explain architectural concepts that influence system design, including ubiquitous computing and software, components, protocols, interoperability, and distributed architectures
- Describe and draw location, network, and deployment diagrams
- Describe a system's environment by drawing architectural diagrams and answering key questions
- Design larger application components based on use cases and other analysis models

Notes on Opening Case and EOC Cases

Opening Case

Technology Decisions at Wysotronics, Inc: This case is a typical situation with many companies today that until recently hosted all of their systems with in-house computers and data centers. With the availability of many different types of hosting services, from colocation to cloud computing, many companies are realizing significant cost savings by using these hosting services. In addition the level of service and availability has increased because hosting companies have created very robust data centers which provide close to 100% availability. In this case, there are two types of systems – production and supply chain systems that have been moved to a virtual private network, and marketing and sales systems that have been placed in a colocation company that also provides managed services.

EOC Cases

Data Integration at Cooper State University: Cooper State University is replacing a paper and pencil student survey system with an automated on-line student survey system. The purpose of the student survey is course and teacher evaluations. The new system will require interfaces into several existing university systems to obtain course, teacher, and student information and login authorizations. The assignment is to draw a network diagram showing the various interconnections. The assignment also requires an assessment of the data requirements to note which interconnected system has ownership of the data.

Community Board of Realtors (running case)**:** Community Board of Realtors is a professional organization that supports real estate offices and agents. In this chapter's case, the assignment is to consider what kind of mobility devices may be appropriate. The assignment also includes a question to consider issues involved to deploy a Web application.

Spring Breaks 'R' Us Travel Services (SBRU) (running case)**:** SBRU is an online travel services that books spring break trips to resorts for college students. The SBRU system has three Web subsystems that use normal Web pages. The fourth subsystem is a social networking subsystem that supports online chatting. The assignment is to develop a network diagram for all four systems, and then to consider whether to host it in-house or on a cloud service. The assignment includes considerations of advantages and disadvantages of in-house versus cloud for various cost comparison alternatives.

On the Spot Courier Services (running case)**:** On the Spot is a small, but growing, courier service that needs to track customers, package pickups, package deliveries, and delivery routes. In this chapter the assignment considers questions of implementing the mobile information system as a smartphone application in contrast with a custom-built system with specific hardware devices.

Sandia Medical Devices (running case)**:** Sandia Medical Devices is a company that specializes in medical monitoring through remote, mobile telecommunication devices. As described in previous chapters, the Real-Time Glucose Monitoring (RTGM) system will include processing components on servers and on mobile devices, such as smartphones, with data exchange via 3G and 4G phone networks. The home office system requires standard technology for the solution. The client side, however, requires more advanced technology to capture information and communicate with the home

office systems. The assignment asks students to consider various options for implementing the client side solution and for communication between medical personnel and the patients. The assignment also includes some questions concerning the value and possible issues related to data mining of the extensive data generated.

Instructor's Notes

Anatomy of a Modern System

Key Terms

- **server** – a computer or group of computers that manages shared resources such as file systems, databases, and Web sites, and enables users and other computers to access those resources over a network
- **Internet backbone network** – a high-capacity and high-speed computer network that carries large amounts of Internet traffic across regions, countries, and continents
- **local area network (LAN)** – a small computer network typically spanning a single home, small office, or one floor of a building
- **World Wide Web (WWW)** – an interconnected set of resources accessed via the Internet
- **Uniform Resource Locator (URL)** – identifier of a Web resource containing a protocol header, server name or address, and resource name
- **hyperlink** – the URL of one Web resource is embedded within another Web resource
- **application software** – software that performs user- or business-specific tasks and is typically constructed as an app or Web-based application
- **app** – application software that is installed on the storage device of a computer or cell phone
- **system software** – software, such as operating systems and Web server software, that works behind the scenes to support application software and control or interact with hardware or software resources
- **Web-based application** – application software that uses a Web browser as the user interface, has a URL for application access, uses a Web server and server-side software components, and uses Web standards for communication between Web browser and server
- **protocol** – a set of languages, rules, and procedures that ensure accurate and efficient data exchange and coordination among hardware and software components
- **virtual private network (VPN)** – secure communication over the Internet using technologies that reliably identify senders and recipients, and encrypt network messages sent among them
- **Hypertext Markup Language (HTML)** – a protocol that defines the structure and content of a Web page

- **Extensible Markup Language (XML)** – an HTML extension that enables the meaning of words, phrases, or numbers to be defined
- **Hypertext Transfer Protocol (HTTP)** – a protocol that defines the format and content of requests for Web documents and related data communication

Lecture Notes

This section examines the technology and architectural features of a modern Web-based system—the Amazon.com shopping application. The discussion serves as a review of modern computer and software technology and an introduction to the architecture of Web-based systems.

Computing Devices

Figure 7-4 illustrates a simplified architecture for the Amazon.com shopping system. **Servers**, which are the repositories for databases and Web sites, provide the core processing for the systems. In the figure servers host the applications for Amazon, for shippers, and for payment processing systems. They are called server because they "serve" or process the requests for information from personal computing devices from the users. These personal computing devices are often called the "client" devices since they receive the services. The communication between the servers and the clients is done through the Internet, which is discussed next.

Networks, the Internet, and the World Wide Web

The true power of modern computing lies not just in the ubiquity and power of individual computers, but in the ability to interconnect them. A computer network is a collection of hardware, software, and transmission media that enables computing devices to communicate with one another and to share resources.

Networks of all sizes interconnect with one another to form the Internet. The largest networks within the Internet are called **Internet backbone networks**. At the other end of the network size scale are **local area networks (LANs)**, which typically span a single home, small office, or one floor of a building. The **World Wide Web**, sometimes called the WWW or simply the Web, is an interconnected set of resources accessed via the Internet. Web resources are identified by a **Uniform Resource Locator (URL)**, which is shown in Figure 7-6. The URL of one Web resource can be embedded within another Web resource as a **hyperlink**.

Software

Software components can be loosely grouped into two types. **Application software** includes software that performs user- or business-specific tasks. **System software** is software that works behind the scenes to support application software and to control or interact with hardware or software resources.

Applications such as the Amazon shopping application are constructed as Web-based applications. Characteristics of **Web-based applications** include:

- Use of a Web browser as the primary user interface on personal computing devices

- User access to the Web application via a URL
- Server-side software components that execute on or are called from a Web server
- Use of Web standards for communication between Web browser and server

A modern laptop computer, tablet computer, or cell phone comes preinstalled with a very complex OS, a Web browser, and a rich set of preinstalled apps. Embedded software components extend the functions of Web browsers and Web servers in ways that enhance Web-based applications and the user experience. It is through the rich set of embedded software that the client devices are able to provide the wide range of services available on smartphones, tablets, and laptops.

Protocols

A **protocol** is a set of languages, rules, and procedures that ensure accurate and efficient data exchange and coordination among hardware and software components. Modern information systems rely on hundreds or thousands of protocols.

Network protocols enable accurate message transmission among the various computers and software components. They function as the "plumbing" that enables messages to find their way from sender to recipient, enable multiple devices to efficiently share wireless and wired connections, and handle tasks such as identifying and retransmitting lost messages. For increased security, the system designer may want to work with security and network specialists to develop a **virtual private network (VPN)** to isolate sensitive communications between servers or between an organization's own employees and servers. However, for public networks, security is usually provided through https, as explained next.

The World Wide Web is built on a small family of protocols for encoding Web documents and hyperlinks, requesting documents from Web servers, and responding to those requests. The most important protocols include the following: Hypertext Markup Language (HTLM), Extensible Markup Language (XML), Hypertext Transfer Protocol (HTTP), and Hypertext Transfer Protocol Secure (HTTPS).

Quick Quiz

Q: Modern Internet based computing systems contain two categories of computing devices. What are they called?

A: Servers and Client computers

Q: What does URL stand for and what is it used for?

A: URL stands for Uniform Resource Locator. It is like an address for the Internet to find or locate specific resources, such as a particular Web page.

Q: What is a hyperlink?

A: A hyperlink is an URL that is embedded within a Web page. Clicking on a hyperlink invokes the process to retrieve the resource specified by the embedded URL.

Q: What is a Web based application?

A: It is an application that resides on a server and utilizes a Web browser for the client display,

and is accessed via the Internet using a URL.

Q: What is a protocol good for?

A: Through the use of protocols programs and systems are able to communicate with each other. A protocol is simply a standard language or procedure that can be used for communication. There are many levels of protocols.

Architectural Concepts

Key Terms

- **software as a service (SaaS)** – a software delivery model similar to a utility, in which application software is accessed via the Internet without locally installed programs
- **Web service** – software function or related set of functions that can be executed via Web standards
- **client/server architecture** – a software design and deployment method that divides software into components that manage resources and components that use those resources
- **three-layer architecture** – a client/server architecture that divides an application into view layer, business logic layer, and data layer
- **view layer** – the part of a three-layer architecture that contains the user interface
- **business logic layer** – also known as the domain layer, the part of a three-layer architecture that contains the programs that implement the business rules and processes
- **data layer** – the part of a three-layer architecture that interacts with the data

Lecture Notes

Software as a Service

If an organization requires some services it could build or buy a software system to carry out that service. Alternatively, it could find a firm that provides that service and only buy the service itself, much like a utility. This is **Software as a Service**. Two common features shared by most applications that employ the SaaS model include the following:

- Little or no application software is installed on the user’s device.
- User data is stored on servers, though copies may be stored on the user’s device for improved performance.

Web Services

A Web service is a software service accessed over the Internet using Web protocols. It commonly refers to small, focused applications, such as transmitting shipping information from seller to shipper, or

single functions, such as looking up the zip code that matches a shipping address. In essence, a Web service is a software function, subroutine, method, or program that meets the following criteria:

- Is called from one application via a URL or other Web protocol
- Accepts input data embedded within the URL or via another protocol
- Executes on the Web service owner's servers
- Returns processing results encoded within a Web page or document

The implications of Web services for system design are significant. At the strategic level, information system developers must do the following:

- Scan the range of available Web services and decide which to incorporate into their software. To the extent they include other Web services, they expand the functions of their own software with minimal related development cost.
- Decide which (if any) functions of their own software should be implemented as Web services and made available to other systems.
- How will data passed to/from Web services be secured?
- Even if no Web services are made available to external users, should some portions of the system be structured as Web services to facilitate access and use by other internal systems?
- What performance and availability guarantees will be provided to Web service users?

Distributed Architectures

Client/server architecture is a method of organizing software to provide and access distributed information and computing resources. It divides software into two classes: client and server. A server manages system resources and provides access to these resources through a well-defined communication interface. A client uses the communication interface to request resources, and the server responds to these requests.

Three-layer architecture is a variant of client/server architecture that is used for all types of systems, from internally deployed desktop applications to globally distributed Web-based applications. Three-layer architecture divides the application software into three layers that interact, as shown in Figure 7-11 including the **view layer**, the **business logic layer** and the **data layer**.

A major benefit of three-layer architecture is its inherent flexibility. Multiple layers can execute on the same computer, or each layer can operate on a separate computer. Complex layers can be split across two or more computers. System capacity can be increased by splitting layer functions across computers or by load sharing across redundant computers.

Quick Quiz

Q: What does SAAS stand for? What are the unique characteristics of SAAS?

> A: Software as a Service. The application software does not reside on the organizations computers. The data can reside on either the organization's computers, or on the service

providers computers.

Q: What is the difference between SAAS and a Web Service?

A: SAAS is usually a full featured application, while a Web service is usually a single service or simple service.

Q: What is 3-layer architecture? What are the three layers?

A: Three layer architecture is a client/server architecture that divides the application into view layer, business logic layer, and data access layer.

Interoperability

Key Terms

- **interoperability** – the ability of a component or system to interact with other components or systems

Lecture Notes

Interoperability is the ability (or lack thereof) of a component or system to interact with other components or systems. It forms the foundation for almost all new software development. The operating environment of almost all organizations is extremely complex with many different elements that must all work together. To ensure interoperability, system designers must do several things, including the following:

- Understand and describe the current environment in which the system will operate.
- Purchase existing software components and services that can provide needed functions for the new system.
- Build components that can't be purchased as software modules or used as a service.
- Structure and assemble the components so that it is all interoperable over the long term.

Quick Quiz

Q: Why is interoperability such an important concept?

A: In today's computing world, no system operates in a vacuum or in isolation. Information systems today must interact and interface with many other systems. Information systems today are created from both purchased components and in-house constructed components.

Architectural Diagrams

Key Terms

- **network diagram** – a model that shows how locations and hardware components are interconnected with network devices and wiring

Lecture Notes

Location Diagrams

Location diagrams are commonly used to show the geographic placement of various system components, including hardware, buildings, and users.

Network Diagrams

A **network diagram** shows how locations and hardware components are interconnected with network devices and wiring. There are many different kinds of network diagrams—each emphasizing different aspects of the network, connected hardware resources, and users.

Deployment Diagrams

A deployment diagram describes how software components are distributed across hardware and system software components.

Quick Quiz

Q: What is the difference between a location diagram and a network diagram?

> A: A location diagram just identifies the locations and the functions at each location. A network diagram does show locations, but it also shows the network connections that interconnect those locations. A network diagram can also apply to a single location.

Q: What is the difference between a network diagram and a deployment diagram?

> A: A network diagram focuses on the interconnections between and within a location. A deployment diagram focuses on the software components at each location.

Describing the Environment

Key Terms

None

Lecture Notes

In Figure 7-1 *Describing the Environment* is the first design activity of the System Design core process. As discussed in Chapter 6, that description includes two key elements: External systems and Technology architecture

To describe these key elements the systems developers must answer many questions. The following questions help to discover the important aspects of the environment.

Key Questions

The following questions are a typical starting point:

1. What are the key features of the existing or proposed technology environment that will support or constrain the system?

    a. What operating systems will be used?

    b. What other system software (e.g., Web server, database management, and intrusion detection software) will be used?

    c. In what ways are network messages filtered or otherwise secured? Are any changes required to support interactions with external systems or user-interface devices?

    d. What APIs and development tools are compatible with the existing technology environment?

2. With what external systems and databases will the system under development interact? For each system or database, answer the following questions:

    a. What is the timing and frequency of each interaction?

    b. What is the data content of inputs to and outputs from the system?

    c. What protocols will format and encode data flowing to or from the external system?

    d. What are the security requirements of each inflow and outflow?

    e. What security methods and protocols will be used to satisfy the security requirements?

3. What devices will be used for automated inputs and outputs?

    a. What protocols will format and encode data flowing to or from the devices?

    b. What are the security requirements of each inflow and outflow?

    c. What security methods and protocols will be used to satisfy the security requirements?

    d. What APIs and development tools are compatible with the existing technology environment and required automated inputs and outputs?

4. What user-interface technology will be used?

    a. Where will users be located?

    b. What hardware device(s) will users use?

    c. What operating systems will run on "smart" user-interface devices?

    d. On what other user device software will the system rely (e.g., browsers, plug-ins, and software utilities embedded in the device)?

    e. What protocols will format and encode data flowing to or from user devices?

    f. What are the security requirements of each inflow and outflow?

    g. What security methods and protocols will be used to satisfy the security requirements?

    h. What APIs and development tools are compatible with the existing technology environment and required user interfaces?

RMO Environment Description

Figure 7-16 shows a network diagram describing RMO's current technology architecture. This section focuses on a handful of updates to demonstrate how the environment is affected.

Mobile Devices and Apps: The current technology architecture connects users to the system using desktop or laptop computers. Expanding the range of user devices to include tablets and smartphones will require updates to the supporting system software, APIs, and development tools.

Web Technologies and Adapted Content: Because some users won't install an app, the CSMS must also support a browser-based user interface with support for multiple screen sizes, Web browsers, and plug-ins. That will require more complex user-interface coding than exists in the current system, which simply serves static Web pages and forms. The updated user-interface coding will need to query the user's device and browser and adjust the content of the Web pages transmitted to match the device characteristics.

Social Networking: Social networking services provide the ability to interface with external systems in two ways: A Web services interface and an API and toolkit that enable developers to create customized functions and embed them within the social networking site and interface.

Security Implications: Supporting apps, multiple browsers with plug-ins, and interfaces to social networking sites will probably require security updates to the current technology architecture. Apps and some browser plug-ins will need to be digitally signed prior to distribution via the app stores for each device's operating system.

External Hosting: External hosting of all or part of the CSMS is an option that can reduce risk, improve performance, and possibly reduce cost. Hosting the application with a national or global company, such as Google or Amazon, would enable RMO to take advantage of an existing and highly distributed computing infrastructure. Key components could be replicated at multiple locations to improve performance and to provide fault tolerance.

Designing Application Components

Key Terms

system of record – a system or application component that maintains the current and correct master copy of one or more data items

Lecture Notes

Chapter 6 defined an application component as a well-defined unit of software that performs one or more specific tasks. That definition masked some important details, including variations in component size ranging from single subroutines or methods to entire subsystems; variations in programming language, protocols, and supporting system software; and the ability to build, buy, or freely access components as Web services of entire SaaS systems. This section concentrates on defining the functions and boundaries of larger application components.

Application Component Boundaries

A key question to be answered when designing application components is which components will perform which functions? To answer this question, the designer looks for similarities among system functions to guide factoring or grouping. But how does a designer determine or measure similarity among system functions? During analysis use cases are defined. There are three factors from associated use cases that can help group functions together.

- Actors. Each use case identifies one or more specific actors. Software for use cases that interact with the same actors is a candidate for grouping into a single application component.
- Shared data. Use cases that interact with the same domain class(es) are candidates for grouping into a single application component.
- Events. Use cases that are triggered by the same external, temporal, or state event are candidates for grouping into a single application component.

RMO CSMS Application Architecture

This section is an extended example of grouping system functions together on the basis of use cases and actors. Figure 7-18 illustrates a matrix of use cases, data (domain classes), and events that is used to group use cases together into components. Figure7-19 illustrates how these use cases can be divided into packages for a three-layer architecture for the various view layer components, the business logic layer, and the data layer. There are multiple view layer components due to the wide variety of users and devices.

Application Component Integration

Modern application developers are often faced with the task of integrating legacy systems, purchased application components, third-party SaaS applications, and custom-developed components. Integration can occur at two levels, (1) API or program interface level or (2) data level, which can either consist of

sending transactions between applications, or sharing access through a database. Figure 7-20 illustrates the existing RMO systems and the flow of information between the five subsystems. When data is maintained via a database, often local copies of the data may be made to increase efficiency. Local copies need to be synchronized periodically. Updates can be made on any copy. A **system of record** is used to note the master copy and how synchronization must proceed.

Quick Quiz

Q: What are three criteria that can be used to group use cases together to define component boundaries.

A: Actors, Shared data, or Events.

Q: What does a system of record represent?

A: It is the "boss" system. Or the system that maintains the original copy of shared data.

Classroom Activities

The material in this chapter covers a very broad topic. An entire semester can be spent on understanding these topics. This chapter simply skims across the top of the issues related to the software architectural environment.

A helpful classroom activity is to take Figure 7-4 and ask the students to apply the concepts the figure. In other words, where does each concept appear in the figure and how would it be implemented. The discussion could include:

- Computing Devices
- Networks – LAN and Internet
- Software
- Web applications and local software (including embedded software)
- Protocols (refer to Figure 7-8)
- SAAS
- Web Services
- Client/Server architecture

A classroom activity for Describing the Environment that is always interesting is to invite a guest speaker, perhaps a senior member of the university technical staff to come and describe the university operating environment. With a little preparation, the speaker could address both Describing the Environment and Designing the Application Components.

Troubleshooting Tips

Probably the major problem in this chapter is that there are so many terms and many topics. Help the students focus on the key issues. Make sure they understand the basics of what the operating environment is and what application components are.

Discussion Questions

1. The Importance of Describing the Environment correctly

The purpose of this discussion is to help the students realize that value and quality of all other design activities will depend on understanding the details of the environment into which the system will need to be deployed. Both design and implementation activities can be done incorrectly if the environment is misunderstood. For example, even the database SQL statements will vary depending on the DBMS to be used. Discussion questions could include:

Who in the organization is the best person(s) to talk with to get correct answers?

How do you know if you have describe the environment accurately? In enough detail?

2. Issues for Designing the Application Components

Application components can be as large as subsystems and as small as a single function module. Application components can come from existing components within the organization, purchased from outside suppliers, downloaded and installed as open-source software, or custom built in-house. How do developers decide which components should be purchased, obtained as open-source, or built in-house? How does the development team limit the alternatives? How do they decide on what the components could be and what alternatives are available for each component? Ask the students how they would approach these issues if they were the project manager, or the project team.