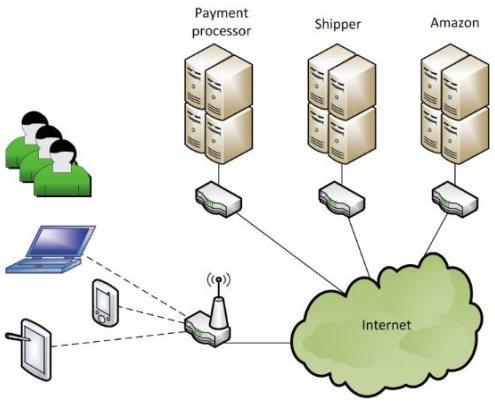
Lesson 9 Defining the System Architecture

Anatomy of a Modern System

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

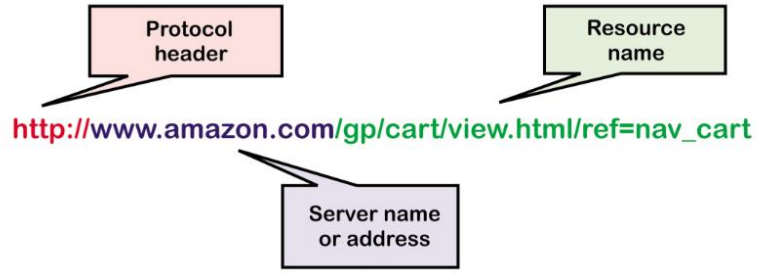
Figure 1 below illustrates a simplified architecture for the Amazon.com shopping system. **Servers**, which are the repositories for databases and Web sites, provide the core processing for the systems. In the figure servers host the applications for Amazon, for shippers, and for payment processing systems. 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 2. 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).

Architectural Concepts

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

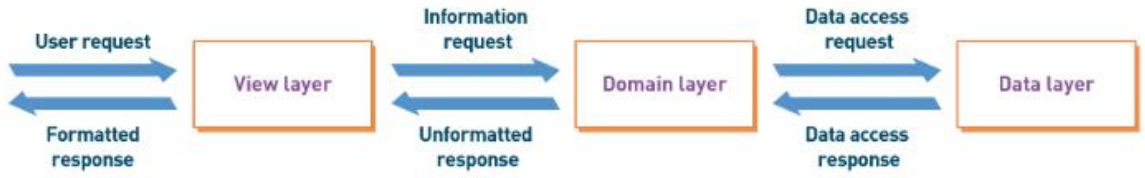
The implications of Web services for system design are significant. At the strategic level, information system developers must do the following:

- 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 below 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.

Interoperability

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.

Architectural Diagrams

Location Diagrams

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

Network Diagrams

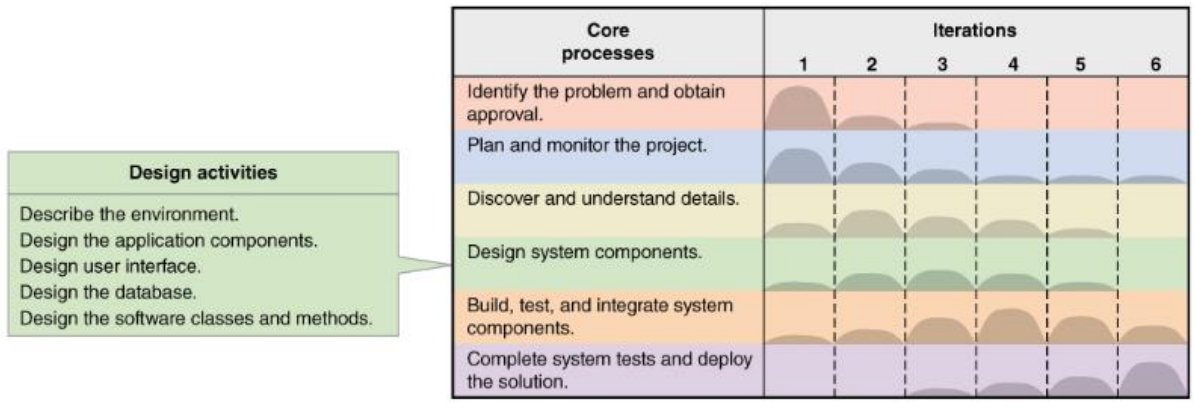
A **network diagram** shows how locations and hardware components are interconnected with network devices and wiring. There are many different kinds of network diagrams—each emphasizing different aspects of the network, connected hardware resources, and users.

Deployment Diagrams

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

Describing the Environment

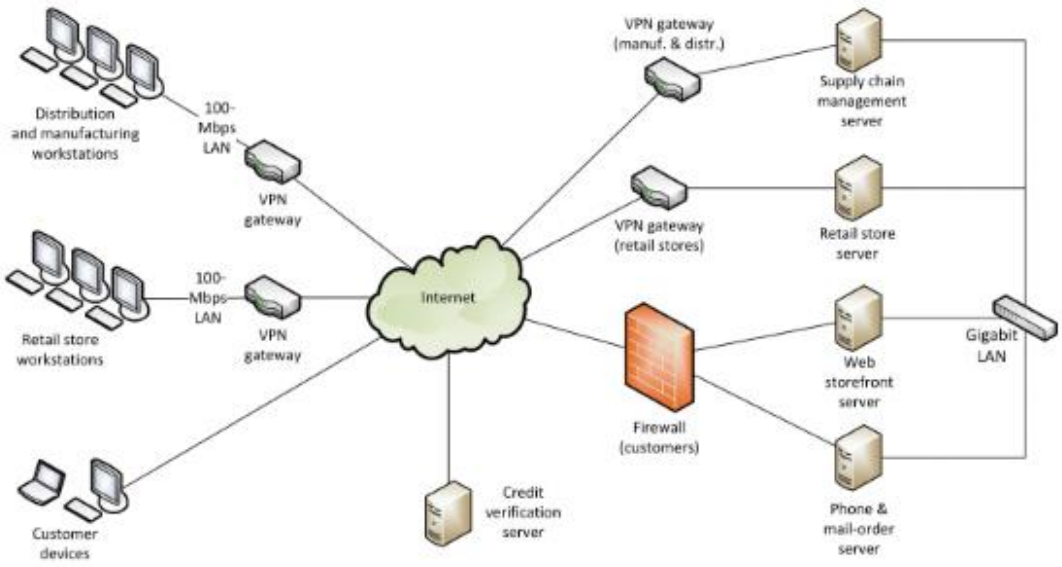
In figure below *Describing the Environment* is the first design activity of the System Design core process. As discussed in Chapter 6, that description includes two key elements: External systems and Technology architecture



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

RMO Environment Description

Figure below 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

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.

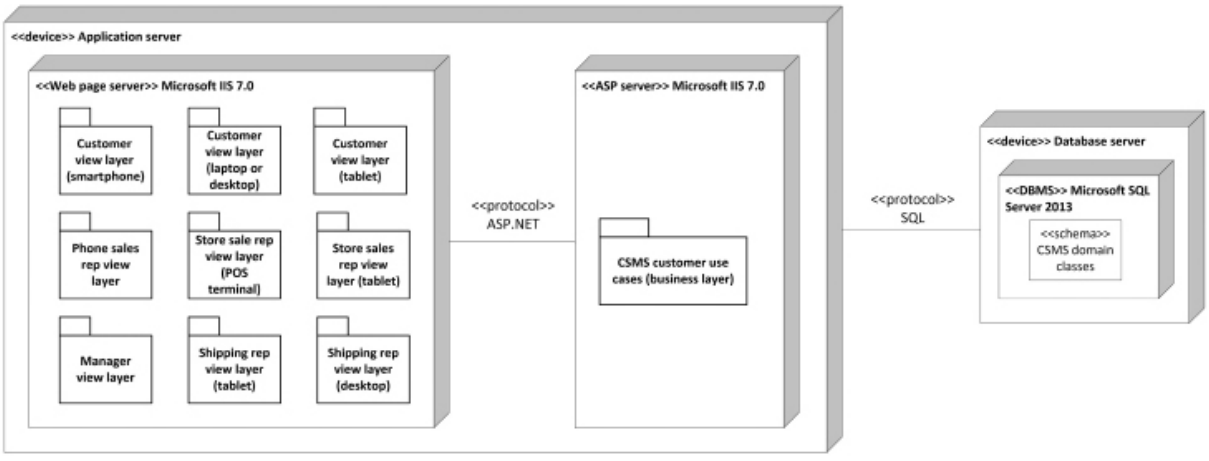
RMO CSMS Application Architecture

This section is an extended example of grouping system functions together on the basis of use cases and actors. Figure 1 illustrates a matrix of use cases, data (domain classes), and events that is used to group use cases together into components.

Use case	User/actor	Domain class(es)	Event(s)	Group
Create phone sale	Phone sales representative	ProductItem, InventoryItem, SaleItem, Sale, SaleTrans	Customer request while shopping by phone	A
Create store sale	Store sales representative	ProductItem, InventoryItem, SaleItem, Sale, SaleTrans	Customer request while shopping in store	B
Create/update customer account	Customer, phone or store sales representative	Customer, Account, Address	Customer request or sale to a new customer	C
Look up order status	Shipping, customer, management, phone or store sales representative	ProductItem, InventoryItem, SaleItem, Sale, SaleTrans, Shipment, ReturnItem	Customer, representative, shipping, or management request	
Track shipment	Shipping, customer, management, phone or store sales representative	Shipment, Shipper, SaleItem	Customer, representative, shipping, or management request	
Create item return	Customer, phone or store sales representative	SaleItem, ReturnItem	Customer requests return	
Search for item	Customer, phone or store sales representative	ProductItem	Customer request while shopping online, by phone, or in store	
View product comments and ratings	Customer, phone or store sales representative	ProductItem, ProductComment	Customer request while shopping online, by phone, or in store	
View accessory combinations	Customer, phone or store sales representative	ProductItem, AcessoryPackage	Customer request while shopping online, by phone, or in store	

Fill shopping cart	Customer	ProductItem, InventoryItem, CartItem, OnlineCart	Customer request, usually after sale completed	
Empty shopping cart	Customer	ProductItem, InventoryItem, CartItem, OnlineCart	Customer request while shopping online	
Check out shopping cart	Customer	ProductItem, InventoryItem, CartItem, OnlineCart, SaleItem, Sale, SaleTrans	Customer request while shopping online	
Fill reserve cart	Customer	ProductItem, InventoryItem, CartItem, OnlineCart	Customer request while shopping online	D
Empty reserve cart	Customer	ProductItem, InventoryItem, CartItem, OnlineCart	Customer request while shopping online	
Convert reserve cart	Customer	ProductItem, InventoryItem, CartItem, OnlineCart	Customer request while shopping online	
Rate and comment on product	Customer	Customer, ProductComment, ProductItem	Customer request, usually after sale completed	
Provide suggestion	Customer	Customer, Suggestion	Customer request while shopping online	
Send message	Customer	Customer, Message	Customer request while shopping online	
Browse messages	Customer	Customer, Message	Customer request while shopping online	
Request friend linkup	Customer	Customer, FriendLink	Customer request while shopping online	E
Reply to linkup request	Customer	Customer, FriendLink	Customer request while shopping online	
Send/receive partner credits	Customer	Customer, CustPartnerCredit, PromoPartner	Customer request while shopping online	
View "mountain bucks"	Customer	Customer, Sale	Customer request while shopping online	
Transfer "mountain bucks"	Customer	Customer, Sale	Customer request while shopping online	

Figure 2 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 3 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.