CHAPTER 9 System Architecture

LEARNING OUTCOMES

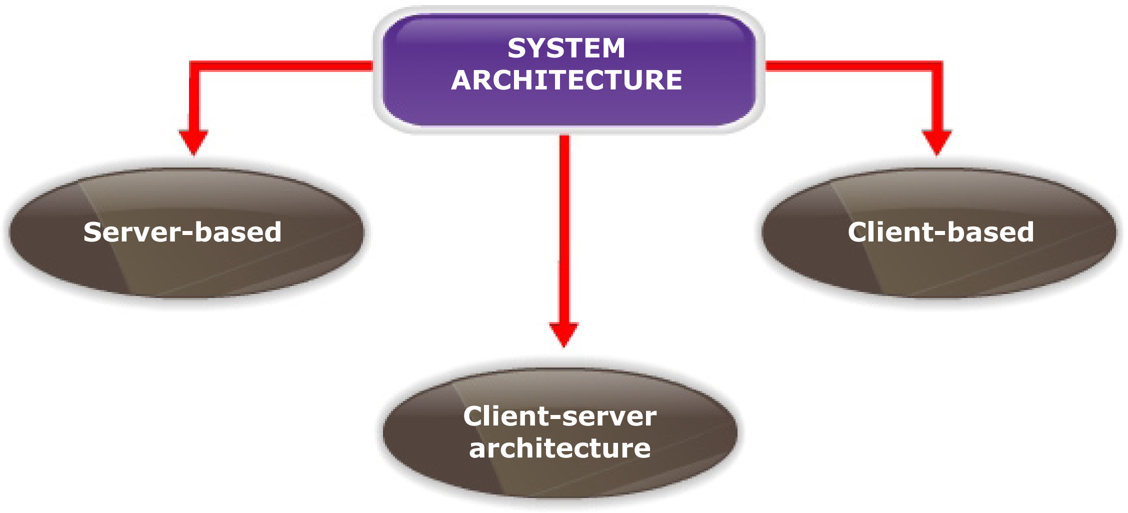
By the end of this chapter, you should be able to:

1. Identify server-based, client-based and client server architectures;
2. Be familiar with how to choose the right architecture;
3. Describe the impact of client-server tiers;
4. Explain hardware and software specification; and
5. Explain the process involved in finalising design specification.

INTRODUCTION

In this chapter, your objective is to determine an overall architecture to design an information system. An information system requires hardware, software, data and people to accomplish a specific set of functions. System architecture translates the logical design of an information system into a physical structure which includes system's hardware, software and network environment.

This chapter discusses the architecture components and planning to understand how the software can be divided into different parts and distributed over the architecture. There are three principals system architecture planning to used today:



Hardware and software specification are used to determine factor for selection of hardware and software in supporting the architecture plan. The deliverable from system design phase is the system design specification. If this document is approved, the next phase will be implemented.

9.1 ARCHITECTURE COMPONENTS

Every information system can be divided into four basic functions as shown in Table 9.1:

Table 9.1: Basic Functions of Information System

Function	Description
Data storage	Most application programs require data to be stored and retrieved, whether the information is a small file such as a memo produced by a word processor, or a large database that stores an organisation's accounting records.
Data access logic	This is a process that is required to access data, which often means database request using Structured Query Language (SQL) commands.
Application logic	It is an application program to handle processing logic. The process can be simple or complex depending on the application. The logic used is documented in Data Flow Diagram (DFD).
Presentation logic	An interface that allows users to interact with the system. The presentation of information to the user, and the acceptance of the user's commands through user interface.

These four functions (data storage, data access logic, application logic, and presentation logic) are the basic building blocks of any application. It is used in the three primary hardware components of a system as shown in Table 9.2;

Table 9.1: Basic Functions of Information System

Hardware Component	Description
Client	Computers are the input/output devices employed by the user, and are usually desktop or laptop computers, but can also be handheld devices, cell phones, special-purpose terminals, and so on.
Servers	Typically larger computers that are used to store software and hardware that can be accessed by anyone who has permission. Servers can come in several flavors: mainframes (very large, powerful computers), minicomputers (large computers), and microcomputers (small desktop computers). Server can support large number of client computers at various locations.
Network	The network that connects the computers can vary in speed from a slow cell phone or modem connection that must be dialed, to medium speed always-on frame relay networks, to fast always-on broadband connections such as cable modem, DSL, or T1 circuits, to high speed always-on Ethernet, T3, or ATM circuits.

Depending on the architecture, the four functions can be performed on a server, on a client or divided between the server and the client. As you plan the system design, you must determine where the functions will be carried out and the advantages and disadvantages of each design approach.

ACTIVITY

What are the four basic functions of any information system?

9.2 PLANNING THE ARCHITECTURE

This section discusses several server and client architecture and how each design alternative handles system on four basic functions: data storage, data access logic, application logic and presentation logic.

9.2.1 Server-Based Architectures

The very first computing architectures were server-based architectures, with the server (usually a central mainframe computer) performing all four application functions. The clients (usually terminals) enabled users to send and receive messages to and from the server computer. Servers process everything, while clients receive instructions from servers and users. Figure 9.1 shows the server-based architecture.

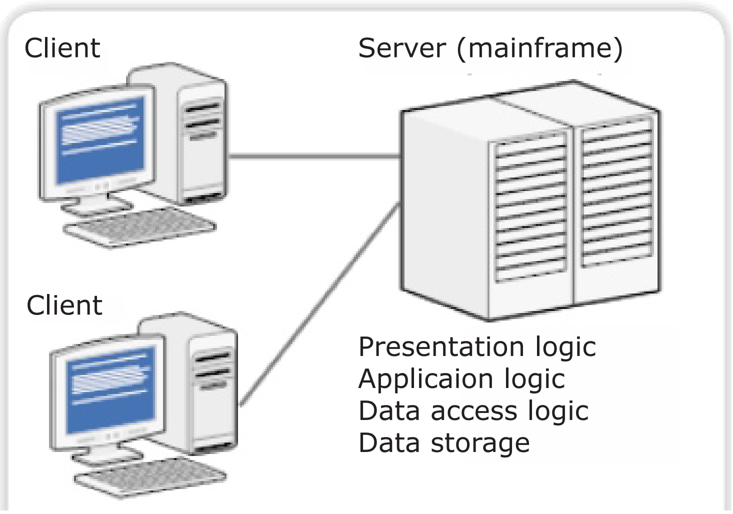


Figure 9.1: Server-based architecture

Here, the application software is developed and stored in one computer, and all the data are also found on the same computer. If there are many servers, there is one point of control, where all messages move via the one central server.

The fundamental problem with server-based architectures is that the server must process all messages. As the demands for more and more applications grow, many server computers become overloaded and unable to quickly process all the users' demands. Response time becomes slower, and network managers are required to spend increasingly more money to upgrade the server computer. Unfortunately, upgrades come in large increments and are expensive.

9.2.2 Client-Based Architectures

With **client-based architectures**, the clients are personal computers on a local area network, and the server computer is a server on the same network.

The application software on the client computers is responsible for presentation logic, application logic and data access logic while the server simply stores the data. Figure 9.2 shows the client-based architectures

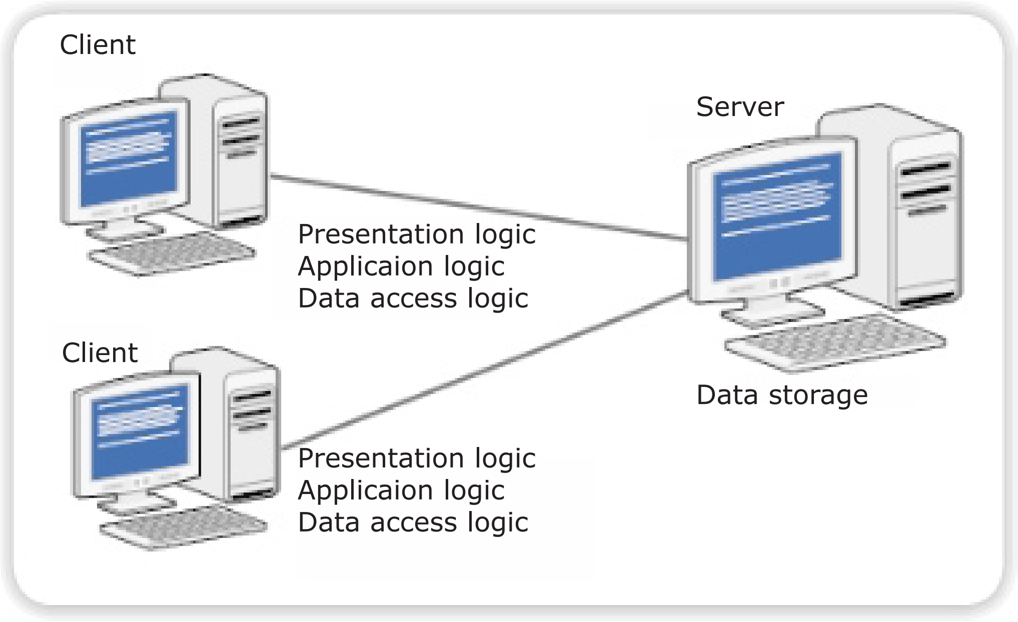
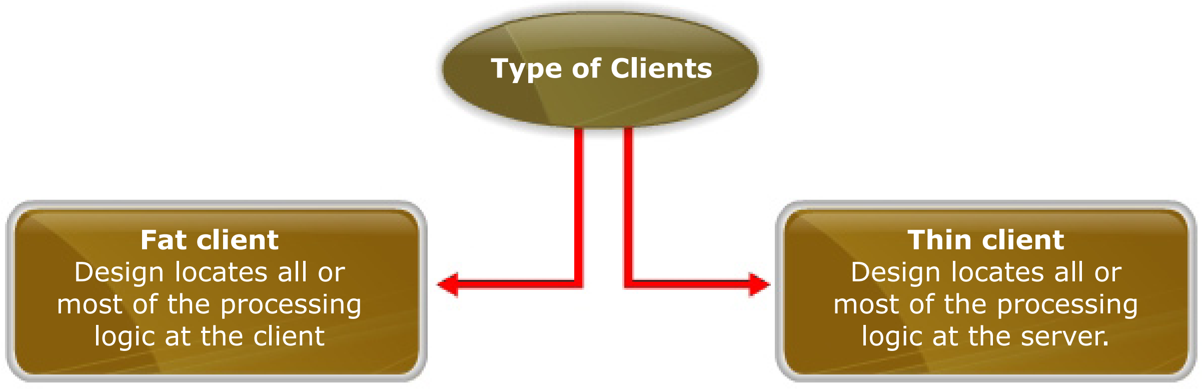


Figure 9.2 Client-based architectures

This architecture often works well. However, as the demands for more and more network applications grow, the network circuits can become overloaded. The fundamental problem in client-based architectures is that all data on the server must travel to the client for processing.

For example, suppose the user wishes to display a list of all students in Asia e University. All the data in the database must travel from the server where the database is stored over the network to the client, which then examines each record to see if it matches the data requested by the user. This can overload both the network and the power of the client computers.


Type of Clients
Fat client
Design locates all or most of the processing logic at the client
Thin client
Design locates all or most of the processing logic at the server.

Most IT experts choose thin client designs for better performance, because most processing logic resides on the server near the data. In contrast, a fat client handles more of the processing logic must access the data from the server more often. Table 9.3 compares the characteristics of fat and thin clients.

NOTES

What are the four basic functions of any information system?

Table 9.3: Fat and Thin Clients Characteristic

Characteristic	Fat client	Thin client
Network traffic	Computers are the input/output devices employed by the user, and are usually desktop or laptop computers, but can also be handheld devices, cell phones, special-purpose terminals, and so on.	Lower, because most interaction between code and data takes place at the server.
Performance	Slower, because more network traffic is required.	Faster, because less network traffic is required.
Initial cost	Higher, because more powerful hardware is required.	Lower, because workstation hardware requirements are not as stringent.
Maintenance cost	Higher, because more program code resides on client.	Lower, because most program codes reside on the central server.
Development ease	Easier, because systems resembles traditional designs where all processing was performed at the client.	More difficult, because developers must optimise the division of processing logic.

9.2.3 Client–Server Architectures

Most organisations today choose client-server architectures, which attempt to balance the processing between the client and the server by having both do some of the application functions. In these architectures, the client (usually microcomputer) is responsible for the presentation logic while the server (usually mainframe, minicomputer or microcomputer) is responsible for the data access logic and data storage. The application logic may reside on either the client, the server, or be split between both (see Figure 9.3).

The client shown in Figure 9.3 can be referred to as a thick or fat client if it contains the bulk of application logic. A current practice is to create client-server architectures using thin clients because there is less overhead and maintenance in supporting thin-client applications.

For example, many Web-based systems are designed with the Web browser performing presentation, with only minimal application logic using programming languages like JavaScript, and the Web server having the application logic, data access logic, and data storage.

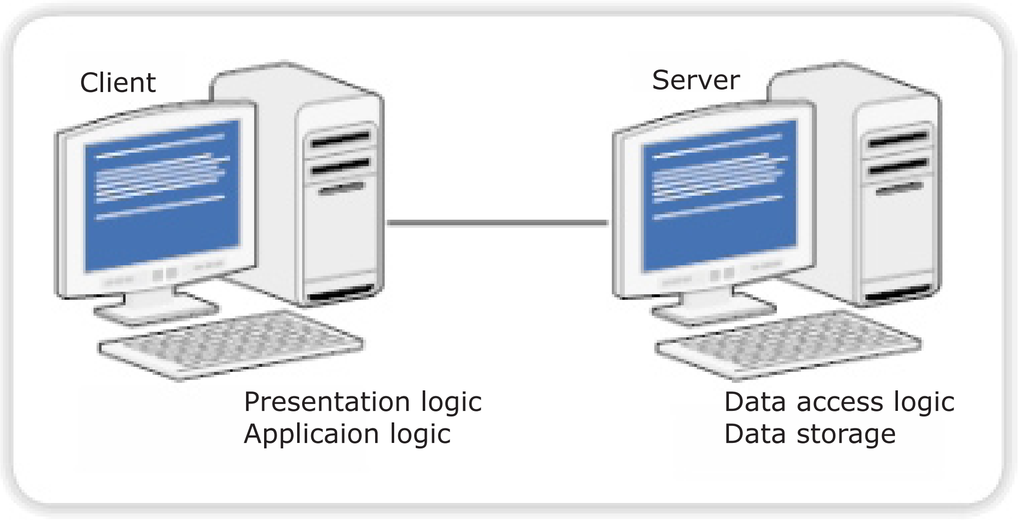


Table 9.3: Fat and Thin Clients Characteristic

Client-server architectures have several important benefits:

- They are **scalable** which means it is easy to increase or decrease the storage and processing capabilities of the servers. If one server becomes overloaded, you simply add another server so that many servers are used to perform the application logic, data access logic, or data storage. The cost to upgrade is much more gradual, and you can upgrade in smaller steps rather than spending hundreds of thousands to upgrade a mainframe server.

- Client-server architectures can support many different types of clients and servers. It is possible to connect computers that use different operating systems so that users can choose which type of computer they prefer (e.g., combining both Windows computers and Apple Macintoshes on the same network).

- **Middleware** is a type of system software designed to translate between different vendors' software. Middleware provides a transparent interface that enables system designers to integrate dissimilar software and hardware. Middleware is installed on both the client computer and the server computer. The client software communicates with the middleware which can reformat the message into a standard language. This message can be understood by the middleware that assists the server software.

Because no single server computer that supports all the applications, the network is generally more reliable. There is no central point of failure that will halt the entire network if it fails, as there is in a server-based architecture. If any of the servers fails in a client-server environment, the network can continue to function using all the other servers (but, of course, any applications that require the failed server will not work).

Client-server architectures also have some critical limitations, the most important of which is its complexity. All applications in client-server computing have two parts, the software on the client and the software on the server. Writing this software is more complicated than writing the traditional all-in-one software used in server-based architectures. Updating the network with a new version of the software is more complicated, too. In server-based architectures, there is only one place to update the application software. With client-server architectures, both clients and servers need to be updated.

9.2.4 Client-Server Tiers

Early client-server design was called two-tiered architecture because it uses only two sets of computers, clients and servers as shown in Figure 9.3. In this case, the server is responsible for the data, and the client is responsible for the application and presentation.

A three-tiered architecture uses three sets of computers as shown in Figure 9.4. In this case, the software on the client computer is responsible for presentation logic, an application server(s) is responsible for the application logic, and a separate database server(s) (usually mainframe or minicomputer) is responsible for the data access logic and data storage.

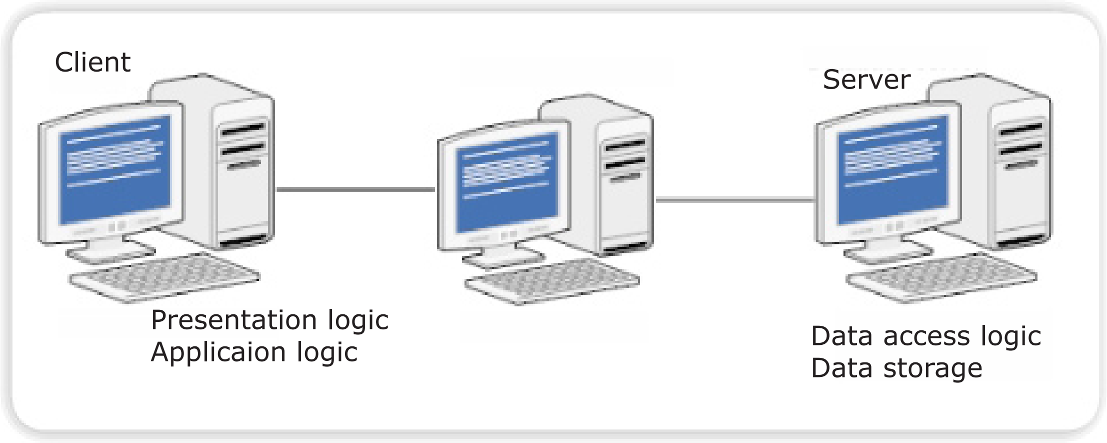


Figure 9.4: A Three-tier client-server architecture

An **n-tiered architecture** uses more than three sets of computers. In this case, the client is responsible for presentation, a database server(s) (usually mainframe or minicomputer) is responsible for the data access logic and data storage, and the application logic is spread across two or more different sets of servers. Figure 9.5 shows an example of an n-tiered architecture.

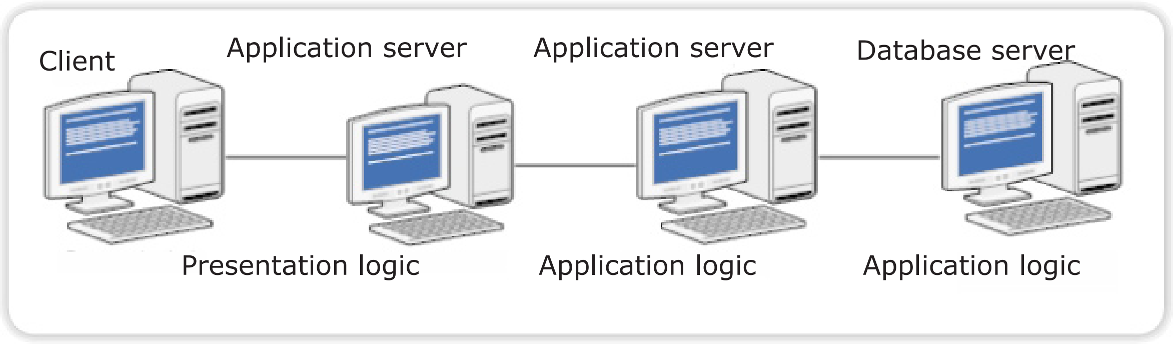


Figure 9.5: A Four-tier client-server architecture

The primary advantage of n-tiered client-server architecture is that it separates out the processing that occurs to better balance the load on the different servers which make it more scalable. In Figure 9.5, we have three separate servers, a configuration that provides more power than if we had used a two-tiered architecture with only one server. If we discover that the application server is too heavily loaded, we can simply replace it with a more powerful server, or just put in several more application servers. Conversely, if we discover the database server is underused, we could store data from another application on it.

There are two primary disadvantages to an n-tiered architecture:

The configuration puts a greater load on the network. If you compare Figures 9.3, 9.4 and 9.5 you will see that the n-tiered model requires more communication among the servers. It generates more network traffic, so you need a higher-capacity network.

It is much more difficult to program and test software in n-tier architectures than in two-tiered architectures because more devices have to communicate to complete a user's transaction.

ACTIVITY

What is the biggest problem with server-based and client-based architecture?

9.3 CHOOSING ARCHITECTURE

Most systems are constructed to use the existing infrastructure in the organisation. Sometimes the current infrastructure restricts the choice of architecture. For example, if the new system is to be built for a mainframe-centric organisation, a server-based architecture may be the best option. Other factors like corporate standards, existing licensing agreements, and product/vendor relationships can influence what architecture the project team needs to design. However, many organisations now have a variety of infrastructures available or are openly looking for pilot projects to test new architectures and infrastructures, enabling a project team to select an architecture based on other important factors.

Each of the computing architectures just discussed has its strengths and weaknesses, and no architecture is inherently better than the others. Thus, it is important to understand the strengths and weaknesses of each of the computing architectures and when to use each one of them. Table 9.4 presents a summary of the important characteristics of each of the architecture.

Table 9.4: Computing Architecture Characteristics

Characteristic	Server-based	Client-based	Client-server
Infrastructure cost - is cost of software, hardware and network that will support application systems. For example personal computers are more than 1,000 times cheaper than mainframes for the same amount of computing power. The personal computers on our desks today have more processing power, memory, and hard disk space than the typical mainframe of the past, and the cost of the personal computers is a fraction of the cost of the mainframe.	Very high	Medium	Low
Development cost - .The cost of developing systems is an important factor when considering the financial benefits of client–server architectures. Today developing application software is faster and cheaper, because there are many graphical user interface (GUI) development tools for simple stand-alone computers that communicate with database servers (e.g., Visual Basic, Access).	Medium	Low	High
Development ease - client-based and client-server architectures can rely on GUI development tools that can be intuitive and easy to use as compared to server-based architecture. The development of applications for these architectures can be fast and painless. Unfortunately, the applications for client-server can be very complex because they must be built for several layers of hardware (e.g., database servers, Web servers, client workstations) that need to communicate	Low	High	Low-medium

effectively with each other. Project teams often underestimate the effort involved in creating secure, efficient client–server applications.			
Interface capabilities - Today, most users of systems expect a graphical user interface (GUI) or a Web-based interface, which they can operate using a mouse and graphical objects (e.g., pushbuttons, drop-down lists, icons, and so on). GUI and Web development tools typically are created to support client-based or client-server applications; rarely can server-based environments support these types of applications.	Low	High	High
Control and Security - The server-based architecture was originally developed to control and secure data, and it is much easier to administer because all of the data are stored in a single location. In contrast, client–server computing requires a high degree of coordination among many components, and the chance for security holes or control problems is much more likely.	High	Low	Medium
Scalability - Refers to the ability to increase or decrease the capacity of the computing infrastructure in response to changing capacity needs. The most scalable architecture is client–server computing because servers can be added to (or removed from) the architecture when processing needs change. Also, the types of hardware that are used in client-server (e.g., minicomputers) typically can be upgraded at a pace that most closely matches the growth of the application.	Low	Medium	High

9.4 HARDWARE AND SOFTWARE SPECIFICATION

The time to begin acquiring the hardware and software that will be needed for the future system is during the design of the system. In many cases, the new system will simply run on the existing equipment in the organisation. Other times, however, new hardware and software (usually for servers) must be purchased. The hardware and software specification is a document that describes what hardware and software are needed to support the application. The actual acquisition of hardware and software should be left to the purchasing department or the area in the organisation that handles capital procurement. However, the analyst can use the hardware and software specification to communicate the project needs to the appropriate people. Figure 9.6 shows the several steps involved in creating document.

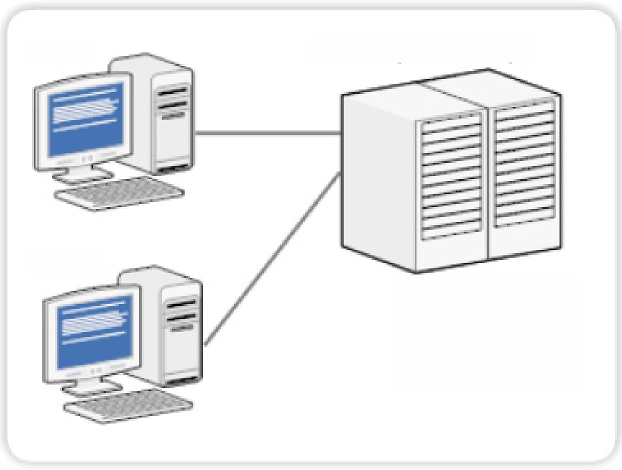


Figure 9.6: Steps to creating document

This step becomes easier with experience however, there are some hints that can help you describe hardware needs as shown in Table 9.5.

For example, consider the hardware standards within the organisation or those recommended by vendors. Talk with experienced system developers or other companies with similar systems. Finally, think about the factors that affect hardware performance, such as the response time expectations of the users, data volumes, software memory requirements, the number of users accessing the system, the number of external connections, and growth projections.

Factors	Description
i.Functions and Features.	What specific functions and features are needed (e.g., size of monitor, software features).
ii.Performance	How fast the hardware and software operates (e.g., processor, number of database writes per second).
iii.Legacy databases (old computer system)and current systems	How well the hardware and software interacts with legacy systems (e.g., can it write to this database).
iv.Hardware and OS Strategy	What are the future migration plans (e.g., the goal is to have all of one vendor's equipment) .
v.Cost of Ownership	What are the costs beyond purchase (e.g., incremental license costs, annual maintenance, training costs, salary costs).
vi.Political Preferences	People are creatures of habit and are resistant to change, so changes should be minimised.
vii.Vendor Performance	Some vendors have reputations or future prospects that are different from those of a specific hardware or software system they currently sell.

ACTIVITY

1. Define scalable. Why is this term important to system developers?

9.5 FINALISING DESIGN SPECIFICATION

As you know, much of the work in analysis and in the early stages of design involves gathering an enormous amount of information about the target system. In earlier topics, you have seen much of this information has been structured as diagrams and models physically and logically. In projects where detailed design specification are to be created and passed on to programmers, however, even more aspects of the system must be specified. The result of all this detailed design work is the system design specification, sometimes known as technical design specification or detailed design specification or software requirement specification. The system design specification is one of the major deliverables from the design phase of the system development life cycle.

9.5.1 System Design Specification

System design specification is a document that presents the complete design for the new information system, along with detailed costs, staffing and scheduling for completing the SDLC next phase (system implementation). Though time consuming, the process of developing system design specification is extremely important for projects. It is much easier and much cheaper to detect and correct errors at this point than it is to detect and correct errors during implementation.

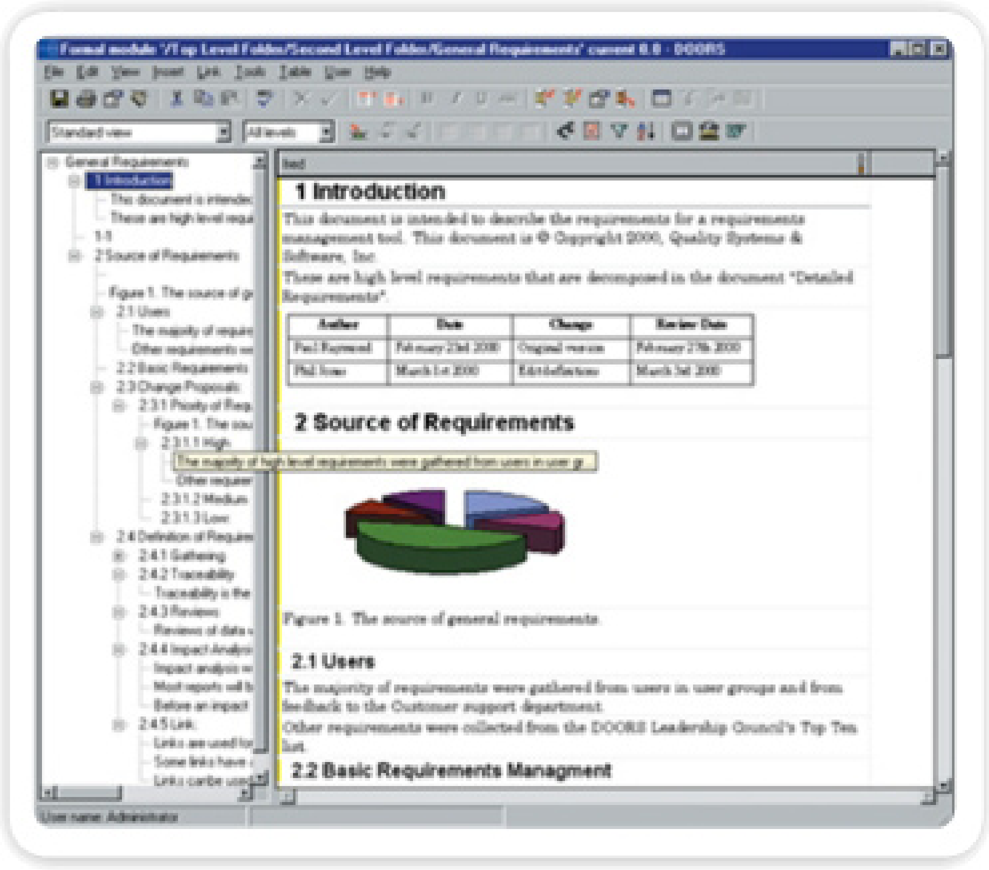
System design specification is oriented towards the programmers who will use it to create the necessary programs. Some sections in the analysis phase are repeated in the system design specification such as system requirement, process and logic modeling. The format of and amount of detail in a system design specification will be driven largely by its intended audience. If the specification is to be handed over to other members of a small team, then a high-level document created in a spreadsheet or a word processing document is probably enough. On the other hand, if the specification is to be handed over to a team, or several teams, of programmers several continents and time zones away, where there will be little direct interaction between the programmers and the designers then an extreme level of detail is called for.

Regardless of the intended audience, the process of finalising system design specification involves moving from logical design specifications to requirements that are specific enough to write code from. Table 9.6 shows a typical system design specification uses a structure.

Table 9.6: System Design Specification

Specification	Description
Executive Summary	The management summary provides a brief overview of the project for company managers and executives. It outlines the development efforts to date, provides a current status report, summarises current project costs and costs for the remaining phases, reviews the overall benefits of the new system, presents the systems development phase schedule, and highlights any issues that management will need to address.
System Components	This section contains the complete design for the new system, including the user interface, outputs, inputs, files, databases, and network specifications. You should include source documents, report and screen layouts, DFDs, and all other relevant documentation. You also should include the requirements for all support processing, such as backup and recovery, startup processing, and file retention. If the purchase of a software package is part of the strategy, you must include any interface information required between the package and the system you are developing. If you use a CASE design tool, you can print design diagrams and most other documentation directly from the tool.
System Environment	This section describes the constraints, or conditions, affecting the system, including any requirements that involve operations, hardware, systems software, or security. Hardware and software specification document will be in this structure too. Examples of operational constraints include transaction volumes that must be supported, data storage requirements, processing schedules, reporting deadlines, and online response times.
System Environment	In this section, you specify startup processing, initial data entry or acquisition, user training requirements, and software test plans.
Time and Cost Estimates	This section provides detailed schedules, cost estimates and staffing requirements for the systems development phase and revised projections for the remainder of the SDLC. You also present total costs-to-date for the project and compare those costs with your prior estimates.
Appendices	Supplemental material can be included in appendices at the end of the system design specification. In this section, you might include copies of documents from the first three phases if they would provide a helpful reference for readers.

System design specification can be extremely complicated phase. Many different pieces must work together and many different parts of the organisation are affected by a single system. Given the time and effort necessary to collect and spell out all of this information, it should not be surprising that organisations and developers have been looking for ways to make the creation and maintenance of these documents faster and cheaper. One method that has been developed is computer-based requirement management tools as shown in Figure 9.6 and Figure 9.7. These tools make it easier for analysts to keep requirement documents up to date, to add additional information about specifications and to define links between different parts of the overall specifications package.


Formal module '/Top Level Folder/Second Level Folder/General Requirements' current 0.0 - DOORS
File Edit View Insert Link Tools Table User Help
Standard view
All levels
User name: Administrator

1 Introduction

This document is intended to describe the requirements for a requirements management tool. This document is © Copyright 2000, Quality Systems & Software, Inc.

These are high level requirements that are decomposed in the document "Detailed Requirements".

Author	Date	Change	Review Date
Paul Raymond	February 23rd 2000	Original version	February 27th 2000
Phil Ivens	March 1st 2000	Edit/deflections	March 3rd 2000

2 Source of Requirements

The majority of high level requirements were gathered from users in user g...

Figure 1. The source of general requirements

2.1 Users

The majority of requirements were gathered from users in user groups and from feedback to the Customer support department.

Other requirements were collected from the DOORS Leadership Council's Top Ten list.

2.2 Basic Requirements Managment

Figure 9.7: A screen shot from DOORS Enterprise Requirement Suite
Source: www.telelogic.com

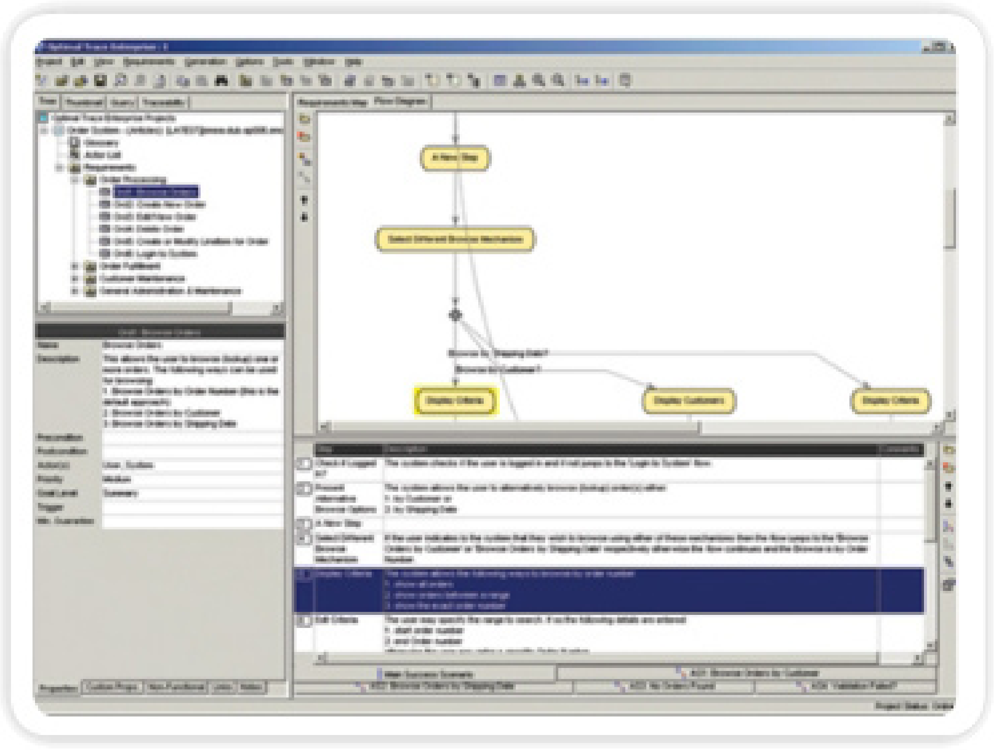
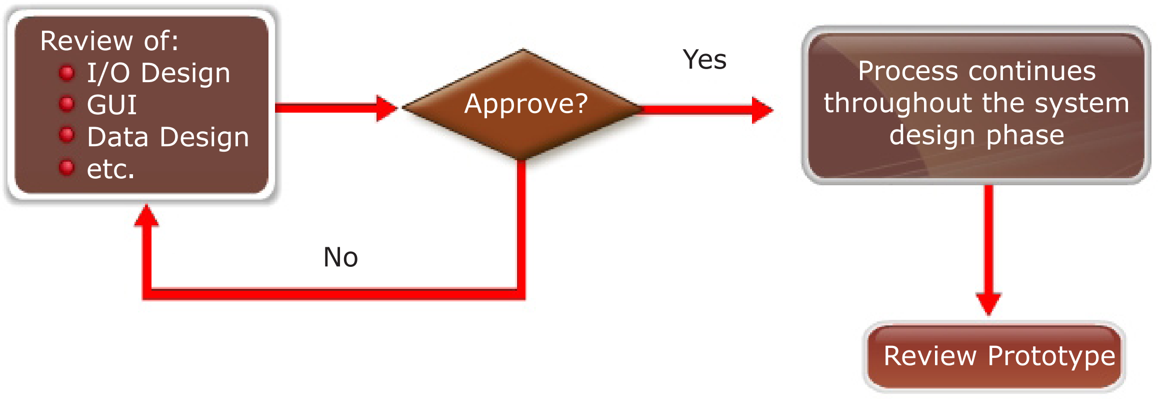


Figure 9.8: A screen shot from Compuware Optimal Trace requirement
Source: www.compuware.com

System design specification can be extremely complicated phase. Many different pieces must work together and many different parts of the organisation are affected by a single system. Given the time and effort necessary to collect and spell out all of this information, it should not be surprising that organisations and developers ha ve been looking for ways to make the creation and maintenance of these documents faster and cheaper. One method that has been developed is computer-based requirement management tools as shown in Figure 9.6 and Figure 9.7. These tools make it easier for analysts to keep requirement documents up to date, to add additional information about specifications and to define links between different parts of the overall specifications package.

9.5.2 User Approval

Users must review and approve the input output design, interface design, data design, architecture design and other areas of the system that affect them. The review and approval process continues throughout the system design phase. When you complete the design for a report, you should meet with the users to review the prototype, adjust the design if necessary and obtain written approval.



Securing approvals from users throughout the design phase is very important. That approach ensures that you do not have a major task of obtaining approvals at the end. It keeps the users involved with the system's development and gives you feedback about whether or not you are on target. Some sections of the system design specification might not interest users, but anything that does affect them should be approved as early as possible.

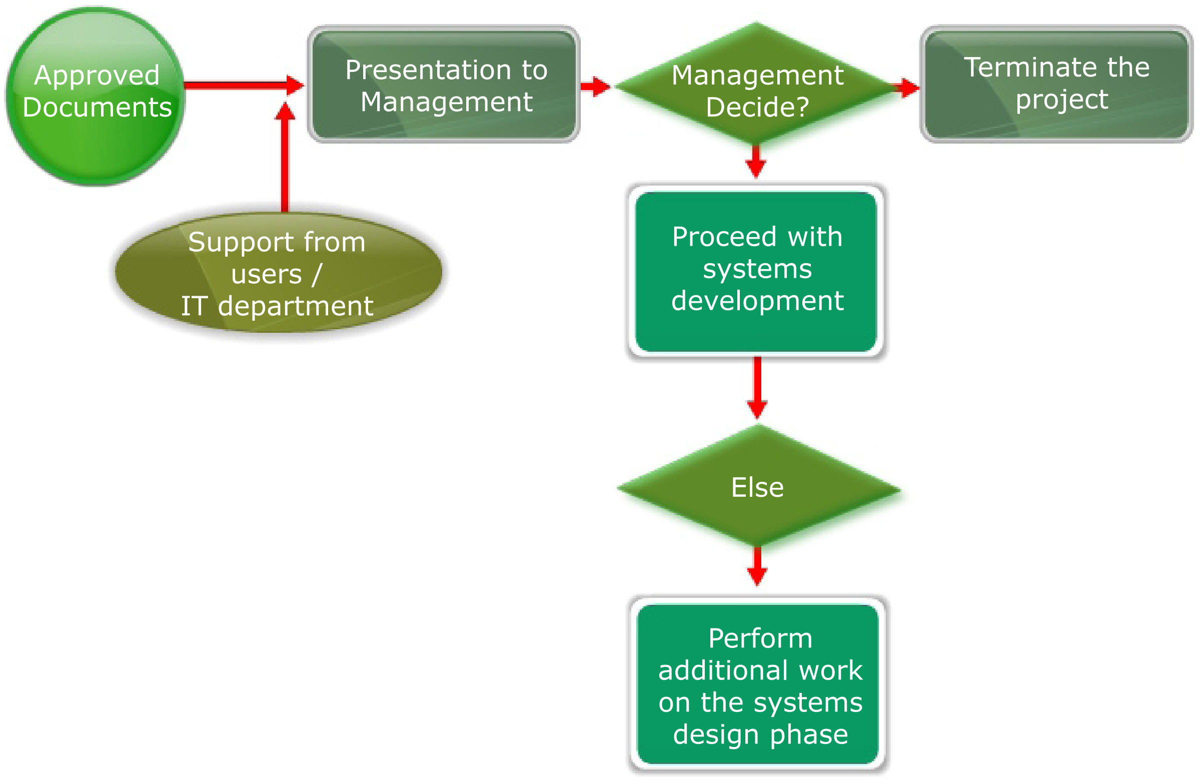
Other IT department members also need to review the system design specification. IT management will be concerned with staffing, costs, hardware and systems software requirements, network impact, and the effect on the operating environment when the new system is added. The programming team will want to get ready for its role, and the operations group will be interested in processing support, report distribution, network loads, integration with other systems, and any hardware or software issues for which they need to prepare. You must be a good communicator to keep people up-to-date, obtain their input and suggestions, and obtain necessary approvals.

When the system design specification is complete, you distribute the document to a target group of users, IT department personnel, and company management. You should distribute the document at least one week before your presentation to allow the recipients enough time to review the material.

9.5.3 Presentation

Usually, you will give several presentations at the end of the systems design phase. The presentations give you an opportunity to explain the system, answer questions, consider comments, and secure final approval. The first presentation is to the systems analysts, programmers, and technical support staff members who will be involved in future project phases or operational support for the system. Because of the audience, the presentation is technically oriented.

Your next presentation is to department managers and users from departments affected by the system. As in the first presentation, your primary objective is to obtain support and approval for the systems design. This is not a technical presentation. It is aimed at user interaction with the system and management's interest in budgets, schedule, staffing, and impact on the production environment.


Approved Documents
Presentation to Management
Management Decide?
Terminate the project
Support from users / IT department
Proceed with systems development
Else
Perform additional work on the systems design phase

The final presentation is for company management. By the time you give this presentation, you should have obtained all necessary approvals from prior presentations, and you should have the support of users and the IT department. Based on the presentation and the data you submitted, management might reach one of three decisions: proceed with systems development, perform additional work on the systems design phase, or terminate the project.

SUMMARY

- An important component of system design is the design of system architecture, which includes the hardware, software and network for the new system and the way in which the information system will be distributed over the architecture. Each of the architecture has its strength and weaknesses and no architecture is inherently better than the others. The choices of architecture selection are based on several criteria including infrastructure cost, development cost, development ease, interface capability, security control and scalability.
- Hardware and software specification document describes on what hardware and software are needed to support the application. Finally, design specifications were finalised with system design specification document to present the complete system design for an information system. It is the basis on which users approve of the system design specification and presentations that marks the completion of systems design phase.

KEY TERMS

Client

Client-Based Architectures

Client–Server Architectures

Fat client

Hardware and software specification

Middleware

Network

N-tiered architecture

Scalable

Server-Based Architectures

Thin client

Three-tiered architecture

Two-tiered architecture

REFERENCES

1. Alan D. & Barbara H.W, System Analysis and Design with UML (John Wiley & Sons, 2000)
2. Brett C. Tjaden, Fundamentals of Secure Computer Systems (Wilsonville, OR: Franklin, Beedle, and Associates, 2004)
3. Irv Englander, The Architecture of Computer Hardware and Systems Software: An Information Technology Approach, 3rd Ed. (New York: Wiley, 2003)
4. Stephen D. Burd, Systems Architecture, 4th Ed. (Boston: Course Technology, 2003)
5. William Stallings, Computer Organisation & Architecture: Designing for Performance (Upper Saddle River, NJ: Prentice Hall, 2003)