Systems Analysis and Design (CACS203)

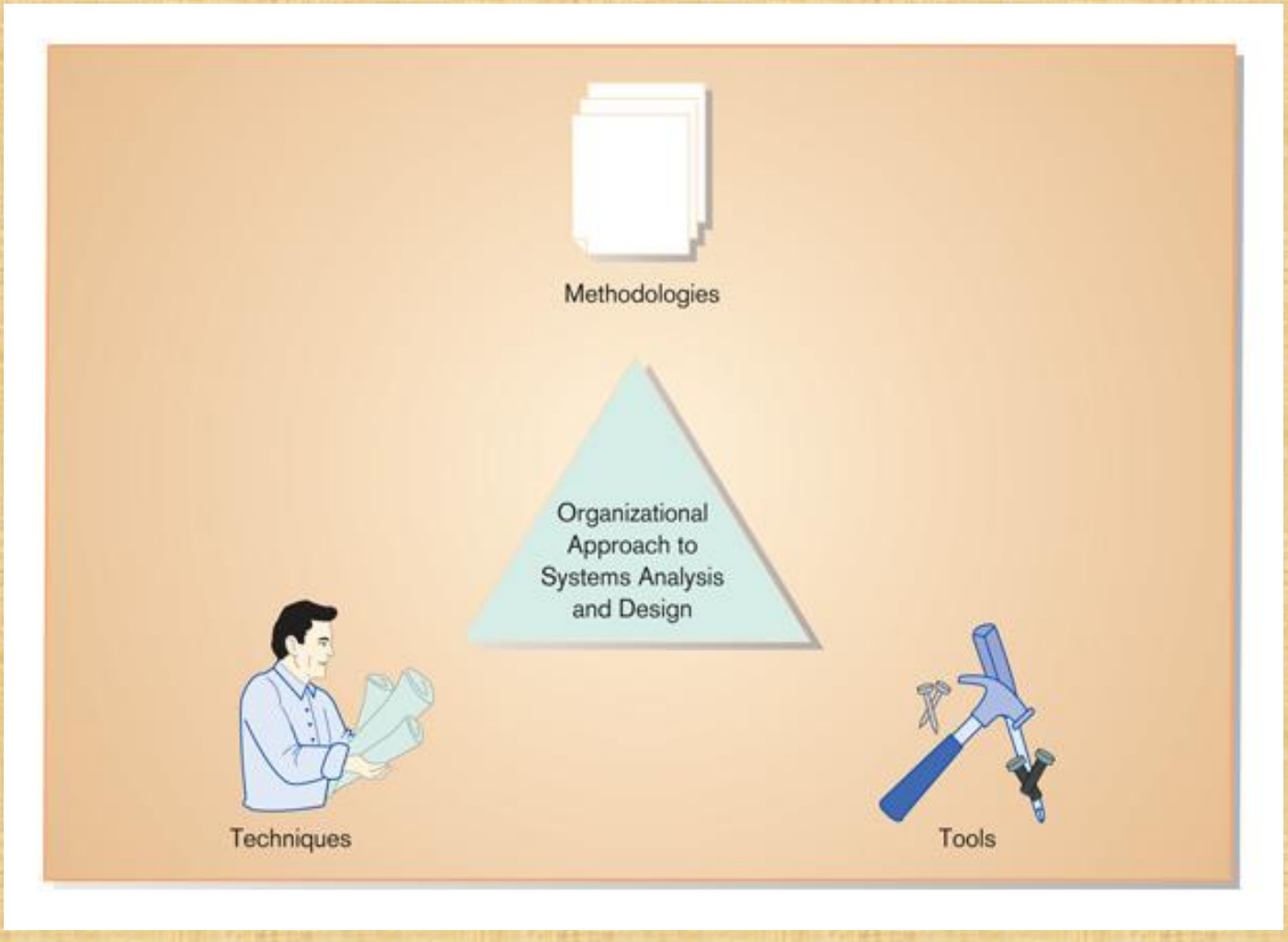
- BCA Third Semester
- Compiled By G.P.Lekhak

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.

The Systems Development Environment

Introduction

- Information Systems Analysis and Design (**ISAD**)
  - Complex organizational process.
  - Used to develop and maintain computer-based information systems.
  - Used by a team of business and systems professionals.
  - An organizational improvement process (respond to and anticipate problems).
  - Uses technology (Internet, WWW marketing, online business, eBay, Amazon.com etc).



- Figure 1-1 An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools

- An output of system analysis and design is **application software**.
- **Methodologies** are comprehensive, multiple-step approaches to system development. Methodologies uses different techniques.
- **Techniques** are particular processes that you as an analyst, will follow to ensure your work is well thought out, complete, and comprehensive to others on your project team. Eg: conducting interviews to determine what your system should do, diagramming the system's logic, designing the reports your system will generate.

- Tools are computer programs that make it easy to use and benefit from techniques.

A Modern Approach to Systems Analysis and Design

- 1950s: All applications had to be developed in machine language or assembly language. They had to be developed from scratch because due to the absence of software industry.
- 1960s: Smaller, faster, less expensive computers, beginning of the software industry, use in-house development.
- 1970s: Realized how expensive to develop customized information system for every application , started development of database management system.
- 1980s: , The software industry expended greatly, CASE(computer aided software engineering) tools.

- Started writing application software in oop languages, graphics were used, developed less software in-house and bought more from software vendors.
- 1990s: focus on system integration, GUI(Graphical user interface) applications, client/server platforms, Internet.
- The new century: Web application development, wireless PDAs (personal digital assistants, eg pocket PCs), ASP(application service provider).

- Application Software
  - Computer software designed to support organizational functions or processes.
- Systems Analyst
  - Organizational role most responsible for analysis and design of information systems.

Types of Information Systems and Systems Development

- Transaction Processing Systems (TPS)
  - Automate handling of data about business activities (transactions)
  - Process orientation
- Management Information Systems (MIS)
  - Converts raw data from transaction processing system into meaningful form
  - Data orientation

- Decision Support Systems (DSS)
  - Designed to help decision makers
  - Provides interactive environment for decision making
  - Involves data warehouses, executive information systems (EIS)
  - Database, model base, user dialogue

Summary of Information Systems Types

Table 1-1 Systems Development for Different IS Types

IS Type	*IS Characteristics*	*Systems Development Methods*
Transaction processing system	High-volume, data capture focus; goal is efficiency of data movement and processing and interfacing different TPSs	Process orientation; concern with capturing, validating, and storing data and with moving data between each required step
Management information system	Draws on diverse yet predictable data resources to aggregate and summarize data; may involve forecasting future data from historical trends and business knowledge	Data orientation; concern with understanding relationships among data so data can be accessed and summarized in a variety of ways; builds a model of data that supports a variety of uses
Decision support system	Provides guidance in identifying problems, finding and evaluating alternative solutions, and selecting or comparing alternatives; potentially involves groups of decision makers; often involves semi-structured problems and the need to access data at different levels of detail	Data and decision logic orientations; design of user dialogue; group communication may also be key, and access to unpredictable data may be necessary; nature of systems requires iterative development and almost constant updating

Developing Information Systems

- **System Development Methodology** is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

Systems Development Life Cycle (SDLC)

- Traditional methodology used to develop, maintain, and replace information systems.
- Phases in SDLC:
  - Planning
  - Analysis
  - Design
  - Implementation
  - Maintenance

Standard and Evolutionary Views of SDLC

Figure 1-3 The systems development life cycle

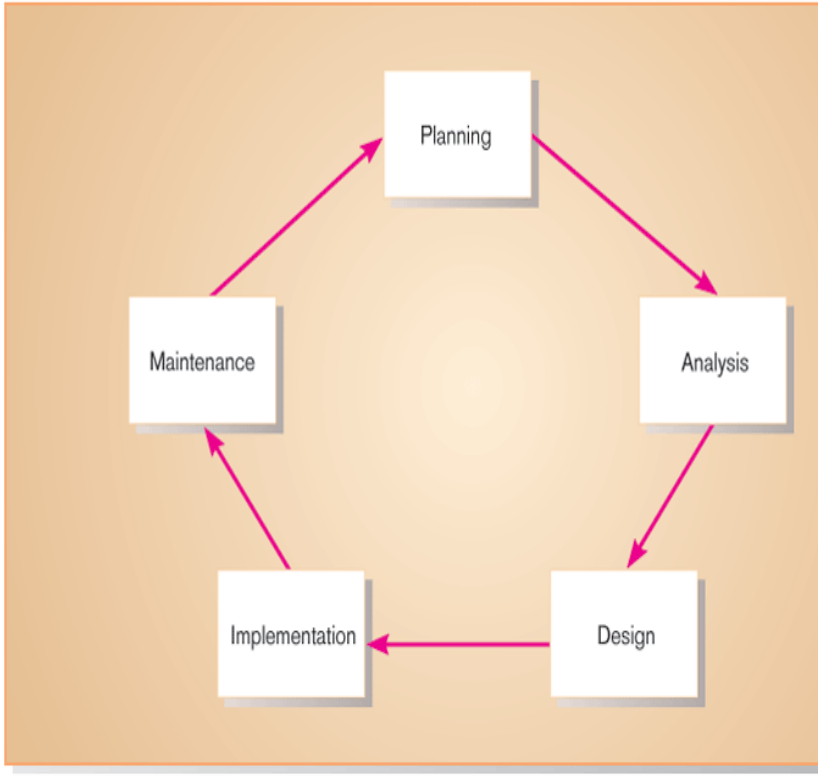
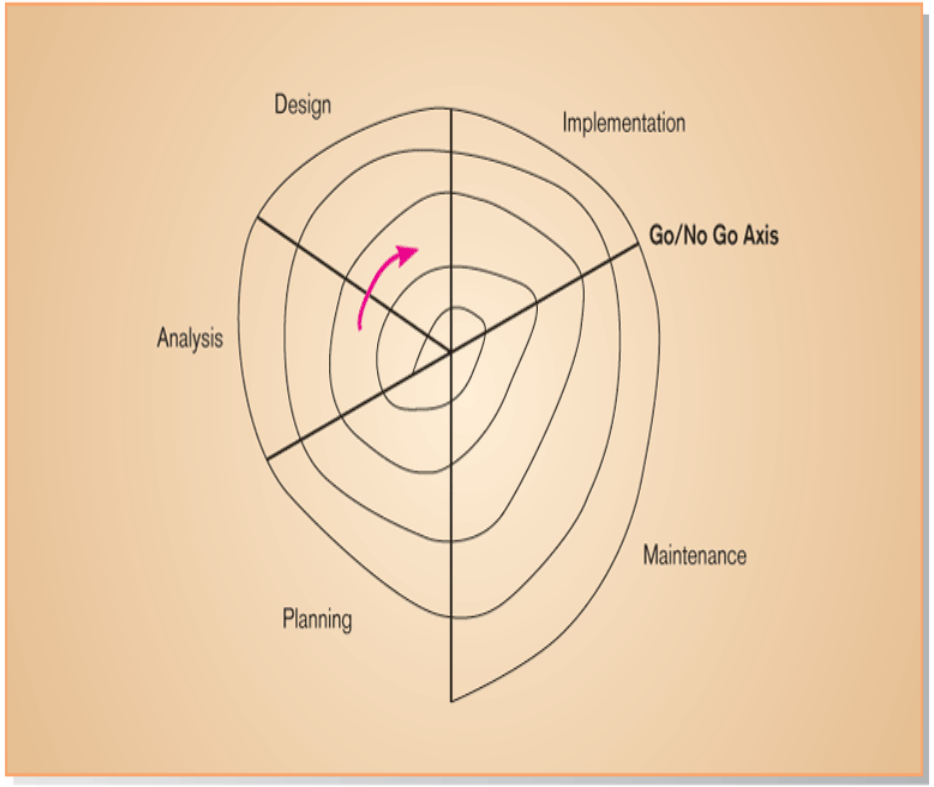


Figure 1-4 Evolutionary model SDLC



Systems Development Life Cycle (SDLC) (Cont.)

- **Planning** – an organization's total information system needs are identified, analyzed, prioritized, and arranged. The outcome of the project identification and selection process is a determination of which system development project should be undertaken by the organization, at least in terms of initial study.
- Major activities during planning are: **i)** Investigation of the system problem or opportunity. **ii)** presentation of reasons why the system should or should not be developed by the organization. Determining the **scope** of the proposed system

- Also produce a specific plan for the proposed project the team will follow. This specifies the time and resources needed for execution.
- Also identify whether the costs of developing system would give benefit.

- The second phase in the SDLC is **analysis.** During this phase, the analyst thoroughly studies the organization's current procedures and the information systems used to perform organizational tasks. Analysis has **two sub phases**. The first is requirements determination. In this sub phase, analysts work with users to determine what the users want from a proposed system. The requirements determination process usually involves a careful study of any current systems, manual and computerized, that might be replaced or enhanced as part of the project. In the second part of analysis, analysts study the requirements and structure them according to their interrelationships and eliminate

any redundancies. The output of the analysis phase is a description of (but not a detailed design for) the alternative solution recommended by the analysis team. Once the recommendation is accepted by those with funding authority, the analysts can begin to make plans to acquire any hardware and system software necessary to build or operate the system as proposed.

- The third phase in the SDLC is **design. During design, analysts convert the** description of the recommended alternative solution into **logical** and then **physical** system specifications. The analysts must design all aspects of the system, from input and output screens to reports, databases, and computer processes. The analysts must then provide the physical specifics of the system they have designed, either as a model or as detailed documentation, to guide those who will build the new system.
- That part of the design process that is independent of any specific hardware or software platform is referred to as **logical design. Theoretically, the system could** be implemented on any hardware and systems software. The idea is to make sure that the system functions as intended.

- Once the overall high-level design of the system is worked out, the analysts begin turning **logical specifications into physical ones**. This process is referred to as **physical design.** As part of physical design, analysts design the various parts of the system to perform the physical operations necessary to facilitate data capture, processing, and information output. This can be done in many ways, from creating a working model of the system to be implemented to writing detailed specifications describing all the different parts of the system and how they should be built.

  In many cases, the working model becomes the basis for the actual system to be used. During physical design, the analyst team must determine many of the physical details necessary to build the final system, from the programming language the system will be written in, to the database system that will store the data, to the hardware platform on which the system will run.

- Often the choices of language, database, and platform are already decided by the organization or by the client, and at this point these information technologies must be taken into account in the physical design of the system. The final product of the design phase is the physical system specifications in a form ready to be turned over to programmers and other system builders for construction. Figure 1-6 illustrates the differences between logical and physical design.

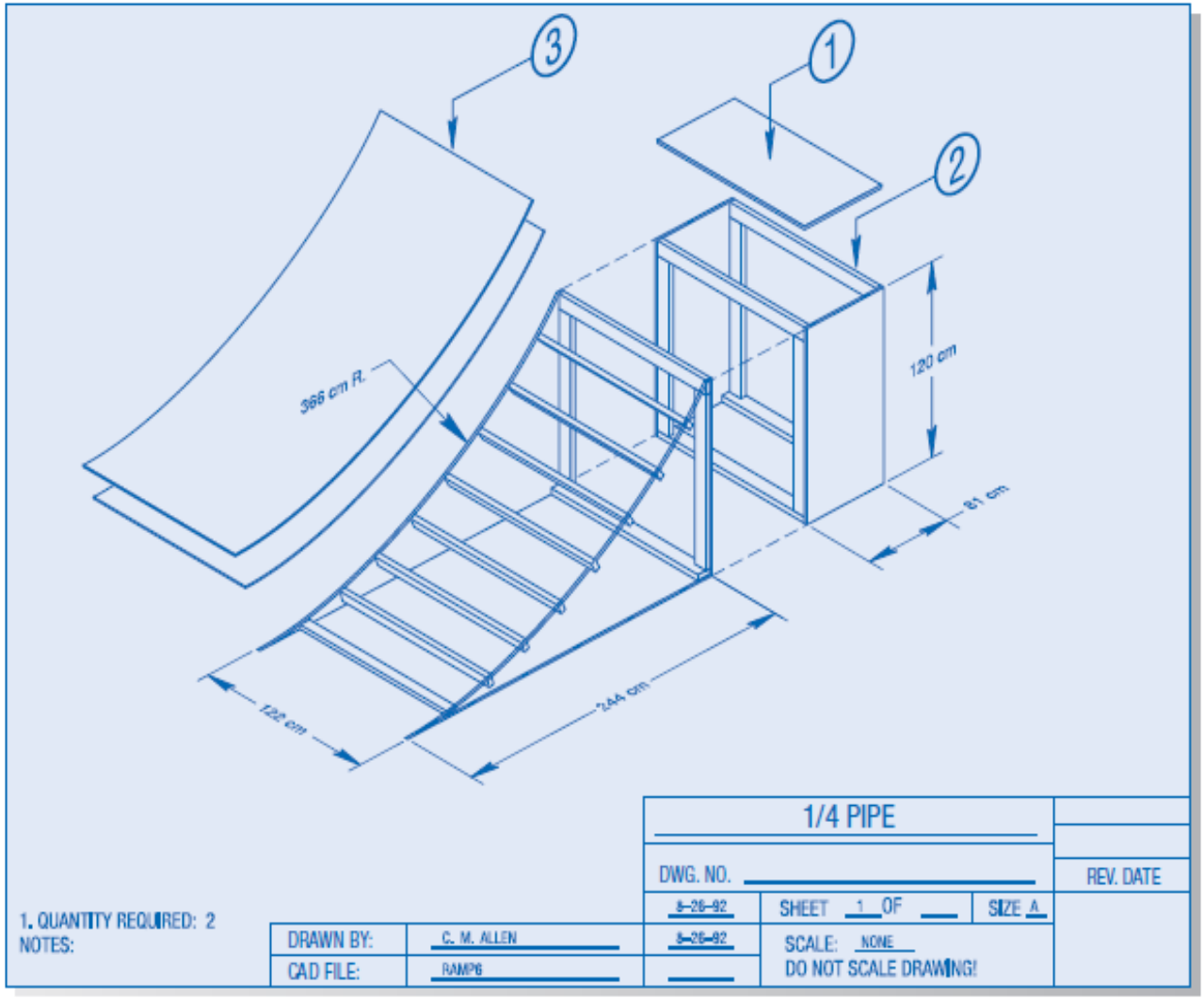
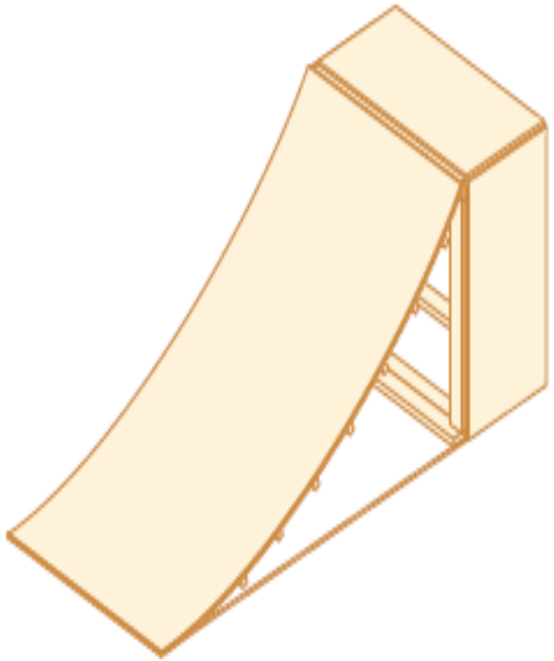


FIGURE 1-6
Difference between logical design and physical design
(a) A skateboard ramp blueprint (logical design)

(*Sources: www.tumyeto.com/tydu/skatebrd/organizations/plans/14pipe.jpg; www.tumyeto.com/tydu/skatebrd/organizations/plans/iuscblue.html.* Accessed September 16, 1999. Reprinted by permission of the International Association of Skateboard Companies.)



(b) A skateboard ramp (physical design)

- The fourth phase in the SDLC is **implementation.** The physical system specifications, whether in the form of a detailed model or as detailed written specifications, are turned over to programmers as the first part of the implementation phase. During implementation, analysts turn system specifications into a working system that is tested and then put into use. **Implementation includes coding, testing, and installation**.
- During **coding**, programmers write the programs that make up the system. Sometimes the code is generated by the same system used to build the detailed model of the system.
- During **testing**, programmers and analysts test individual programs and the entire system in order to find and correct errors.
- During **installation**, the new system becomes part of the daily activities of the organization. Application software is installed, or loaded, on existing or new hardware, and users are introduced to the new system and trained. Testing and installation should be planned for as early as the project initiation and planning phase; both testing and installation require extensive analysis in order to develop exactly the right approach.

- Finalization of documentation, training programs
- Note that documentation and training programs are finalized during implementation; documentation is produced throughout the life cycle.

- The fifth and final phase in the SDLC is **maintenance. When a system (including** its training, documentation, and support) is operating in an organization, users sometimes find problems with how it works and often think of better ways to perform its functions. Also, the organization's needs with respect to the system change over time. **In maintenance, programmers make the changes that users ask for and modify the system** to reflect evolving business conditions. These changes are necessary to keep the system running and useful. In a sense, maintenance is not a separate phase but a repetition of the other life cycle phases required to study and implement the needed changes. The amount of time and effort devoted to maintenance depends a great deal on the performance of the previous phases of the life cycle.

  when an information system is no longer performing as desired, when maintenance costs become prohibitive, or when an organization's needs have changed substantially.

Such problems indicate that it is time to begin designing the system's replacement, thereby completing the loop and starting the life cycle over again.

The heart Of The System Development Proscess

- Do your self……..

The Heart of the Systems Development Process

Figure 1-8 The analysis–design–code–test loop

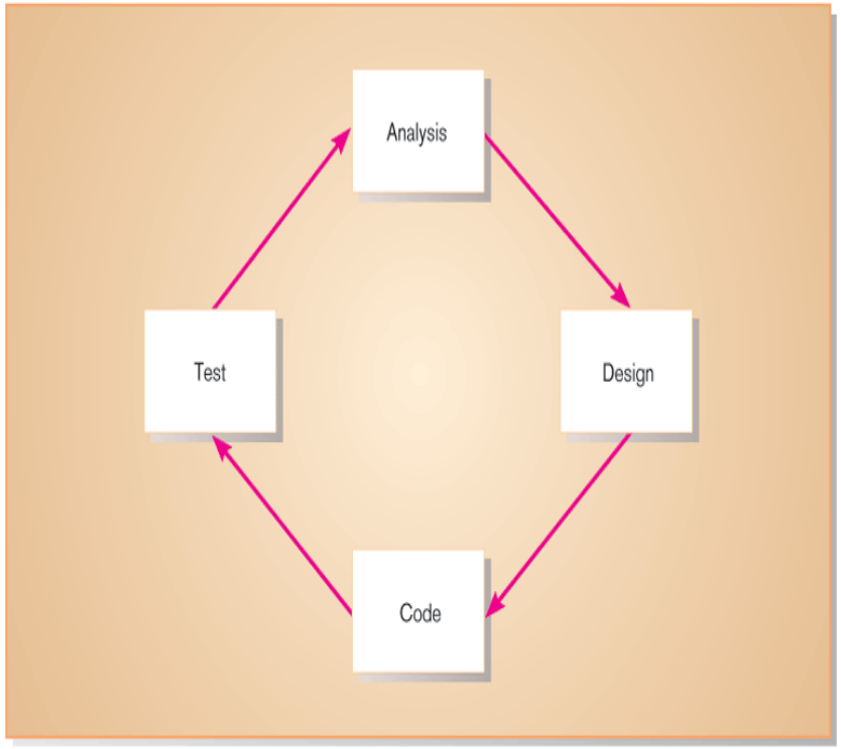
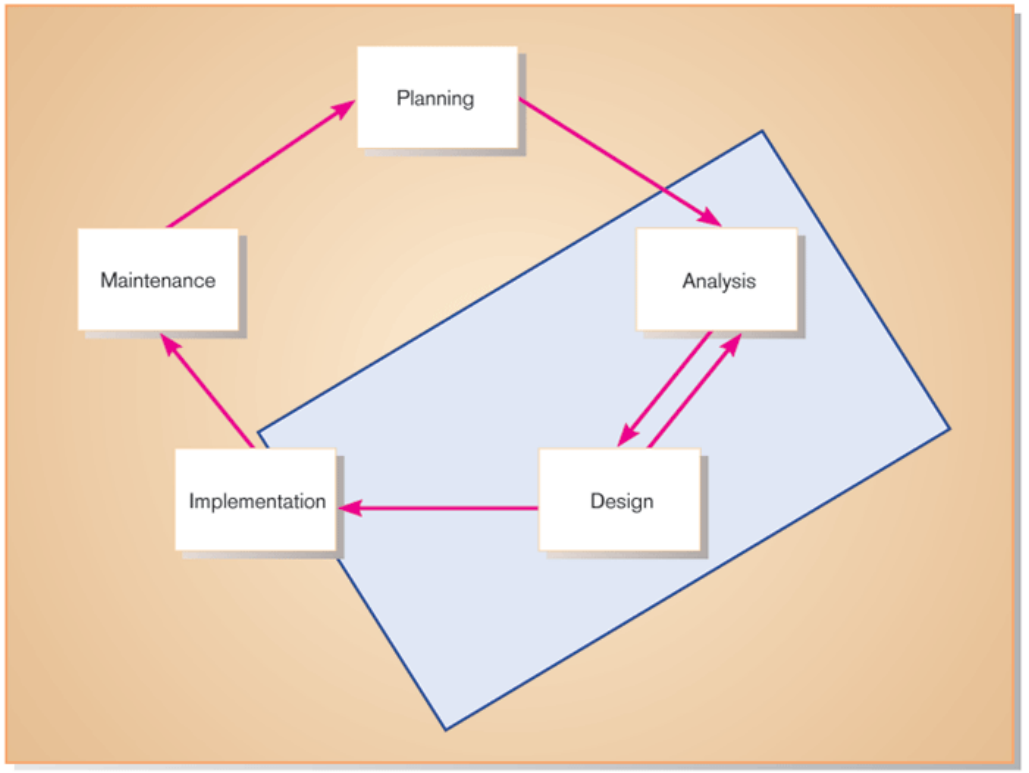


Figure 1-9 The heart of systems development



Current practice combines analysis, design, and implementation into a single iterative and parallel process of activities

Traditional Waterfall SDLC

- Treat each phase as complete unto itself, never to be revisited once finished.
- Feedback came to be ignored in implementation.
- Traditionally, one phase ended and another began once a milestone had been reached.
- System requirements “locked in” after being determined (can't change). Once the milestone had been reached and the new phase initiated, it became difficult to go back. The enormous amount of effort and time necessary to implement a specific design meant that it would be very expensive to make changes in a system once it was developed.
- Limited user involvement (only in requirements phase).

- Yet another criticism of the traditional waterfall SDLC is that the role of system users or customers was narrowly defined (Kay, 2002). User roles were often relegated to the requirements determination or analysis phases of the project, where it was assumed that all of the requirements could be specified in advance. Such an assumption, coupled with limited user involvement, reinforced the tendency of the waterfall model to lock in requirements too early, even after business conditions had changed.
- In addition, under the traditional waterfall approach, nebulous and intangible processes such as analysis and design are given hard-and-fast dates for completion, and success is overwhelmingly measured by whether those dates are met.

- The focus on milestone deadlines, instead of on obtaining and interpreting feedback from the development process, leads to too little focus on doing good analysis and design. The focus on deadlines leads to systems that do not match users' needs and that require extensive maintenance, unnecessarily increasing development costs. Finding and fixing a software problem after the delivery of the system is often far more expensive than finding and fixing it during analysis and design (Griss, 2003). The result of focusing on deadlines rather than on good practice is unnecessary rework and maintenance effort, both of which are expensive. According to some estimates, maintenance costs account for 40 to 70 percent of systems development costs (Dorfman and Thayer, 1997). Given these problems, people working in systems development began to look for better ways to conduct systems analysis and design.

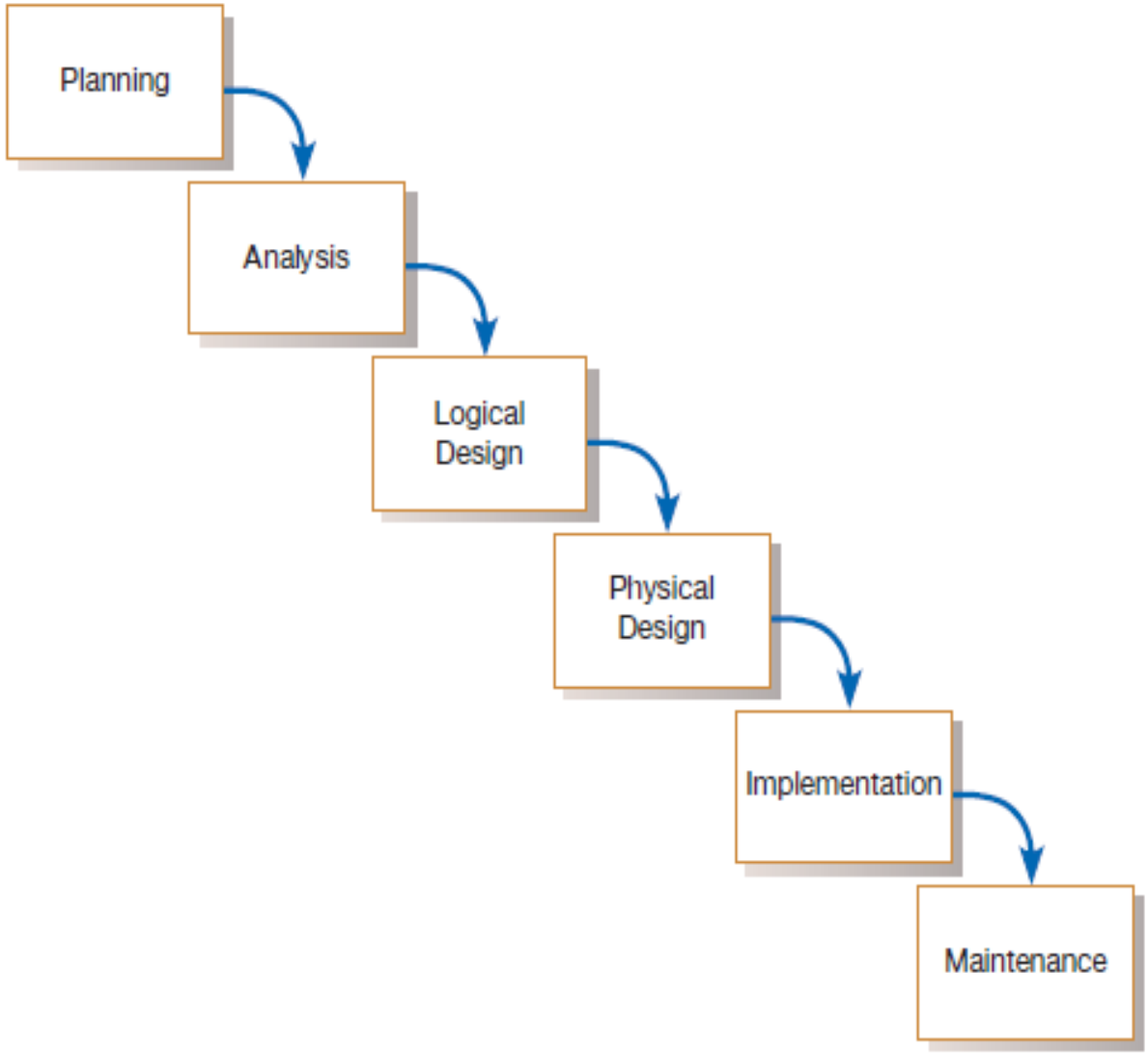


FIGURE 1-10
Traditional waterfall SDLC

Different Approaches to Improving Development

- Attempts to make systems development less of an art and more of a science are usually referred to as **systems engineering or software engineering**
- CASE TOOLS
- Rapid Application Development
- Service-Oriented Architecture
- Agile Methodologies
- eXtreme Programming
- Object-Oriented Analysis And Design

- **CASE TOOLS**: CASE tools have been developed for internal use and for sale by several leading firms. CASE tools are used to support a wide variety of SDLC activities. CASE tools can be used to help in multiple phases of the SDLC: project identification and

  selection, project initiation and planning, analysis, design, and implementation and maintenance. An integrated and standard database called a repository is the common

  method for providing product and tool integration, and has been a key factor in enabling CASE to more easily manage larger, more complex projects and to seamlessly (to be full of good material and ideas to use) integrate data across various tools and products. The idea of a central repository of information about a project is not new—the manual form of such a repository is called a project dictionary or workbook.

- The general types of CASE tools are listed below:
- Diagramming tools enable system process, data, and control structures to be represented graphically.
- Computer display and report generators help prototype how systems "look and feel." Display (or form) and report generators make it easier for the systems analyst to identify data requirements and relationships.
- Analysis tools automatically check for incomplete, inconsistent, or incorrect specifications in diagrams, forms, and reports.
- A central repository enables the integrated storage of specifications, diagrams, reports, and project management information.
- Documentation generators produce technical and user documentation in standard formats.
- Code generators enable the automatic generation of program and database definition code directly from the design documents, diagrams, forms, and reports.

TABLE 1-2 Examples of CASE Usage within the SDLC

SDLC Phase	Key Activities	CASE Tool Usage
Project identification and selection	Display and structure high-level organizational information	Diagramming and matrix tools to create and structure information
Project initiation and planning	Develop project scope and feasibility	Repository and documentation generators to develop project plans
Analysis	Determine and structure system requirements	Diagramming to create process, logic, and data models
Logical and physical design	Create new system designs	Form and report generators to prototype designs; analysis and documentation generators to define specifications
Implementation	Translate designs into an information system	Code generators and analysis, form and report generators to develop system; documentation generators to develop system and user documentation
Maintenance	Evolve information system	All tools are used (repeat life cycle)

- CASE helps programmers and analysts do their jobs more efficiently and more effectively by automating routine tasks. However, many organizations that use CASE tools do not use them to support all phases of the SDLC. Some organizations may extensively use the diagramming features but not the code generators. Table 1-2 summarizes how CASE is commonly used within each SDLC phase. There are a variety of reasons why organizations choose to adopt CASE partially or not use it at all. These reasons range from a lack of vision for applying CASE to all aspects of the SDLC, to the belief that CASE technology will fail to meet an organization's unique system development needs. In some organizations, CASE has been extremely successful, whereas in others it has not

- CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering. It means, development and maintenance of software projects with help of various automated software tools.
- CASE Tools
- CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.
- There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.
- Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.
- **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.
- **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.
- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.
- CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.
- Diagram tools : These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

- Design Tools : These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

- **Agile Methodologies:**
- According to Fowler (2003), the Agile Methodologies share three key principles: (1) a focus on **adaptive** rather than **predictive** methodologies, (2) a focus on **people** rather than **roles**, and (3) a focus on **self-adaptive** processes.

(1) Agile Methodologies share **iterative development** (Martin, 1999).

Iterative development focuses on the frequent production

of working versions of a system that have a subset of the total number of required features. Iterative development provides feedback to customers and developers alike.

(2) Systems analyst or tester or manager, are not as important as the **individuals** who fill those roles. Fowler argues that the application of engineering principles to systems development has resulted in a view of people as **interchangeable units** instead of a view of people as **talented** individuals, each bringing something unique to the development team.

(3) The Agile Methodologies promote a **self-adaptive** software development process. As software is developed, the process used to develop it should be refined and improved. Development teams can do this through a review process, often associated with the completion of iterations. The implication is that, as processes are adapted, one would not expect to find a single monolithic methodology within a given corporation or enterprise. Instead, one would find many variations of the methodology, each of which reflects the particular talents and experience of the team using it.

- Agile Methodologies are not for every project. Fowler (2003) recommends an agile or adaptive process if your project involves
  - **Unpredictable or dynamic requirements**,
  - **Responsible and motivated developers**, and
  - **Customers who understand the process and will get involved.**

- What is Agile?
- The word ‘agile’ means – Able to move your body quickly and easily.
- Able to think quickly and clearly.
- In business, ‘agile’ is used for describing ways of planning and doing work wherein it is understood that making changes as needed is an important part of the job. Business‘agililty’ means that a company is always in a position to take account of the market changes.
- Ref: Cambridge Dictionaries online.
- In software development, the term ‘agile’ is adapted to mean ‘the ability to respond to changes – changes from Requirements, Technology and People.

TABLE 1-4 Five Critical Factors That Distinguish Agile and Traditional Approaches to Systems Development

Factor	Agile Methods	Traditional Methods
Size	Well matched to small products and teams. Reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams. Hard to tailor down to small projects.
Criticality	Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products. Hard to tailor down to products that are not critical.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments.	Detailed plans and Big Design Up Front, excellent for highly stable environment but a source of expensive rework for highly dynamic environments.
Personnel	Requires continuous presence of a critical mass of scarce experts. Risky to use non-agile people.	Needs a critical mass of scarce experts during project definition but can work with fewer later in the project, unless the environment is highly dynamic.
Culture	Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom (thriving on chaos).	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear practices and procedures (thriving on order).

- **eXtreme programming:** eXtreme Programming is an approach to software development put together by Beck and Andres (2004). It is distinguished by
- **its short cycles**
- **incremental planning approach**
- **focus on automated tests written by programmers and customers** to monitor the development process.
- reliance on an **evolutionary approach** to development that lasts throughout the lifetime of the system.
- **use of two-person programming teams.**
- **Planning**, **analysis**, **design**, and **construction** are all fused into a single phase of activity.

- Under this approach, coding and testing are intimately related parts of the same process. The programmers who write the code also develop the tests. The emphasis is on testing those things that can break or go wrong, not on testing everything.
- Code is tested very soon after it is written. The overall philosophy behind eXtreme Programming is that the code will be integrated into the system it is being developed for and tested within a few hours after it has been written. If all the tests run successfully,

  then development proceeds. If not, the code is reworked until the tests are successful.

- Another part of eXtreme Programming that makes the code-and-test process work more smoothly is the practice of pair programming. All coding and testing is done by two people working together to write code and develop tests. **Beck** says that pair programming is not one person typing while the other one watches; rather, the two programmers work together on the problem they are trying to solve, exchanging information and insight and sharing skills. Compared to traditional coding practices, the advantages of pair programming include: (1) **more (and better) communication**

  among developers, (2) **higher levels of productivity, (3) higher-quality code**, and (4) reinforcement of the other practices in eXtreme Programming, such as the code and-test discipline (Beck & Andres, 2004). Although the eXtreme Programming process has its advantages, just as with any other approach to systems development, it is not for everyone and is not applicable to every project.

- **Extreme Programming – A way to handle the common shortcomings**
- Software Engineering involves –
- Creativity
- Learning and improving through trials and errors
- Iterations
- Extreme Programming builds on these activities and coding. It is the detailed (not the only) design activity with multiple tight feedback loops through effective implementation, testing and refactoring continuously.
- Extreme Programming is based on the following values –
- Communication
- Simplicity
- Feedback
- Courage
- Respect

- What is Extreme Programming?
- XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
- e**X**treme **P**rogramming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements.
- Extreme Programming is one of the Agile software development methodologies. It provides values and principles to guide the team behavior. The team is expected to self-organize. Extreme Programming provides specific core practices where −
- Each practice is simple and self-complete.
- Combination of practices produces more complex and emergent behavior.

- Why is it called “Extreme?”
- Extreme Programming takes the effective principles and practices to extreme levels.
- Code reviews are effective as the code is reviewed all the time.
- Testing is effective as there is continuous regression and testing.
- Design is effective as everybody needs to do refactoring daily.
- Integration testing is important as integrate and test several times a day.
- Short iterations are effective as the planning game for release planning and iteration planning.

- **Object-Oriented Analysis And Design:**
- Object-oriented (O-O) analysis and design is an approach that is intended to facilitate the development of systems that must change rapidly in response to dynamic business environments. Chapter 10 helps you understand what object-oriented systems analysis and design is, how it differs from the structured approach of the SDLC, and when it may be appropriate to use an object-oriented approach.
- Object-oriented techniques are thought to work well in situations in which complicated information systems are undergoing continuous maintenance, adaptation, and redesign. Object-oriented approaches use the industry standard for modeling object-oriented systems, called the unified modeling language (UML), to break down a system into a use case model.
- Object-oriented programming differs from traditional procedural programming by examining objects that are part of a system. Each object is a computer representation of some actual thing or event. Objects may be customers, items, orders, and so on. Objects are represented by and grouped into classes that are optimal for reuse and maintainability. A class defines the set of shared attributes and behaviors found in each object in the class.

- The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.
- Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.
- What is RAD?
- Rapid application development is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.
- In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.
- RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.
- The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable

- **Object-Oriented Analysis And Design:**
- The object-oriented approach combines data and processes (**called methods**) into single entities called **objects. Objects usually** correspond to the real things an information system deals with, such as customers, suppliers, contracts, and rental agreements.
- Putting data and processes together in one place recognizes the fact that there are a limited number of operations for any given data structure, and the object-oriented approach makes sense even though typical systems development keeps data and processes independent of each other.
- The goal of OOAD is to make systems elements more reusable, thus improving system quality and the productivity of systems analysis and design.

- Another key idea behind object orientation is **inheritance.** Inheritance allows the creation of new classes that share some of the characteristics of existing classes.
- For example, from a class of objects called "person," you can use inheritance to define another class of objects called "customer."
- Objects of the class "customer" would share certain characteristics with objects of the class "person": They would both have names, addresses, phone numbers, and so on. Because "person" is the more general class and "customer" is more specific, every customer is a person but not every person is a customer.
- The object-oriented approach to systems development shares the **iterative development approach of the Agile Methodologies.**

- One of the most popular realizations of the iterative approach for object-oriented development is the **Rational Unified Process (RUP),** which is based on an **iterative**, **incremental** approach to systems development. RUP has four phases: **inception**, **elaboration**, **construction**, and **transition** (see Figure 1-11).
- In the **inception** phase, analysts define the scope, determine the feasibility of the project, understand user requirements, and prepare a software development plan. In the **elaboration** phase, analysts detail user requirements and develop a baseline architecture. Analysis and design activities constitute the bulk of the elaboration phase. In the **construction phase**, the software is actually coded, tested, and documented.
- In the **transition** phase, the system is deployed, and the users are trained and supported. As is evident from Figure 1-11, the construction phase is generally the longest and the most resource intensive. The elaboration phase is also long, but less resource intensive. The transition phase is resource intensive but short. The inception phase is short and the least resource intensive. The areas of the rectangles in Figure 1-11 provide an estimate of the overall resources allocated to each phase.

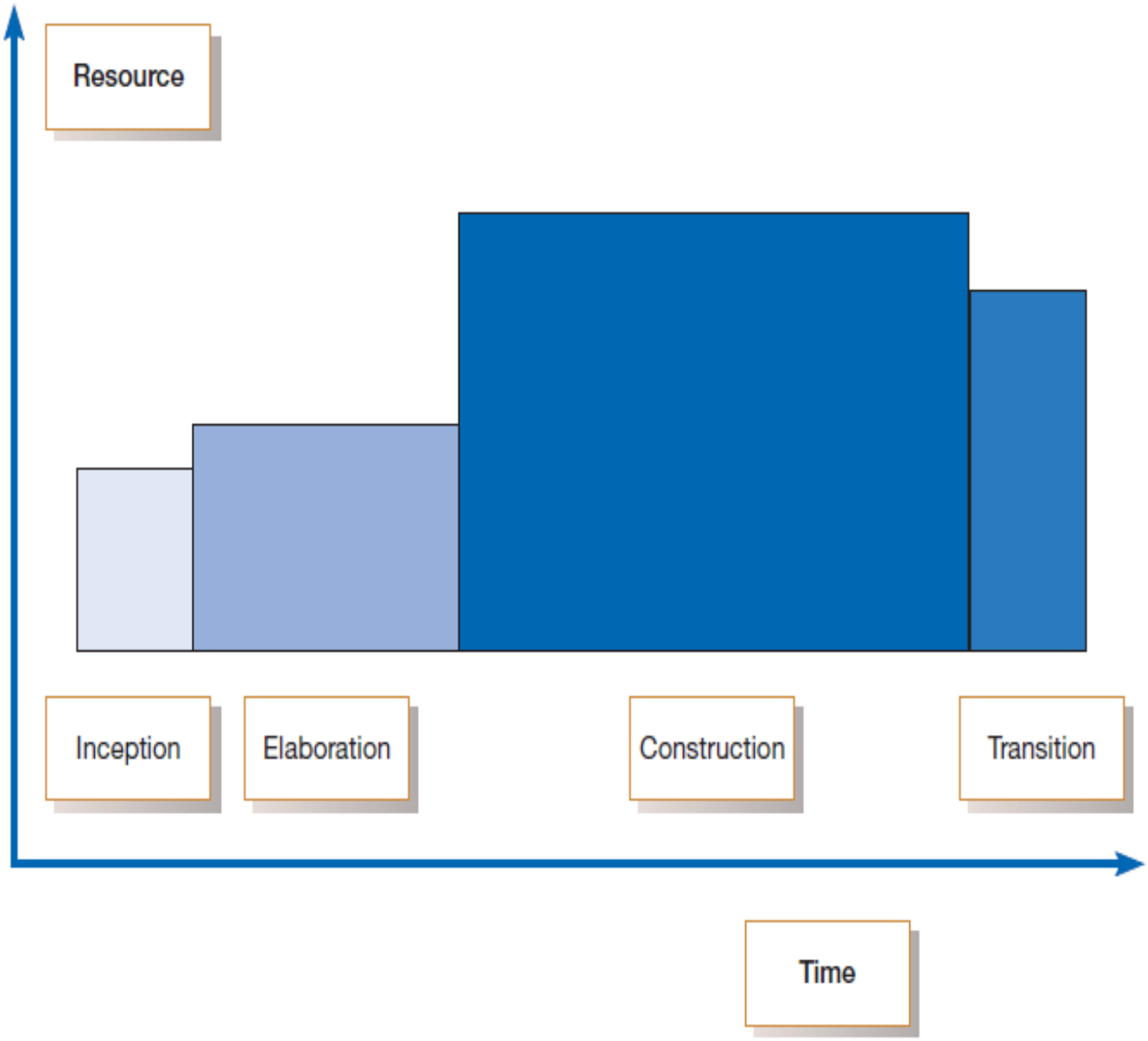
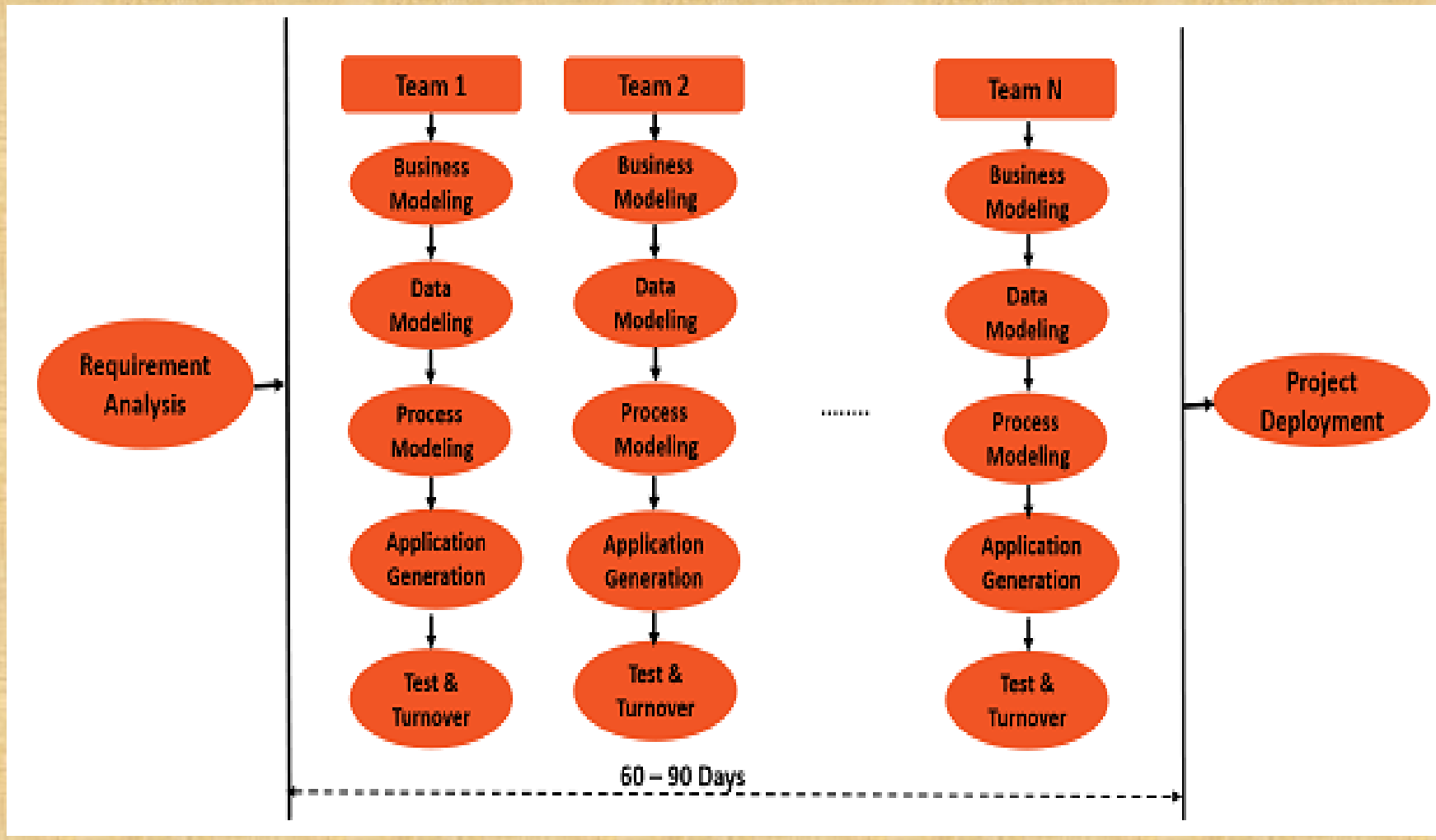


FIGURE 1-11
Phases of OOAD-based development

- Each phase can be further divided into iterations. The software is developed incrementally as a series of iterations. The **inception phase** will generally entail a **single** iteration. The scope and feasibility of the project is determined at this stage. The **elaboration** phase may have one or two iterations and is generally considered the most critical of the four phases (Kruchten, 2000). The elaboration phase is mainly about systems analysis and design, although other activities are also involved. At the end of the **elaboration phase, the architecture of the project should have been developed**. The architecture includes a vision of the product, an executable demonstration of the critical pieces, a detailed glossary and a preliminary user manual, a detailed construction plan, and a revised estimate of planned expenditures.

- RAD


Requirement Analysis
Team 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Test & Turnover
Team 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Test & Turnover
.....
Team N
Business Modeling
Data Modeling
Process Modeling
Application Generation
Test & Turnover
Project Deployment
60 – 90 Days

- Rapid Application Development (RAD) model has the following phases −
- **Requirements Planning phase** − In the requirements planning phase, aworkshop needs to be conducted to discuss business problems in a structured manner.
- **User Description phase** − In the User Description phase, automated tools are used to capture information from users.
- **Construction phase** − In the Construction phase, productivity tools, such as code generators, screen generators, etc. are used inside a time-box, with a “Do until Done” approach.
- **Cut Over phase** − In the Cut over phase, installation of the system, user acceptance testing and user training are performed.

- **Rapid Application Development Model – Strengths**
- The advantages or strengths of the Rapid Application Development model are as follows −
- Reduced cycle time and improved productivity with fewer team members would mean lower costs.
- Customer's involvement throughout the complete cycle minimizes the risk of not achieving customer satisfaction and business value.
- Focus moves to the code in a what-you-see-is-what-you-get mode (WYSIWYG). This brings clarity on what is being built is the right thing.
- Uses modelling concepts to capture information about business, data, and processes

- **Rapid Application Development Model – Weaknesses**
- The disadvantages or strengths of Rapid Application Development model are as follows −
- Accelerated development process must give quick responses to the user.
- Risk of never achieving closure.
- Hard to use with legacy systems.
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

- RAD model is Rapid Application Development model. It is a type of **incremental model**. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.
- **Advantages of the RAD model:**
- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of **integration issues**.
- **Disadvantages of RAD model:**
- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

- **When to use RAD model:**
- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD **SDLC model** should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

- **Service-Oriented Architecture:** It is modern and new concept of software development . It make individual SOA services are unassociated or loosely coupled to another. Each service executes one action. Each service can be used in other application within the organization or even in other organizations.
- We can say that service-oriented architecture is simply a group of services that can be called upon to provide specific functions. Rather than including calls to other services, a service can use certain defined protocols so that it can communicate with other services.

- **Service-Oriented Architecture**
- Advantages
- SOA allows reuse the service of an existing system alternately building the new system.
- It allows plugging in new services or upgrading existing services to place the new business requirements.
- It can enhance the performance, functionality of a service and easily makes the system upgrade.
- SOA has capability to adjust or modify the different external environments and large applications can be managed easily.
- The companies can develop applications without replacing the existing applications.
- It provides reliable applications in which you can test and debug the independent services easily as compared to large number of code.
- Disadvantages
- SOA requires high investment cost (means large investment on technology, development and human resource).
- There is greater overhead when a service interacts with another service which increases the response time and machine load while validating the input parameters.
- SOA is not suitable for GUI (graphical user interface) applications which will become more complex when the SOA requires the heavy data exchange.

- **The Origins Of Software**
- **Introduction**: If you wanted to write application software, you did it in-house, and you wrote the software from scratch. Today there are many different sources of software and firms that produce software, rather than in the information systems department of a corporation. But for those of you who do go on to work in a corporate information systems department, the focus is no longer exclusively on in-house development.
- **SYSTEMS ACQUISITION:** Internal corporate information systems departments now spend a smaller and smaller proportion of their time and effort on developing systems from scratch. Companies continue to spend relatively little time and money on traditional software development and maintenance. Instead, they invest in packaged software, open-source software, and outsourced services.

- **Outsourcing:** If one organization develops or runs a computer application for another organization, that practice is called **outsourcing. Outsourcing includes a spectrum** of working arrangements. At one extreme is having a firm develop and run your application on its computers—all you do is supply input and take output. A common example of such an arrangement is a company that runs payroll applications for clients so that clients do not have to develop an independent in-house payroll system. Instead, they simply provide employee payroll information to the company, and, for a fee, the company returns completed paychecks, payroll accounting reports, and tax and other statements for employees. For many organizations, payroll is a very cost-effective operation when outsourced in this way.

- Outsourcing is big business. Some organizations outsource the information technology (IT) development of many of their IT functions at a cost of billions of dollars. Most organizations outsource at least some aspect of their information systems activities.
- One example of the extent of outsourcing is **Shell Oil**. In 2008, Shell signed outsourcing contracts with EDS, T-Systems, and AT&T worth $3.2 billion USD.
- Biggest outsourcing companies are **IBM** and **EDS.**
- Some reasons for outsourcing include
  - freeing up internal resources,
  - increasing the revenue potential of the organization,
  - reducing time to market,
  - increasing process efficiencies, and
  - outsourcing noncore activities.

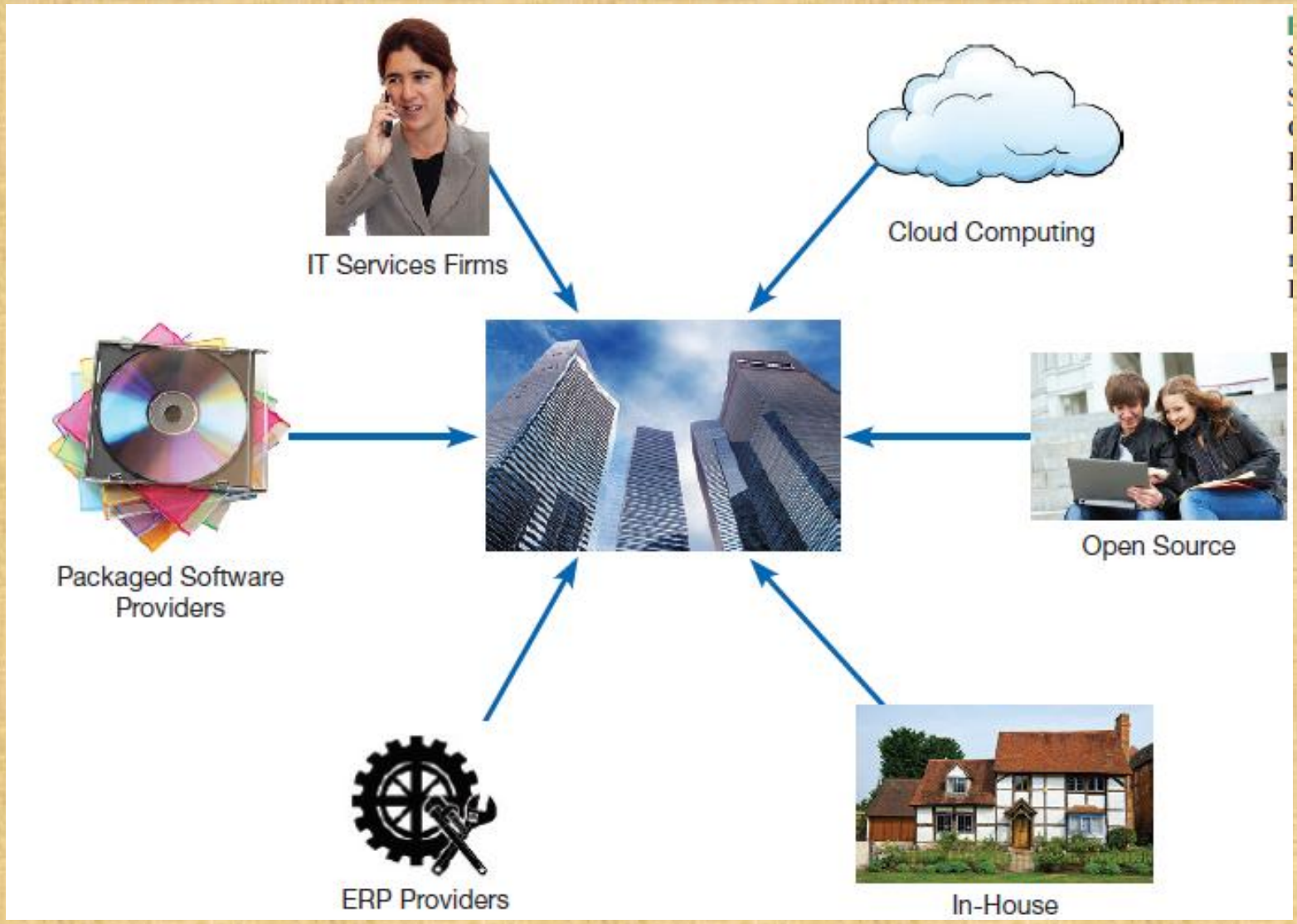


Figure 2-1: Sources of Application Software

- **Sources of Software:**
- We can group the sources of software into **six major** categories: **information technology services firms**, **packaged software producers, enterprise-wide solutions**, **cloud computing vendors**, **open-source software, and in-house developers** (Figure 2-1).
- **Information Technology Services Firms**: If a company needs an information system but does not have the expertise or the personnel to develop the system in-house the company will likely consult an information technology services firm. IT services firms help companies develop custom information systems for internal use, or they develop, host, and run applications for customers, or they provide other services. Note in **Table 2-1** that many of the leading software companies in the world specialize in services, which include custom systems development. These firms employ people with expertise in the development of information systems. Their consultants may also have expertise in a given business area. For example, consultants who work with banks understand financial institutions as well as information systems. Consultants use many of the same methodologies, techniques, and tools that companies use to develop systems in-house.

TABLE 2-1 Leading Software Firms and Their Development Specializations

Specialization	Example Firms or Websites
IT Services	Accenture Computer Sciences Corporation (CSC) IBM HP
Packaged Software Providers	Intuit Microsoft Oracle SAP AG Symantec
Enterprise Software Solutions	Oracle SAP AG
Cloud Computing	Amazon.com Google IBM Microsoft Salesforce.com
Open Source	SourceForge.net

- **Packaged Software Producers:** The growth of the software industry has been phenomenal since its beginnings in the mid-1960s. Some of the largest computer companies in the world are companies that produce software exclusively. A good example is **Microsoft**, probably the best-known software company in the world.
- Almost 87 percent of Microsoft's revenue comes from its software sales, mostly for its Windows operating systems and its personal productivity software, the Microsoft Office Suite. Also listed in Table 2-1, Oracle is exclusively a software company known primarily for its database software, but Oracle also makes enterprise systems.
- Another company on the list, SAP, is also a software-focused company that develops enterprise-wide system solutions.

- Software companies develop what are sometimes called prepackaged or off-the-shelf systems. Microsoft's Word (Figure 2-2) and Intuit's Quicken, QuickPay, and QuickBooks are popular examples of such software.
- Software companies develop software to run on many different computer platforms, from microcomputers to large mainframes.
- The companies range in size from just a few people to thousands of employees.
- Software companies consult with system users after the initial software design has been completed and an early version of the system has been built. The systems are then tested in actual organizations to determine whether there are any problems

  or if any improvements can be made. Until testing is completed, the system is not offered for sale to the public.

- Some off-the-shelf software systems cannot be modified to meet the specific, individual needs of a particular organization. Such application systems are sometimes called **turnkey systems**. *The producer of a turnkey system will make changes* to the software only when a substantial number of users ask for a specific change.
- However, other off-the-shelf application software can be modified or extended, by the producer or by the user, to more closely fit the needs of the organization. Even though many organizations perform similar functions, **no two organizations do the same thing in quite the same way**. A turnkey system may be good enough for a certain level of performance, but it will never perfectly match the way a given organization does business. A reasonable estimate is that off-the-shelf software can at **best meet 70 percent** of an organization's needs. Thus, even in the **best case, 30 percent** of the software system does not match the organization's specifications.

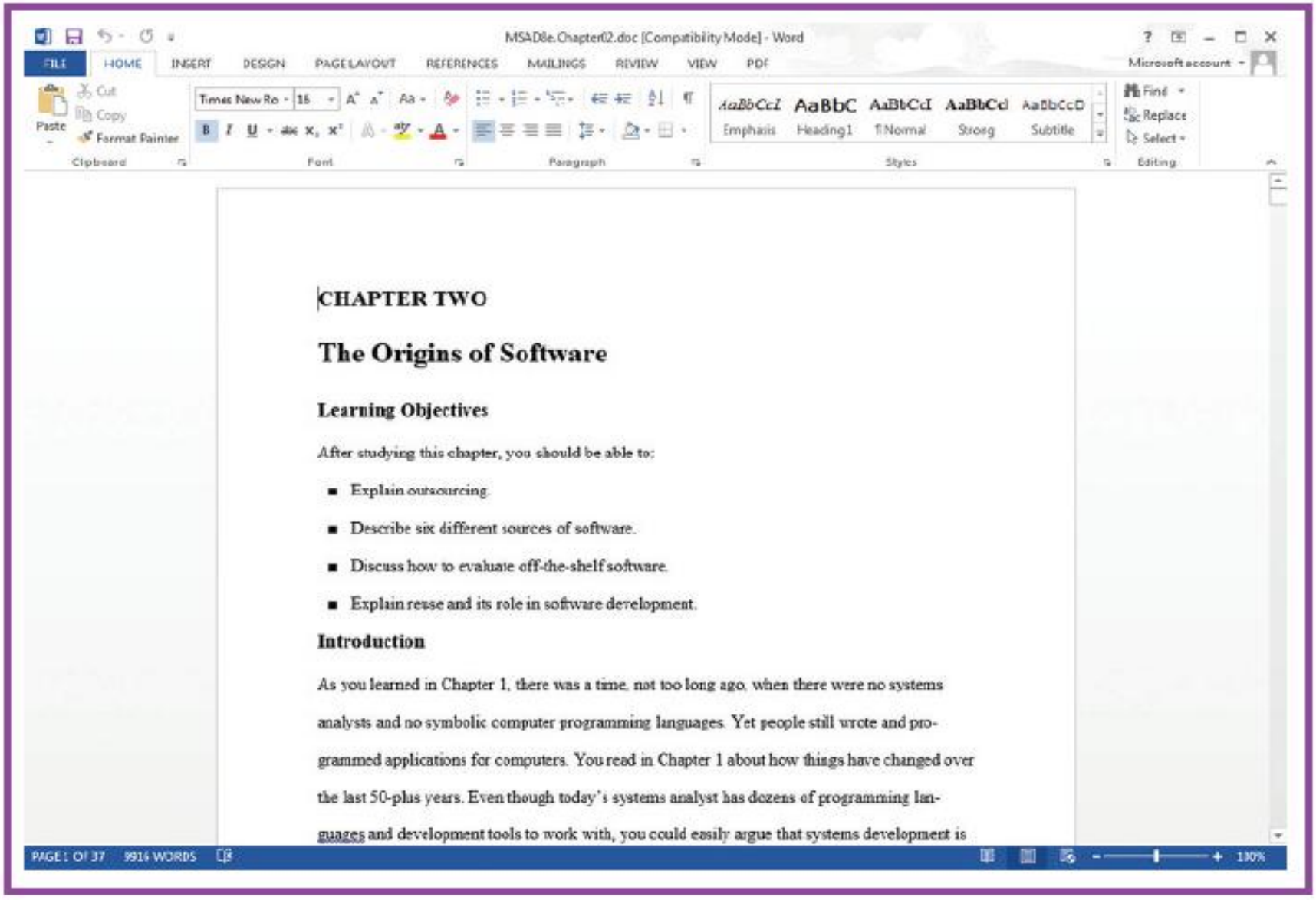


Figure 2-2

A document created in Microsoft's Word

- **Enterprise Solutions Software**
- Many firms have chosen complete software solutions, called enterprise solutions *or* **enterprise resource planning (ERP) systems, to support their operations and business processes. These ERP** software solutions consist of a series of integrated modules. Each module supports an individual, traditional business function, such as accounting, distribution, manufacturing, or human resources. The difference between the modules and traditional approaches is that the **modules are integrated to focus on business processes** rather than on **business functional areas**. For example, a series of modules will support the entire order entry process, from receiving an order, to adjusting inventory, to shipping to billing, to after-the-sale service. The traditional approach would use different systems in different functional areas of the business, such as a billing system in accounting and an inventory system in the warehouse. Using enterprise software solutions, a firm can integrate all parts of a business process in a **unified information system**. All aspects of a single transaction occur seamlessly **within a single information system**, rather than as a series of disjointed, separate systems focused on business functional areas.

- The benefits of the enterprise solutions approach include a single **repository** of data for all aspects of a business process and the flexibility of the modules. A single repository ensures **more consistent and accurate data, as well as less maintenance**.
- The modules are flexible because additional modules can be added as needed once the basic system is in place. Added modules are immediately integrated into the existing system. However, there are **disadvantages** to enterprise solutions software.
- The systems are very complex, so implementation can take a long time to complete. Organizations typically do not have the necessary expertise in-house to implement the systems, so they must rely on consultants or employees of the software vendor, which can be very expensive. In some cases, organizations must change how they do business in order to benefit from a migration to enterprise solutions.

- **Cloud Computing** : Another method for organizations to obtain applications is to rent them or license them from third-party providers who run the applications at remote sites. Users have access to the applications through the Internet or through virtual private networks. The application provider buys, installs, maintains, and upgrades the applications. Users pay on a per-use basis or they license the software,

  typically month to month. Although this practice has been known by many different names over the years, today it is called **cloud computing. Cloud computing refers to** the provision of applications over the Internet, where customers do not have to invest in the **hardware** and **software resources** needed to run and maintain the applications.

  You may have seen the Internet referred to as a cloud in other contexts, which comes from how the Internet is depicted on computer network diagrams. A well-known example of cloud computing is **Google Apps**, where users can **share** and **create documents, spreadsheets, and presentations** (Figure 2-4). Another well-known example is **Salesforce.com**, which provides customer relationship management software online. Cloud computing encompasses many areas of technology, including software as a service (often referred to as *SaaS),* which includes Salesforce.com, and hardware as a service, which includes Amazon Web Services and allows companies to order server capacity and storage on demand.

- Taking the cloud computing route has its advantages. The top three reasons for choosing to go with cloud computing, all of which result in benefits for the company, are
- **(1) freeing internal IT staff**,
- **(2) gaining access to applications faster than via internal development**,
- **(3) achieving lower cost access to corporate-quality applications**. Especially appealing is the ability to gain access to large and complex systems without having to go through the expensive and time-consuming process of implementing the systems themselves in-house.
- Getting your computing through a cloud also makes it easier to walk away from an unsatisfactory systems solution. Other reasons include cost effectiveness, speed to market, and better performance (Moyle & Kelley, 2011).

- **IT managers do have some concerns about cloud computing**, however. The primary concern is over **security**. Concerns over security are based on storing company data on machines one does not own and that others can access. In fact, the top two reasons for not using cloud services are concerns about **unauthorized access to proprietary information** and **unauthorized access to customer information** (Moyle & Kelley, 2011). Another concern is **reliability**. Some warn that the cloud is actually a network of networks, it is vulnerable to unexpected risks due to its **complexity** (kfc, 2012).

- **Open-Source Software**
- Open-source software is unlike the other types of software you have read about so far. Open-source software is different because it is freely available, not just the final product but the source code itself. It is also different because it is developed by a community of interested people instead of by employees of a particular company. Open-source software performs the same functions as commercial software, such as operating systems, e-mail, database systems, web browsers, and so on. Some of the most well-known and popular open-source software names are **Linux**, an operating system; **mySQL**, a database system; and **Firefox**, a web browser. Open source also applies to software components and objects. Open source is developed and maintained by communities of people, and sometimes these communities can be very large.

- If the software is free, you might wonder how anybody makes any money by developing open-source software. Companies and individuals can make money with open source in two primary ways:
- (1) by providing maintenance and other services or
- (2) by providing one version of the software free and selling a more fully featured version.
- Some open-source solutions have more of an impact on the software industry than others. Linux, for example, has been very successful in the server software market, where it is estimated to have as much as 36 percent of the market share (W3Techs, 2015). In the desktop operating systems market, Linux has about 1 percent market share. Other open-source software products, such as mySQL, have also been successful, and open source's share of the software industry seems destined to continue to grow.

- **In-House Development:** We have talked about several different types of external organizations that serve as sources of software, but in-house development remains an option. In-house development has become a progressively smaller piece of all systems development work that takes place in and for organizations. As you read earlier in this chapter, internal corporate information systems departments now spend a smaller and smaller proportion of their time and effort on developing systems from scratch. In-house development can lead to a larger maintenance burden than other development methods, such as **packaged applications**. A study by **Banker, Davis, and Slaughter** found that using a code generator as the basis for in-house development was related to an increase in maintenance hours, whereas using packaged applications was associated with a decrease in maintenance effort.

- Of course, in-house development need not entail (to make something necessary) development of all of the software that will constitute the total system. **Hybrid** solutions involving some purchased and some in-house software components are common. If you choose to acquire software from outside sources, this choice is made at the end of the analysis phase.
- The choice between a package and an external supplier will be

  determined by your needs, not by what the supplier has to sell. As we will discuss, the results of your analysis study will define the type of product you want to buy and will make working with an external supplier much easier, more productive, and worthwhile. Table 2-2 compares the six different software sources discussed in this section.

TABLE 2-2 Comparison of Six Different Sources of Software Components

Producers	When to Go to This Type of Organization for Software	Internal Staffing Requirements
IT services firms	When task requires custom support and system can't be built internally or system needs to be sourced	Internal staff may be needed, depending on application
Packaged software producers	When supported task is generic	Some IS and user staff to define requirements and evaluate packages
Enterprise-wide solutions vendors	For complete systems that cross functional boundaries	Some internal staff necessary but mostly need consultants
Cloud computing	For instant access to an application; when supported task is generic	Few; frees up staff for other IT work
Open-source software	When supported task is generic but cost is an issue	Some IS and user staff to define requirements and evaluate packages
In-house developers	When resources and staff are available and system must be built from scratch	Internal staff necessary though staff size may vary

- REUSE
- Reuse is the use of previously written software resources in new applications. Because so many bits and pieces of applications are relatively generic across applications, it seems intuitive that great savings can be achieved in many areas if those generic bits and pieces do not have to be written a new each time they are needed. Reuse should increase programmer productivity because being able to use existing software for some functions means they can perform more work in the same amount of time. Reuse should also decrease development time, minimizing schedule overruns. Because existing pieces of software have already been tested, reusing them should also result in higher-quality software with lower defect rates, decreasing maintenance costs.
- **Advantages:** Less effort, Time-saving, Reduce cost, Increase software productivity, Utilize fewer resources, Leeds to a better quality software.

- # Managing the information system project
- **Introduction**
- In this chapter, we focus on the systems analyst's role as project manager of an information systems project. Throughout the SDLC, the project manager is responsible for initiating, planning, executing, and closing down the systems development project. Project management is arguably the most important aspect of an information systems development project. Effective project management helps to ensure that systems development projects meet customer expectations and are delivered within budget and time constraints.

- Today, there is a shift in the types of projects most firms are undertaking, which makes project management much more difficult and even more critical to project success (Fuller et al., 2008; Schiff, 2014a). For example, **in the past, organizations focused much of their development on very large, custom-designed, stand-alone applications**. Today, much of the systems development effort in organizations focuses on implementing packaged software such as enterprise resource planning (ERP) and data warehousing systems. Existing legacy applications are also being modified so that business-to-business transactions can occur seamlessly over the Internet. New web-based interfaces are being added to existing legacy systems so that a broader range of users, often distributed globally, can access corporate information and systems. Additionally, software developed by global outsourcing partners that must be integrated into an organization's existing portfolio of applications is now common practice (Overby, 2013). Working with vendors to supply applications, with customers or suppliers to integrate systems, or with a broader and more diverse user community requires that project managers be highly skilled. Consequently, it is important that you gain an understanding of the project management process; this will become a critical skill for your future success.

- The **project management process** involves four phases:

  1. **Initiating the project**
  2. **Planning the project**
  3. **Executing the project**
  4. **Closing down the project**

- **Initiating a project:** During project initiation, the project manager performs several activities to assess the size, scope, and complexity of the project and to establish procedures to support subsequent activities.

  The types of activities that will perform when initiating a project are summarized below:

- 1. **Establishing the project initiation team:** This activity involves organizing project team members to assist in accomplishing the project initiation activities. (For example, during the Purchasing Fulfillment System project at PVF, Chris Martin was assigned to support the Purchasing department. It is a PVF policy that all initiation teams consist of at least one user representative, in this case Juanita Lopez, and one member of the information systems (IS) development group. Therefore, the project initiation team consisted of Chris and Juanita; Chris was the project manager.)

- **2. Establishing a relationship with the customer**
- A thorough understanding of your customer builds stronger partnerships and higher levels of trust.

  (At PVF, management has tried to foster strong working relationships between business units (like Purchasing) and the IS development group by assigning a specific individual to work as a liaison between both groups. Because Chris had been assigned to the Purchasing unit for some time, he was already aware of some of the problems with the existing purchasing systems. PVF's policy of assigning specific individuals to each business unit helped to ensure that both Chris and Juanita were comfortable working together prior to establishing relationships with customers.)
- 3. **Establishing the project initiation plan:** This step defines the activities required to organize the initiation team while it is working to define the goals and scope of the project.

- **4. Establishing management procedures**
- Successful projects require the development of effective management procedures**.**
- **5. Establishing the project management environment and project workbook**. The focus of this activity is to collect and organize the tools that you will use while managing the project and to construct the project workbook. Diagrams, charts, and system descriptions provide much of the project workbook contents. Thus, the project workbook serves as a repository for all project correspondence, inputs, outputs, deliverables, procedures, and standards established by the project team.
- **6. Developing the project charter**. The project charter is a short (typically one page), high-level document prepared for the customer that describes what the project will deliver and outlines many of the key elements of the project.

- **Planning the project**
- **Research has found a positive relationship between effective project planning and positive project outcomes**. Project planning involves defining clear, discrete activities and the work needed to complete each activity within a single project. It often requires you to make numerous assumptions about the availability of resources such as hardware, software, and personnel. It is much easier to plan nearer-term activities than those occurring in the future. (In actual fact, you often have to construct longer-term plans that are more general in scope and nearer-term plans that are more detailed. The repetitive nature of the project management process requires that plans be constantly monitored throughout the project and periodically updated (usually after each phase), based upon the most recent information.)

Project Planning

1. Describing Project Scope, Alternatives, and Feasibility
2. Dividing the Project into Manageable Tasks
3. Estimating Resources and Creating a Resource Plan
4. Developing a Preliminary Schedule
5. Developing a Communication Plan
6. Determining Project Standards and Procedures
7. Identifying and Assessing Risk
8. Creating a Preliminary Budget
9. Developing a Project Scope Statement
10. Setting a Baseline Project Plan

- **1. Describing project scope, alternatives, and feasibility**
- The purpose of this activity is to understand the content and complexity of the project. Within PVF's systems development methodology, one of the first meetings must focus on defining a project's scope. Although project scope information was not included in the SSR developed by Chris and Juanita, it was important that both shared the same vision for the project before moving too far along. During this activity, you should reach agreement on the following questions:
  - What problem or opportunity does the project address?
  - What are the quantifiable results to be achieved?
  - What needs to be done?
  - How will success be measured?
  - How will we know when we are finished?

  After defining the scope of the project, your next objective is to identify and document general alternative solutions for the current business problem or opportunity. You must then assess (to judge or decide value or importance) the feasibility of each alternative solution and choose which to consider during subsequent SDLC phases.

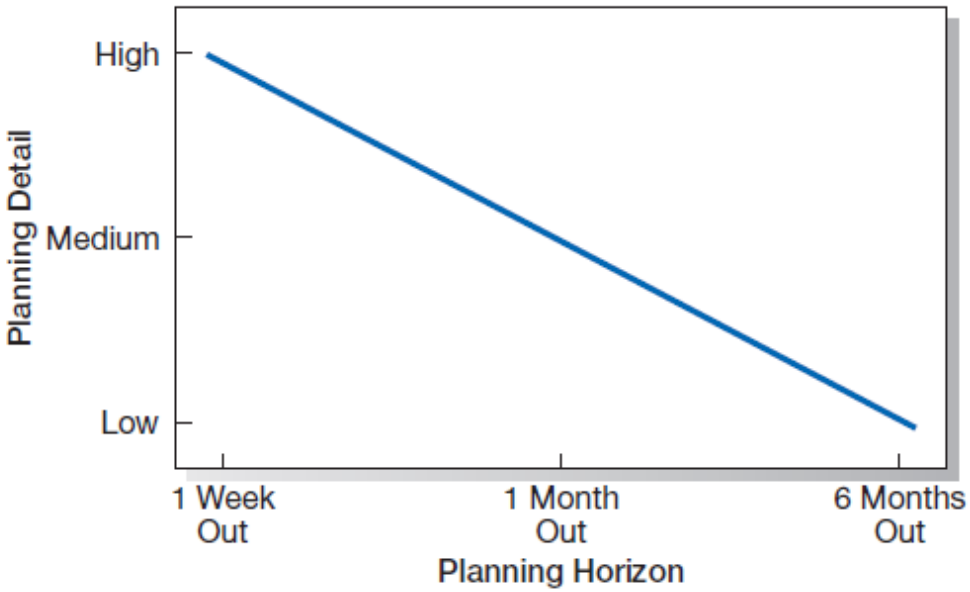


FIGURE 3-8
Level of project planning detail should be high in the short term, with less detail as time goes on

At the start of project, there is no idea of doing work, every thing is unknown so detailed plan is needed to do every step. But after some time ,that is week or month, people gets idea of project scheduling and can do some steps on the basis of previous step or his/her own creativity.

- **2. Dividing the project into manageable tasks**.

  This is a critical activity during the project planning process. Here, you must divide the entire project into manageable tasks and then logically order them to ensure a smooth evolution between tasks.

  For example, suppose that you are working on a new development project and need to collect system requirements by interviewing users of the new system and reviewing reports they currently use to do their job. A work breakdown for these activities is represented in a Gantt chart in Figure 3-10. A **Gantt chart is a** graphical representation of a project that shows each task as a horizontal bar whose length is proportional to its time for completion. Different colors, shades, or shapes can be used to highlight each kind of task. For example, those activities on the critical path (defined later) may be in red and a summary task could have a special bar. Note that the **black horizontal bars**—rows 1, 2, and 6 in Figure 3-10—**represent summary tasks**. Planned versus actual times or progress for an activity can be compared by parallel bars of different colors, shades, or shapes. Gantt charts do not (typically) show how tasks must be ordered (precedence), but simply show **when an activity should begin and end**. In Figure 3-10, the task duration is shown in the second column by days, "d,".

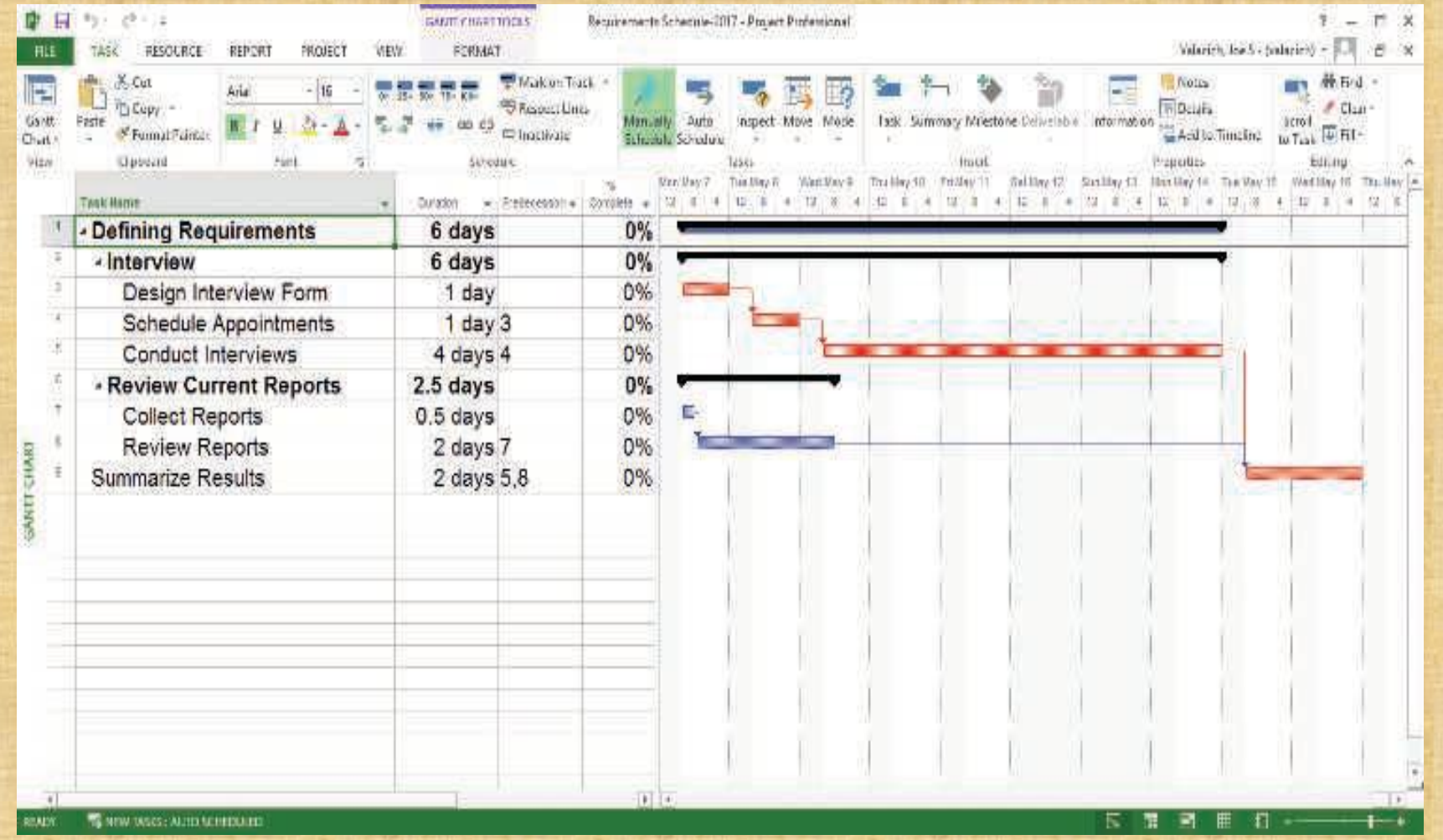


FIGURE 3-10

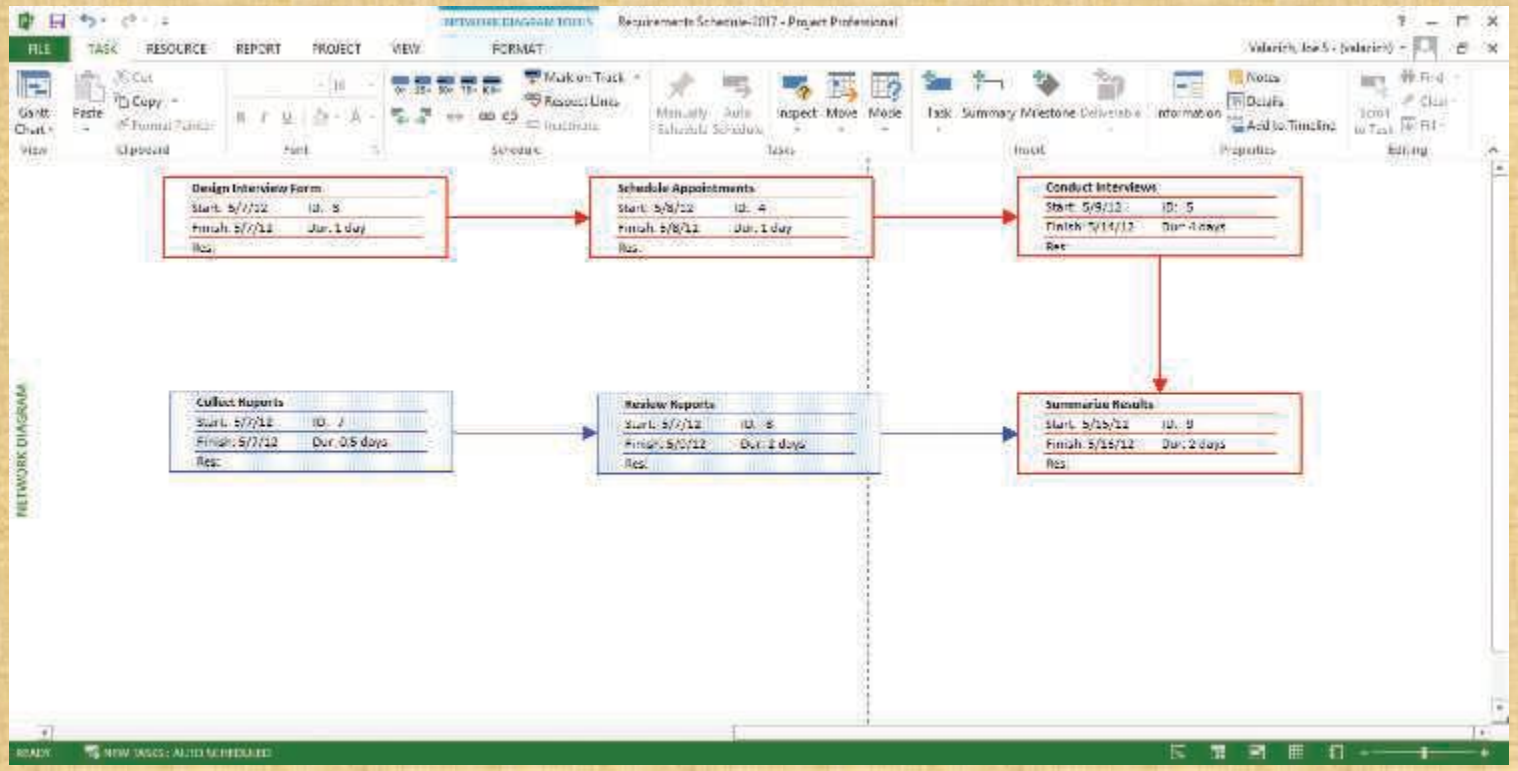
Gantt chart showing project tasks, duration times for those tasks, and predecessors

- **3. Estimating resources and creating a resource plan**

  The goal of this activity is to estimate resource requirements for each project activity and to use this information to create a project resource plan. The resource plan helps assemble and deploy resources in the most effective manner.

  For example, you would not want to bring additional programmers onto the project at a rate faster than you could prepare work for them. Project managers use a variety of tools to assist in making estimates of project size and costs. The most widely used method is called COCOMO (constructive cost model**),** COCOMO predict human resource requirements for basic, intermediate, and very complex systems.

- **4. Developing a preliminary schedule**
- During this activity, you use the information on tasks and resource availability to assign time estimates to each activity in the work breakdown structure. These time estimates will enable you to create target starting and ending dates for the project. Target dates can be revisited and modified until a schedule is produced that is acceptable to the customer. Determining an acceptable schedule may require that you find additional or different resources or that the scope of the project be changed. The schedule may be represented as a Gantt chart or as a network diagram.


Requirements Schedule-2017 - Project Professional
FILE
TASK
RESOURCE
REPORT
PROJECT
VIEW
FORMAT
Design Interview Form
Start: 5/7/12 Id: 3
Finish: 5/7/12 Dur: 1 day
Res:
Schedule Appointments
Start: 5/8/12 Id: 4
Finish: 5/8/12 Dur: 1 day
Res:
Conduct Interviews
Start: 5/9/12 Id: 5
Finish: 5/14/12 Dur: 4 days
Res:
Collect Reports
Start: 5/7/12 Id: 7
Finish: 5/7/12 Dur: 0.5 days
Res:
Review Reports
Start: 5/7/12 Id: 8
Finish: 5/10/12 Dur: 2 days
Res:
Summarize Results
Start: 5/15/12 Id: 9
Finish: 5/15/12 Dur: 2 days
Res:
NETWORK DIAGRAM
READY
NEW TASKS: AUTO SCHEDULED

- **5. Developing a communication plan**

- The goal of this activity is to outline the communication procedures among management, project team members, and the customer. The communication plan includes when and how written and oral reports will be provided by the team, how team members will coordinate work, what messages will be sent to announce the project to interested parties, and what kinds of information will be shared with vendors and external contractors involved with the project.
- It is important that free and open communication occur among all parties with respect to proprietary information and confidentiality with the customer (Fuller et al., 2008; Kettelhut, 1991; Kirsch, 2000; Vaidya nathan, 2013;Verma, 1996).
- When developing a communication plan, numerous questions must be answered in order to assure that the plan is comprehensive and complete, including the following:

- Who are the stakeholders for this project?
- What information does each stakeholder need?
- When, and at what interval, does this information need to be produced?
- What sources will be used to gather and generate this information?
- Who will collect, store, and verify the accuracy of this information?
- Who will organize and package this information into a document?
- Who will be the contact person for each stakeholder should any questions arise?
- What format will be used to package this information?
- What communication medium will be most effective for delivering this information to the stakeholder?
- Once these questions are answered for each stakeholder, a comprehensive communication plan can be developed. In this plan, a summary of communication documents, work assignments, schedules, and distribution methods will be outlined.

6. Determining project standards and procedures

During this activity, you will specify how various deliverables are produced and tested by you and your project team. For example, the team must decide which tools to use, how the standard SDLC might be modified, which SDLC methods will be used, documentation styles (e.g., type fonts and margins for user manuals), how team members will report the status of their assigned activities, and terminology. Setting project standards and procedures for work acceptance is a way to ensure the development of a high-quality system. Also, it is much easier to train new team members when clear standards are in place. Organizational standards for project management and conduct make the determination of individual project standards easier and the interchange or sharing of personnel among different projects feasible.

7. Identifying and assessing risk

The goal of this activity is to identify sources of project risk and estimate the consequences of those risks (Wideman, 1992). Risks might arise from the use of new technology, prospective users' resistance to change, availability of critical resources, competitive reactions or changes in

regulatory actions due to the construction of a system, or team member inexperience with technology or the business area. You should continually try to identify and assess project risk.

8. Creating a preliminary budget

During this phase, you need to create a preliminary budget that outlines the planned expenses and revenues associated with your project. The project justification will demonstrate that the benefits are worth these costs.

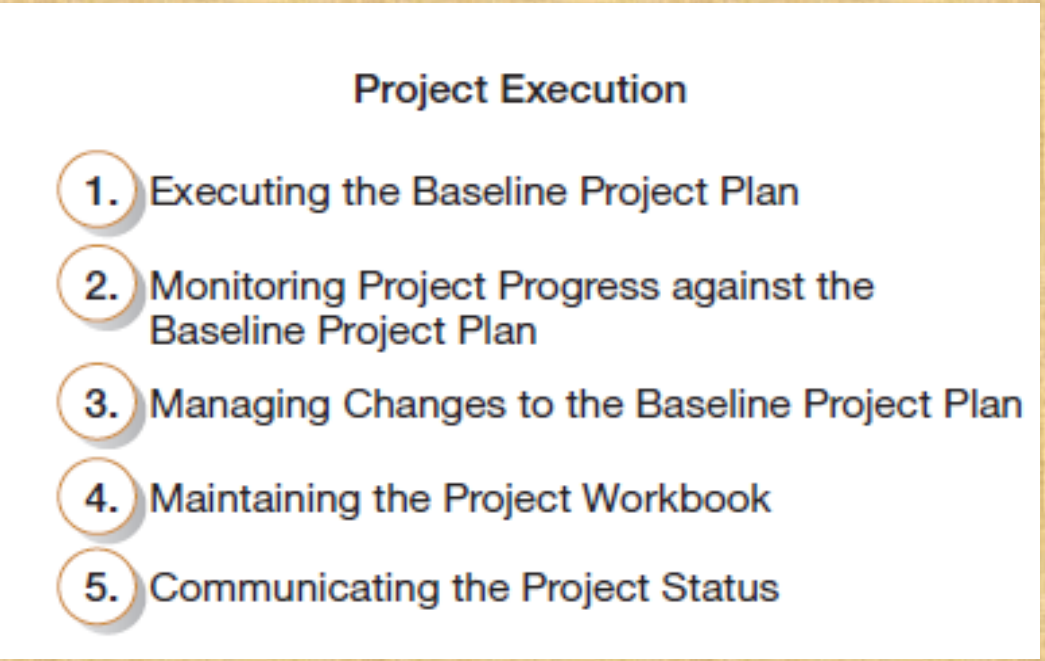
9. Developing a Project Scope Statement

An important activity that occurs near the end of the project planning phase is the development of the Project Scope Statement. Developed primarily for the customer, this document outlines work that will be done and clearly describes what the project will deliver. The Project Scope Statement is useful to make sure that you, the customer, and other project team members have a clear understanding of the intended project size, duration, and outcomes.

10. Setting a Baseline Project Plan

Once all of the prior project planning activities have been completed, you will be able to develop a Baseline Project Plan. This baseline plan provides an estimate of the project's tasks and resource requirements and is used to guide the next project phase—execution. As new information is acquired during project execution, the baseline plan will continue to be updated.

- **Executing the Project:**
- Project execution puts the Baseline Project Plan into action. Within the context of the SDLC, project execution occurs primarily during the analysis, design, and implementation phases.



- **1. Executing the Baseline Project Plan:** This means that you initiate the execution of project activities, acquire and assign resources, orient and train new team members, keep the project on schedule, and ensure the quality of project deliverables. This is a formidable task, but a task made much easier through the use of sound project management techniques.

- **2. Monitoring project progress against the Baseline Project Plan**

  While you execute the Baseline Project Plan, you should monitor your progress. If the project gets ahead of (or behind) schedule, you may have to adjust resources, activities, and budgets. Monitoring project activities can result in modifications to the current plan. Measuring the time and effort expended on each activity will help you improve the accuracy of estimations for future projects. It is possible, with project schedule charts such as Gantt charts, to show progress against a plan, and it is easy with network diagrams to understand the ramifications (**the possible results of an action**)of delays in an activity. Monitoring progress also means that the team leader must evaluate and appraise (**to examine someone or something in order to judge their qualities, success or needs**)each team member, occasionally change work assignments or request changes in personnel, and provide feedback to the employee's supervisor.

- **3. Managing changes to the Baseline Project Plan**

  You will encounter pressure to make changes to the baseline plan.

  Numerous events may initiate a change to the Baseline Project Plan, including the following possibilities:

- A slipped completion date for an activity
- A bungled (to do something wrong, in a careless or stupid way) activity that must be redone
- The identification of a new activity that becomes evident later in the project
- An unforeseen change in personnel due to sickness, resignation, or termination

- **4. Maintaining the project workbook**

  As in all project phases, maintaining complete records of all project events is necessary. The workbook provides the documentation new team members require to assimilate (to take in, fit into, or become similar) project tasks quickly. It explains why design decisions were made and is a primary source of information for producing all project reports.

- **5. Communicating the project status**

  The project manager is responsible for keeping all stakeholders—system developers, managers, and customers—abreast (**describes two or more people who are next to each other and moving in the same direction**) of the project status.

- **Closing Down the project**
- The focus of project closedown is to bring the project to an end. Projects can conclude with a natural or unnatural termination. **A natural** termination occurs when the requirements of the project have been met—the project has been completed and is a success. **An unnatural** termination occurs when the project is stopped before completion (Keil et al., 2000). Several events can cause an unnatural termination of a project. **For example**, it may be learned that the assumption used to guide the project proved to be false, that the performance of the systems or development group was somehow inadequate, or that the requirements are no longer relevant or valid in the customer's business environment. The most likely reasons for the **unnatural termination** of a project relate to running out of time or money, or both.

- **Representing and Scheduling Project Plans**

  A project manager has a wide variety of techniques available for depicting (to represent or show something in a picture or story) and documenting project plans. These planning documents can take the form of graphical or textual reports, although graphical reports have become most popular for depicting project plans. The most commonly used methods are **Gantt charts** and **network diagrams**. Because Gantt charts do not (typically) show how tasks must be ordered (precedence) but simply show when a task should begin and when it should end, they are often more useful for depicting (representing) relatively simple projects or subparts of a larger project, showing the activities of a single worker, or monitoring the progress of activities compared to scheduled completion dates (Figure 3-18). Recall that a network diagram shows the ordering of activities by connecting a task to its predecessor and successor tasks. Sometimes a network diagram is preferable; other times a Gantt chart more easily shows certain aspects of a project. Here are the key differences between these two charts:

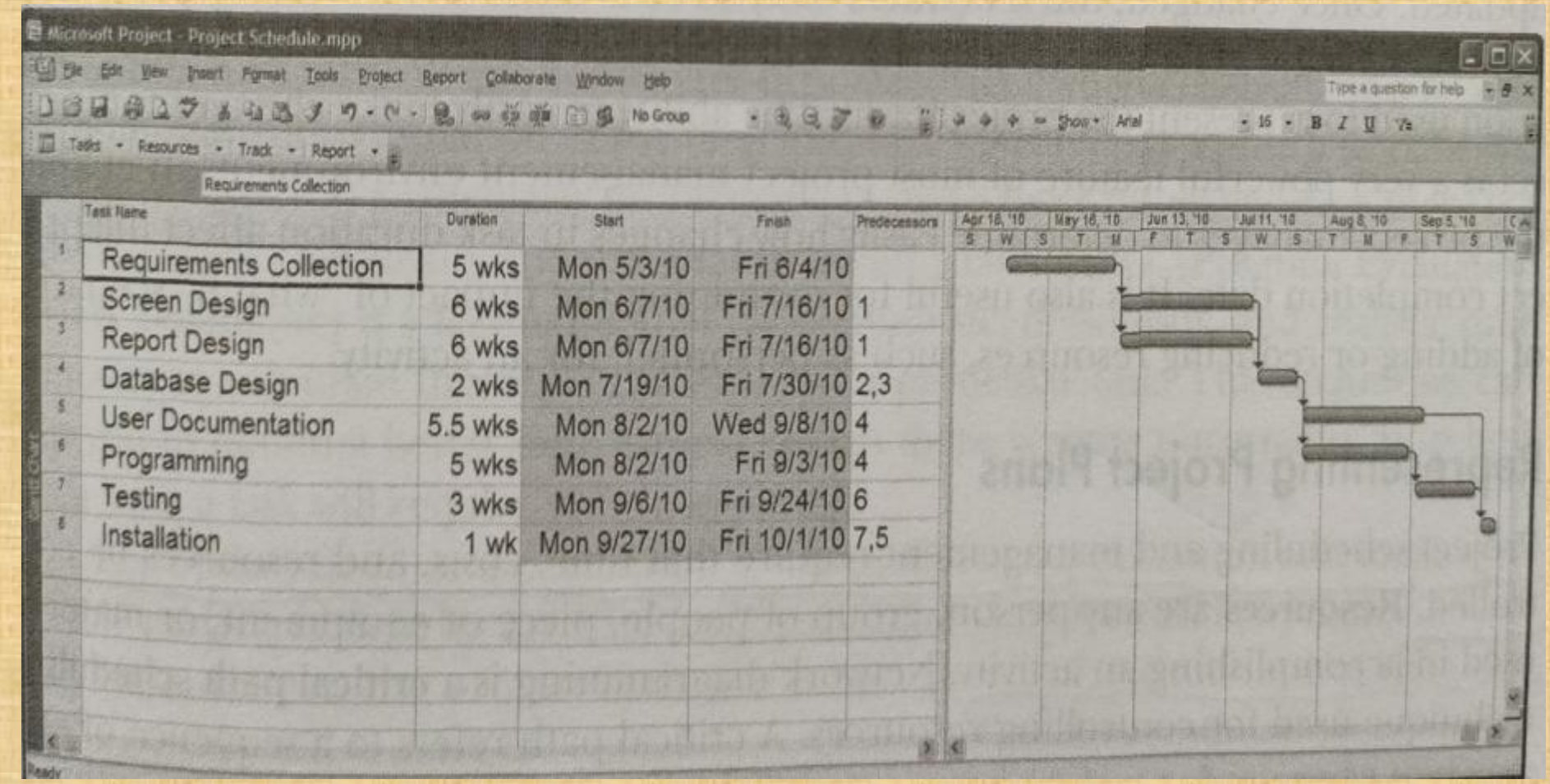
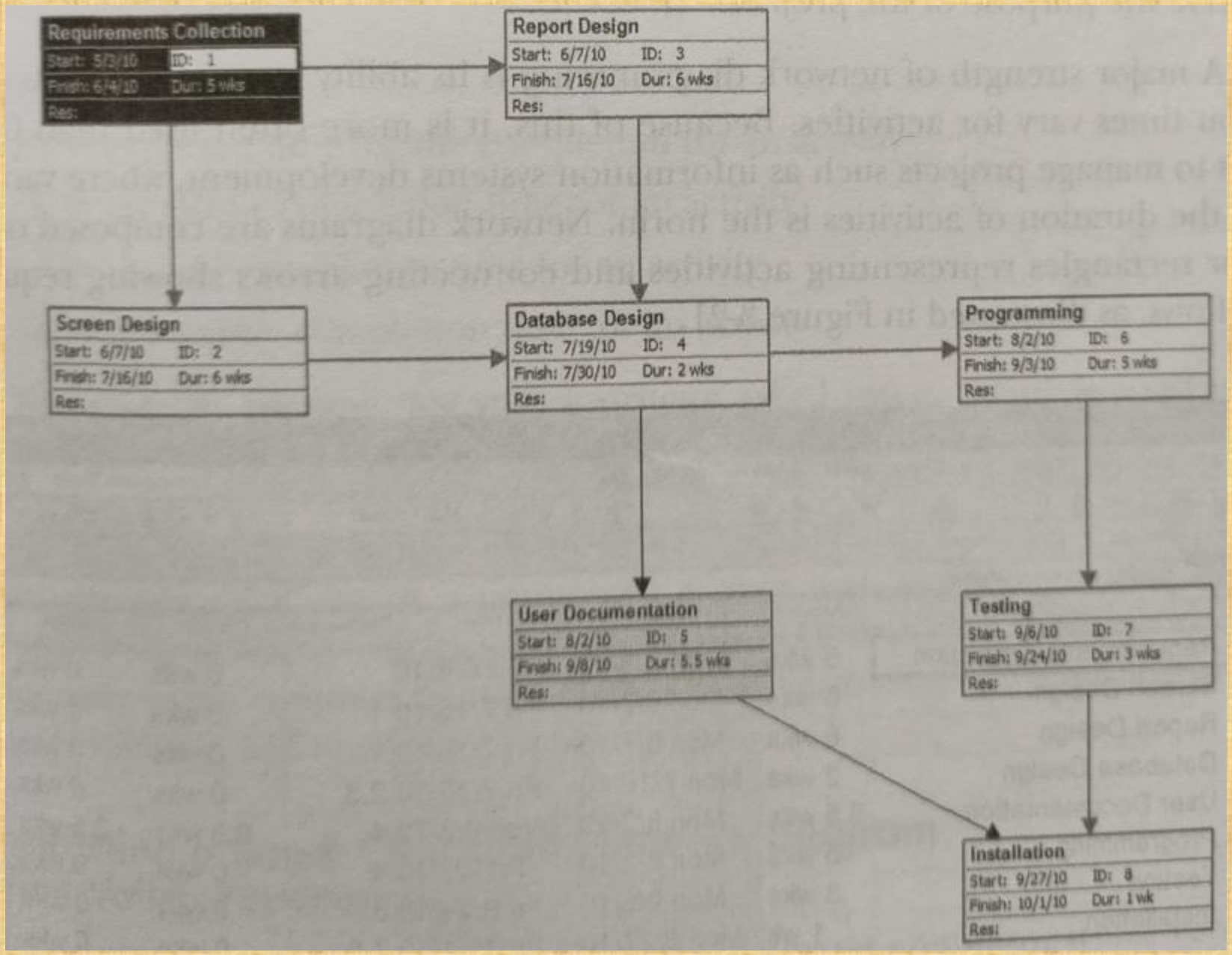


FIGURE 3-18

Graphical diagrams that depict project plans

(a) A Gantt chart

(b) A network diagram


Requirements Collection
Start: 5/3/10 ID: 1
Finish: 6/4/10 Dur: 5 wks
Res:
Report Design
Start: 6/7/10 ID: 3
Finish: 7/16/10 Dur: 6 wks
Res:
Screen Design
Start: 6/7/10 ID: 2
Finish: 7/16/10 Dur: 6 wks
Res:
Database Design
Start: 7/19/10 ID: 4
Finish: 7/30/10 Dur: 2 wks
Res:
Programming
Start: 8/2/10 ID: 6
Finish: 9/3/10 Dur: 5 wks
Res:
User Documentation
Start: 8/2/10 ID: 5
Finish: 9/8/10 Dur: 5.5 wks
Res:
Testing
Start: 9/6/10 ID: 7
Finish: 9/24/10 Dur: 3 wks
Res:
Installation
Start: 9/27/10 ID: 8
Finish: 10/1/10 Dur: 1 wk
Res:

- Gantt charts visually show the duration of tasks, whereas a network diagram visually shows the sequence dependencies between tasks.
- Gantt charts visually show the time overlap of tasks, whereas a network diagram does not show time overlap but does show which tasks could be done in parallel.
- Some forms of Gantt charts can visually show slack time available within an earliest start and latest finish duration. A network diagram shows this by data within activity rectangles.
- Project managers also use textual reports that depict (represent) resource utilization by task, complexity of the project, and cost distributions to control activities. For example, Figure 3-19 shows a screen from Microsoft Project for Windows that summarizes all project activities, their durations in weeks, and their scheduled starting and ending dates. Most project managers use computer-based systems to help develop their graphical and textual reports.

- A project manager will periodically review the status of all ongoing project task activities to assess whether the activities will be completed early, on time, or late. If early or late, the duration of the activity, represented in column 2 of Figure 3-19, can be updated. Once changed, the scheduled start and finish times of all subsequent tasks will also change. Making such a change will also alter a Gantt chart or network diagram used to represent the project tasks. The ability to easily make changes to a project is a very powerful feature of most project management environments. It enables the project manager to determine easily how changes in task duration affect the project completion date. It is also useful for examining the impact of "what if" scenarios of adding or reducing resources, such as personnel, for an activity.

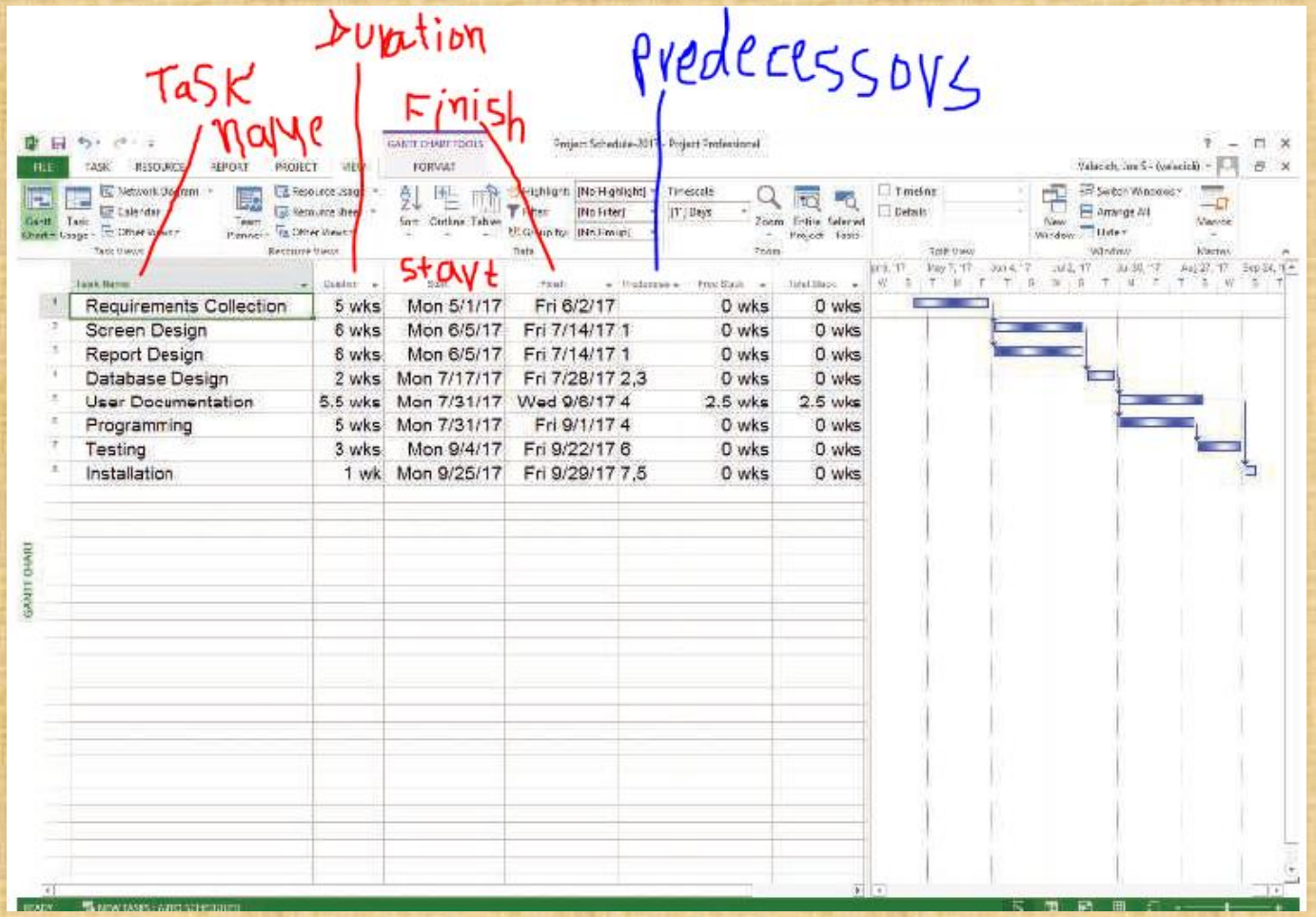

Task name
Duration
Finish
start
Predecessors
Requirements Collection 5 wks Mon 5/1/17 Fri 6/2/17 0 wks 0 wks
Screen Design 6 wks Mon 6/5/17 Fri 7/14/17 1 0 wks 0 wks
Report Design 6 wks Mon 6/5/17 Fri 7/14/17 1 0 wks 0 wks
Database Design 2 wks Mon 7/17/17 Fri 7/28/17 2,3 0 wks 0 wks
User Documentation 5.5 wks Mon 7/31/17 Wed 9/6/17 4 2.5 wks 2.5 wks
Programming 5 wks Mon 7/31/17 Fri 9/1/17 4 0 wks 0 wks
Testing 3 wks Mon 9/4/17 Fri 9/22/17 6 0 wks 0 wks
Installation 1 wk Mon 9/25/17 Fri 9/29/17 7,5 0 wks 0 wks

Figure 3.19

- **Representing Project Plans:** Project scheduling and management require that time, costs, and resources be controlled. **Resources are any person, group of people, piece of equipment, or material** used in accomplishing an activity. Network diagramming is a **critical path scheduling** technique used for controlling resources. **A critical path refers to a sequence of task activities whose order and durations directly affect the completion date of a project**. A network diagram is one of the most widely used and best-known scheduling methods. You would use a network diagram when tasks
  - are well defined and have a clear beginning and end point,
  - can be worked on independently of other tasks,
  - are ordered, and
  - serve the purpose of the project
- A major strength of network diagramming is its ability to represent how completion times vary for activities. Because of this, it is more often used than Gantt charts to manage projects such as information systems development, where variability in the duration of activities is the norm. Network diagrams are composed of circles or rectangles representing activities and connecting arrows showing required work flows, as illustrated in Figure 3-20.

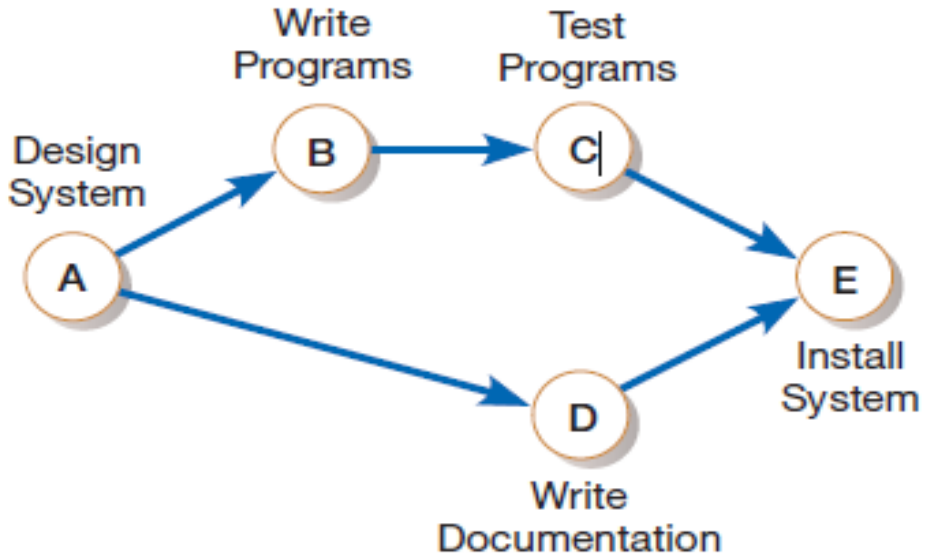


FIGURE 3-20
A network diagram showing activities (represented by circles) and sequence of those activities (represented by arrows)

- **Calculating Expected time durations using PERT:(Program Evaluation Review Technique)**
- One of the most difficult and most error-prone activities when constructing a project schedule is the determination of the time duration for each task within a work breakdown structure. It is particularly problematic to make these estimates when there is a high degree of complexity and uncertainty about a task. **PERT is a technique that uses optimistic, pessimistic, and realistic time** estimates to calculate the expected time for a particular task. This technique can help you to obtain a better time estimate when there is some uncertainty as to how much time a task will require to be completed. The **optimistic** (o) and **pessimistic** (p) times reflect the minimum and maximum possible periods of time for an activity to be completed. The realistic (r) time, or most likely time, reflects the project manager's "best guess" of the amount of time the activity actually will require for completion. Once each of these estimates is made for an activity, an expected time (ET) can be calculated. Because the expected completion time should be closest to the realistic (r) time, it is typically weighted four times more than the optimistic (o) and pessimistic (p) times. Once you add these values together, it must be divided by six to determine the ET. This equation is shown in the following formula:

$$ET = \frac{o + 4r + p}{6}$$

where

- ET = expected time for the completion for an activity
- o = optimistic completion time for an activity
- r = realistic completion time for an activity
- p = pessimistic completion time for an activity
- For example, suppose that your instructor asked you to calculate an expected time for the completion of an upcoming programming assignment. For this assignment, you estimate an optimistic time of two hours, a pessimistic time of eight hours, and a most likely time of six hours. Using PERT, the expected time for completing this assignment is 5.67 hours

- **Using project management software**
- A wide variety of automated project management tools is available to help you manage a development project. New versions of these tools are continuously being developed and released by software vendors. Most of the available tools have a set of common features that include the ability to define and order tasks, assign resources to tasks, and easily modify tasks and resources. Project management tools are available to run on IBM-compatible personal computers, the Macintosh, and larger mainframe and workstation-based systems. These systems vary in the number of task activities supported, the complexity of relationships, system processing and storage requirements, and, of course, cost. For example, numerous shareware project management programs (e.g., OpenProj, Bugzilla, and eGroupWare) can be downloaded from the web (e.g., at www.download.com).