Chapter 1: System Development Environment

A. The System Development Environment

1. System

The word System is derived from Greek word Systema, which means an organized relationship between any set of components to achieve some common cause or objective.

A system is "an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal."

Constraints of a System

A system must have three basic constraints −

- A system must have some **structure and behavior** which is designed to achieve a predefined objective.
- **Interconnectivity** and **interdependence** must exist among the system components.
- The **objectives of the organization** have a **higher priority** than the objectives of its subsystems.

For example, traffic management system, payroll system, automatic library system, human resources information system.

Properties of a System

A system has the following properties −

a. Organization
- Organization implies structure and order.
- It is the arrangement of components that helps to achieve predetermined objectives.

b. Interaction
- It is defined by the manner in which the components operate with each other.
- For example, in an organization, purchasing department must interact with production department and payroll with personnel department.

c. Interdependence
- Interdependence means how the components of a system depend on one another.
- For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is the required by other subsystem as input.

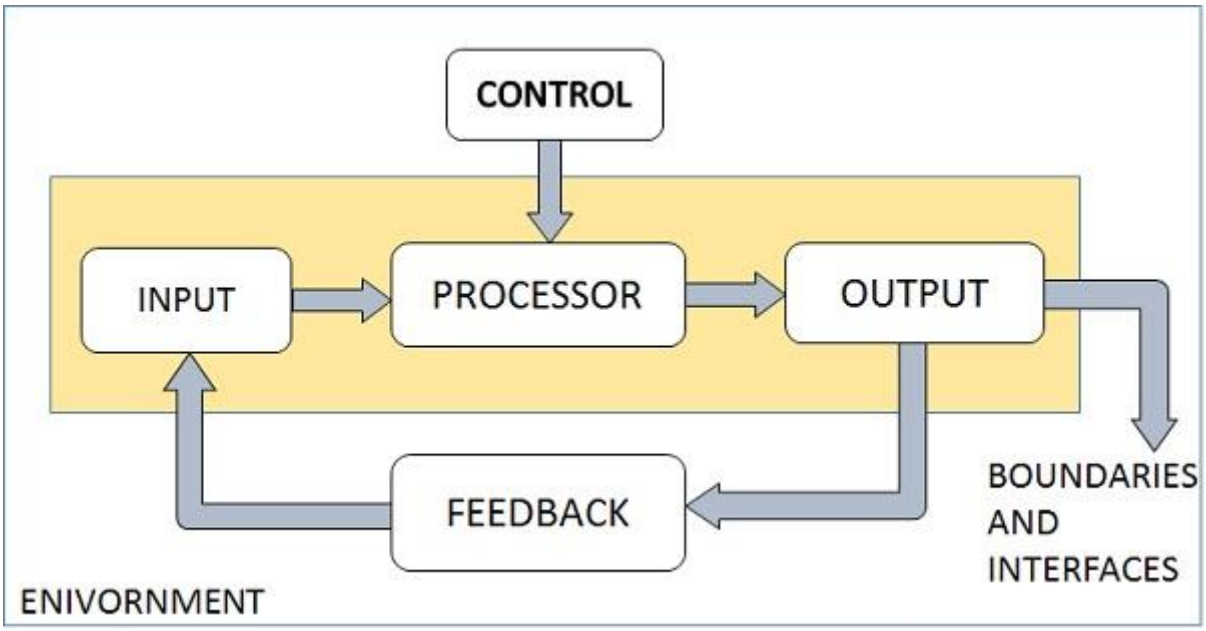
d. Integration
- Integration is concerned with how a system components are connected together.
- It means that the parts of the system work together within the system even if each part performs a unique function.

e. Central Objective

- The objective of system must be central.
- It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.
- The users must know the main objective of a computer application early in the analysis for a successful design and conversion.

Elements of a System

The following diagram shows the elements of a system −



a. Outputs and Inputs

- The main aim of a system is to produce an output which is useful for its user.
- Inputs are the information that enters into the system for processing.
- Output is the outcome of processing.

b. Processor(s)

- The processor is the element of a system that involves the actual transformation of input into output.
- It is the operational component of a system. Processors may modify the input either totally or partially, depending on the output specification.
- As the output specifications change, so does the processing. In some cases, input is also modified to enable the processor for handling the transformation.

c. Control

- The control element guides the system.
- It is the decision–making subsystem that controls the pattern of activities governing input, processing, and output.

- The behavior of a computer System is controlled by the Operating System and software. In order to keep system in balance, what and how much input is needed is determined by Output Specifications.

d. Feedback

- Feedback provides the control in a dynamic system.
- Positive feedback is routine in nature that encourages the performance of the system.
- Negative feedback is informational in nature that provides the controller with information for action.

e. Environment

- The environment is the "super system" within which an organization operates.
- It is the source of external elements that strike on the system.
- It determines how a system must function. For example, vendors and competitors of organization's environment, may provide constraints that affect the actual performance of the business.

f. Boundaries and Interface

- A system should be defined by its boundaries. Boundaries are the limits that identify its components, processes, and interrelationship when it interfaces with another system.
- Each system has boundaries that determine its sphere of influence and control.
- The knowledge of the boundaries of a given system is crucial in determining the nature of its interface with other systems for successful design.

Types of Systems

The systems can be divided into the following types −

a. Physical or Abstract Systems

- Physical systems are tangible entities. We can touch and feel them.
- Physical System may be static or dynamic in nature. For example, desks and chairs are the physical parts of computer center which are static. A programmed computer is a dynamic system in which programs, data, and applications can change according to the user's needs.
- Abstract systems are non-physical entities or conceptual that may be formulas, representation or model of a real system.

b. Open or Closed Systems

- An open system must interact with its environment. It receives inputs from and delivers outputs to the outside of the system. For example, an information system which must adapt to the changing environmental conditions.
- A closed system does not interact with its environment. It is isolated from environmental influences. A completely closed system is rare in reality.

c. Adaptive and Non Adaptive System

- Adaptive System responds to the change in the environment in a way to improve their performance and to survive. For example, human beings, animals.
- Non Adaptive System is the system which does not respond to the environment. For example, machines.

d. Permanent or Temporary System

- Permanent System persists for long time. For example, business policies.
- Temporary System is made for specified time and after that they are demolished. For example, A DJ system is set up for a program and it is dissembled after the program.

e. Natural and Manufactured System

- Natural systems are created by the nature. For example, solar system, seasonal system.
- Manufactured System is the man-made system. For example, Rockets, dams, trains.

f. Deterministic or Probabilistic System

- Deterministic system operates in a predictable manner and the interaction between system components is known with certainty. For example, two molecules of hydrogen and one molecule of oxygen makes water.
- Probabilistic System shows uncertain behavior. The exact output is not known. For example, Weather forecasting, mail delivery.

g. Social, Human-Machine, Machine System

- Social System is made up of people. For example, social clubs, societies.
- In Human-Machine System, both human and machines are involved to perform a particular task. For example, Computer programming.
- Machine System is where human interference is neglected. All the tasks are performed by the machine. For example, an autonomous robot.

h. Man–Made Information Systems

- It is an interconnected set of information resources to manage data for particular organization, under Direct Management Control (DMC).
- This system includes hardware, software, communication, data, and application for producing information according to the need of an organization.
  Man-made information systems are divided into three types −
- **Formal Information System** − It is based on the flow of information in the form of memos, instructions, etc., from top level to lower levels of management.
- **Informal Information System** − this is employee based system which solves the day to day work related problems.
- **Computer Based System** − this system is directly dependent on the computer for managing business applications. For example, automatic library system, railway reservation system, banking system, etc.

Systems Models

a. Schematic Models

- A schematic model is a 2-D chart that shows system elements and their linkages.
- Different arrows are used to show information flow, material flow, and information feedback.

b. Flow System Models

- A flow system model shows the orderly flow of the material, energy, and information that hold the system together.
- Program Evaluation and Review Technique (PERT), for example, is used to abstract a real world system in model form.

c. Static System Models

- They represent one pair of relationships such as *activity–time* or *cost–quantity*.
- The Gantt chart, for example, gives a static picture of an activity-time relationship.

d. Dynamic System Models

- Business organizations are dynamic systems. A dynamic model approximates the type of organization or application that analysts deal with.
- It shows an ongoing, constantly changing status of the system. It consists of −
    - Inputs that enter the system
    - The processor through which transformation takes place
    - The program(s) required for processing
    - The output(s) that result from processing.

2. 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, e.g. pocket PCs), ASP (application service provider).

3. Information System

- In a simplest sense, a system that provides information to people in an organization is called information system (IS).
- Information systems in organizations capture and manage data to produce useful information that supports an organization and its employees, customers, suppliers and partners. So, many organizations consider information system to be the essential one.
- Information systems produce information by using data about significant people, places, and things from within the organization and/or from the external environment to make decisions, control operations, analyze problems, and create new products or services.

Types of Information Systems

a. Transaction Processing Systems (TPSs)

- These are the computerized systems that perform and records the daily routine transactions necessary to conduct business. These systems serve the operational level of the organization.
- Some examples include sales order entry, hotel reservation systems, payroll, employee record keeping, and shipping.

- Transaction processing systems are central to a business. TPS failure for a few hours can cause a firm's demise and perhaps other firms linked to it.
- Managers need TPS to monitor the status of internal operations and the firm's relations with external environment.
- TPS are also major producers of information for the other types of systems.
- Online transaction processing systems (OLTPS) is an interactive data processing system that involves a direct connection between TPS programs and users.
- As soon as a single transaction is entered into a computer system, the program interacts immediately with the user for that transaction.
- It is often known as the live system where there is no time lag between data creation and its processing. A good example of this system is online ticket reservation system.

b. Management Information Systems (MISs)

- These are the information systems at the management level of an organization and serve management-level functions like planning, controlling, and decision-making.
- These systems provide reports that are usually generated on a predetermined schedule and appear in prearranged format.
- Typically, these systems use internal data provided by the transaction processing systems. These systems are used for structured decision-making and in some cases for semi-structured decision making as well. Salary analysis and sales reporting are the examples in which MIS can be used.

c. Decision Support Systems (DSSs)

- These systems also serve at the management level of the organization.
- These systems combine data and sophisticated analytical models or data analysis tools to support semistructured and unstructured decision-making.
- These systems use internal information from TPS and MIS, and often information from external sources, such as current stock prices or product prices of competitors.
- DSS have more analytical power than other systems. Contract cost analysis is an example in which DSS can be used.

d. Executive Information Systems (EISs)

- These systems are also called executive support systems (ESSs) and serve the strategic level of the organization.
- These systems are designed to address unstructured decision making through advanced graphics and communication.
- These systems incorporate data about external events such as new tax laws or competitors, but they also draw summarized information from internal MIS and DSS.
- These systems are not designed to solve a specific problem but they provide a generalized computing and telecommunication capacity that can be applied to a changing array of problems.
- 5-year operating plan is an example in which EIS can be used.

e. Expert Systems

- An expert system is an extension of DSS that captures and reproduces the knowledge and expertise of an expert problem solver or decision maker and then simulates the "thinking" or "actions" of that expert. These systems imitate the logic and reasoning of the experts within their respective fields.
- Expert systems are implemented with artificial intelligence (AI) technology that captures, stores, and provides access to the reasoning of the experts.

f. Communication and Collaboration Systems

- These systems enable more effective communications between workers, partners, customers and suppliers to enhance their ability to collaborate.
- These systems use network technology that allows companies to coordinate with other organizations across great distances.
- These systems create new efficiencies and new relationships between an organization, its customers and suppliers, and business partners redefining organizational boundaries.

g. Office Automation Systems

- Office automation (OA) is more than word processing and spreadsheet applications.
- Office automation systems support the wide range of business office activities for improved work flow and communication between workers, regardless of whether or not those workers are located in the same office.
- Office automation functions include word processing, spreadsheet applications, and electronic mails, work group computing, fax processing, work flow management etc.
- Office automation systems can be designed to support both individuals and work groups. Personnel information systems are those designed to meet the needs of a single user.
- They are designed to boost an individual's productivity. Work group information systems, on the other hand, are designed to meet the needs of a work group. They are designed to boost the group's productivity.

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.
- Most organizations use a standard set of steps, called a systems development methodology to develop and support their information systems.
- Components of an Information System: People, Network ,Software, Hardware and Data

4. Systems Development Life Cycle (SDLC)

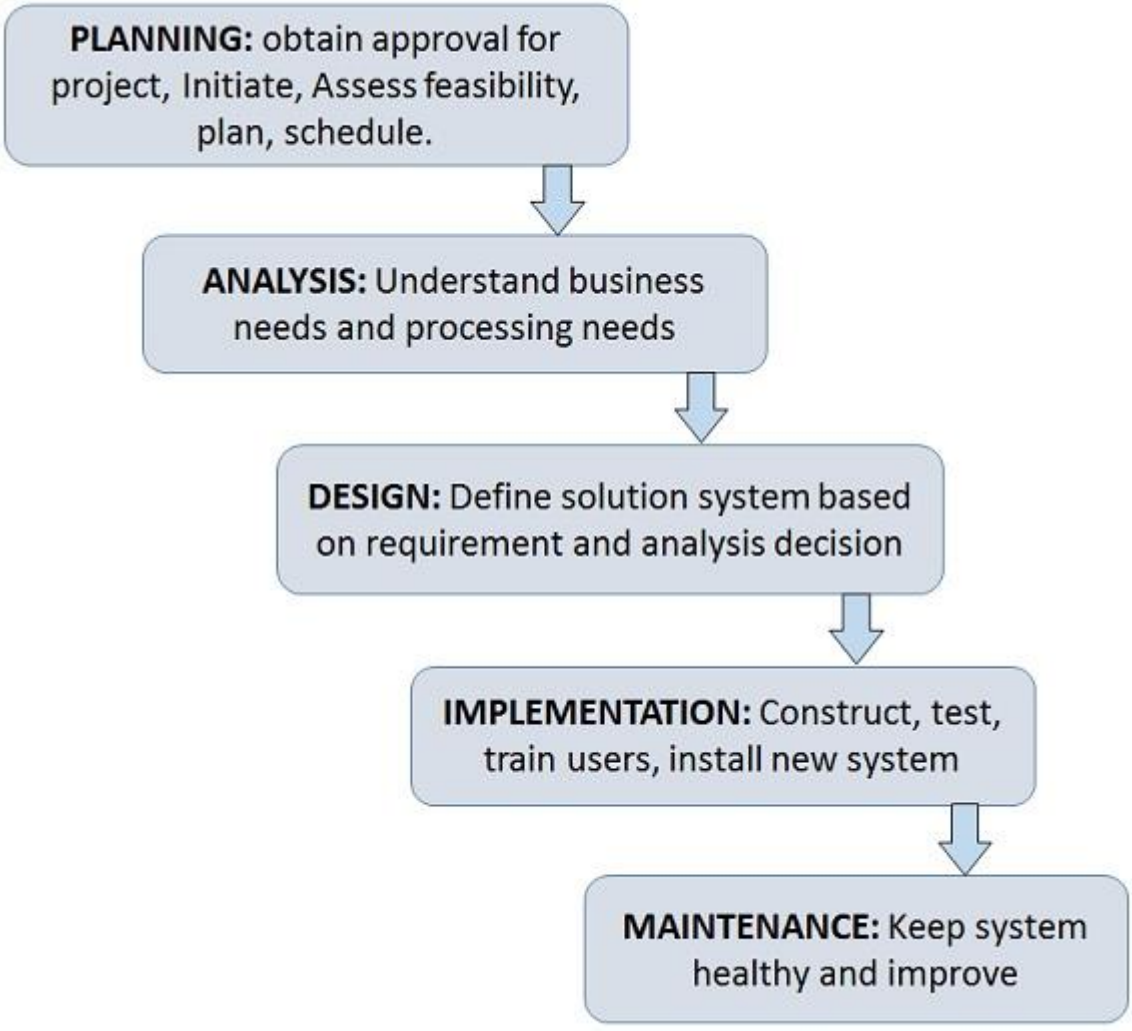
System Development Life Cycle (SDLC) is a conceptual model which includes policies and procedures for developing or altering systems throughout their life cycles.

SDLC is used by analysts to develop an information system. SDLC includes the following activities

- requirements
- design
- implementation
- testing
- deployment
- operations
- maintenance

5. Phases of SDLC

Systems Development Life Cycle is a systematic approach which explicitly breaks down the work into phases that are required to implement either new or modified Information System.



a. Feasibility Study or Planning

- Define the problem and scope of existing system.
- Overview the new system and determine its objectives.
- Confirm project feasibility and produce the project Schedule.
- During this phase, threats, constraints, integration and security of system are also considered.
- A feasibility report for the entire project is created at the end of this phase.

b. Analysis and Specification

- Gather, analyze, and validate the information.
- Define the requirements and prototypes for new system.
- Evaluate the alternatives and prioritize the requirements.
- Examine the information needs of end-user and enhances the system goal.
- A Software Requirement Specification (SRS) document, which specifies the software, hardware, functional, and network requirements of the system is prepared at the end of this phase.

c. System Design

- Includes the design of application, network, databases, user interfaces, and system interfaces.
- Transform the SRS document into logical structure, which contains detailed and complete set of specifications that can be implemented in a programming language.
- Create a contingency, training, maintenance, and operation plan.
- Review the proposed design. Ensure that the final design must meet the requirements stated in SRS document.
- Finally, prepare a design document which will be used during next phases.

d. Implementation

- Implement the design into source code through coding.
- Combine all the modules together into training environment that detects errors and defects.
- A test report which contains errors is prepared through test plan that includes test related tasks such as test case generation, testing criteria, and resource allocation for testing.
- Integrate the information system into its environment and install the new system.

e. Maintenance/Support

- Include all the activities such as phone support or physical on-site support for users that is required once the system is installing.
- Implement the changes that software might undergo over a period of time, or implement any new requirements after the software is deployed at the customer location.
- It also includes handling the residual errors and resolve any issues that may exist in the system even after the testing phase.
- Maintenance and support may be needed for a longer time for large systems and for a short time for smaller systems.

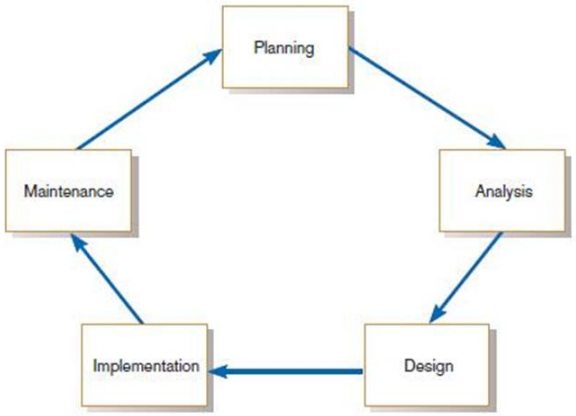


Figure
The systems development life cycle

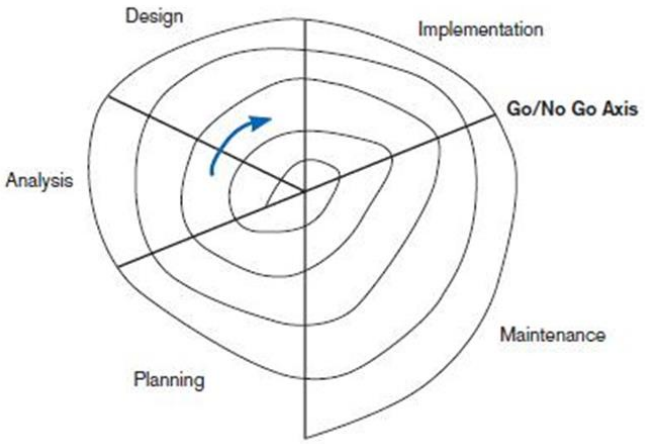


Figure
Evolutionary model

6. The Heart of the Systems Development Process

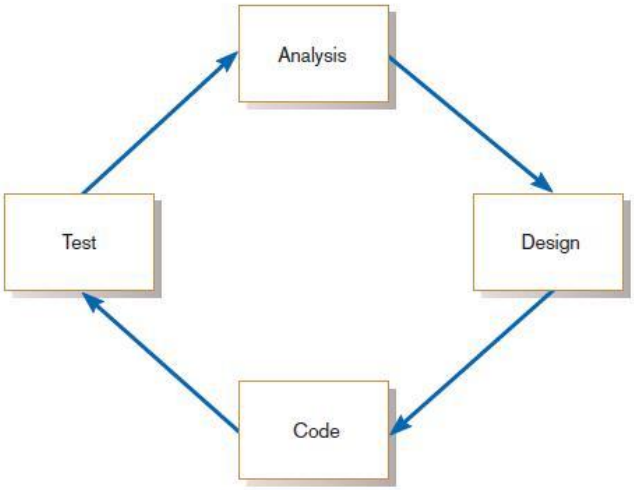


Figure
The analysis–design–code–test loop

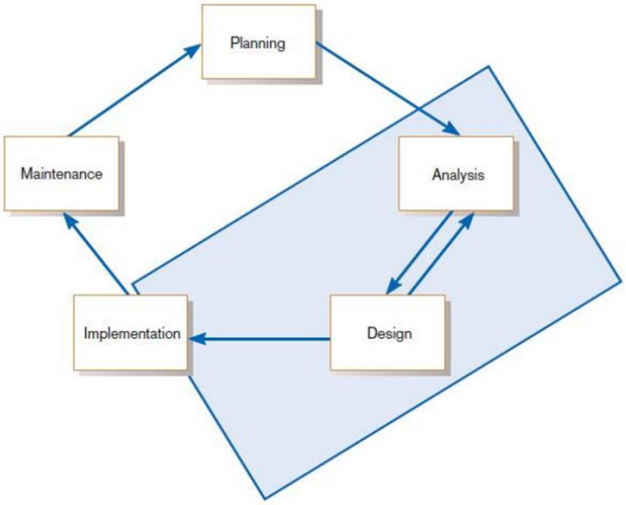
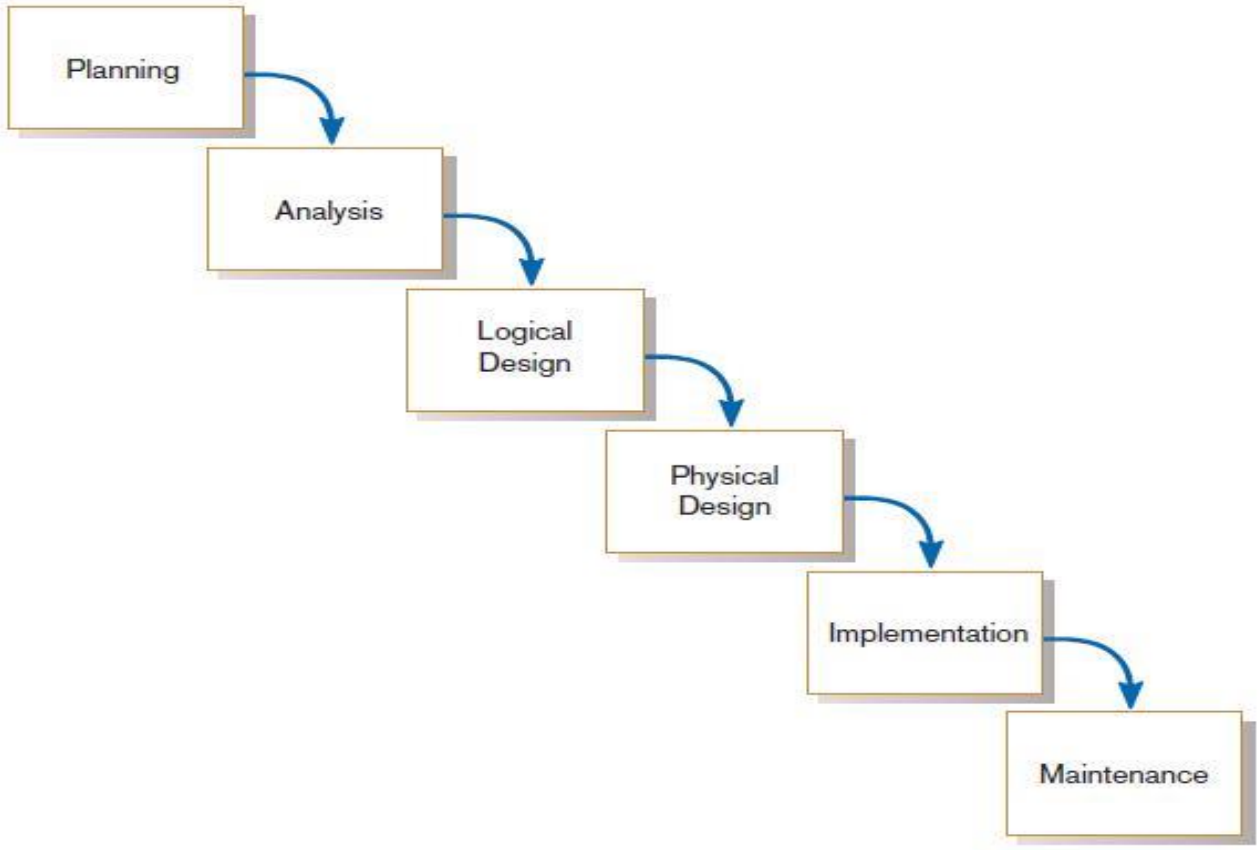


Figure
The heart of systems development

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

7. Traditional Waterfall SDLC

- It is a sequential model that divides software development into pre-defined phases.
- Each phase must be completed before the next phase can begin with no overlap between the phases.
- Each phase is designed for performing specific activity during the SDLC phase.
- It was introduced in 1970 by Winston Royce.



Waterfall Methodology can be used when:

- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

Advantages	Dis-Advantages
• Before the next phase of development, each phase must be completed	• Error can be fixed only during the phase
• Suited for smaller projects where requirements are well defined	• It is not desirable for complex project where requirement changes frequently
• They should perform quality assurance test (Verification and Validation) before completing each stage	• Testing period comes quite late in the developmental process
• Elaborate documentation is done at every phase of the software's development cycle	• Documentation occupies a lot of time of developers and testers
• Project is completely dependent on project team with minimum client intervention	• Clients valuable feedback cannot be included with ongoing development phase
• Any changes in software is made during the process of the development	• Small changes or errors that arise in the completed software may cause a lot of problems

8. Approaches for Improving System Development

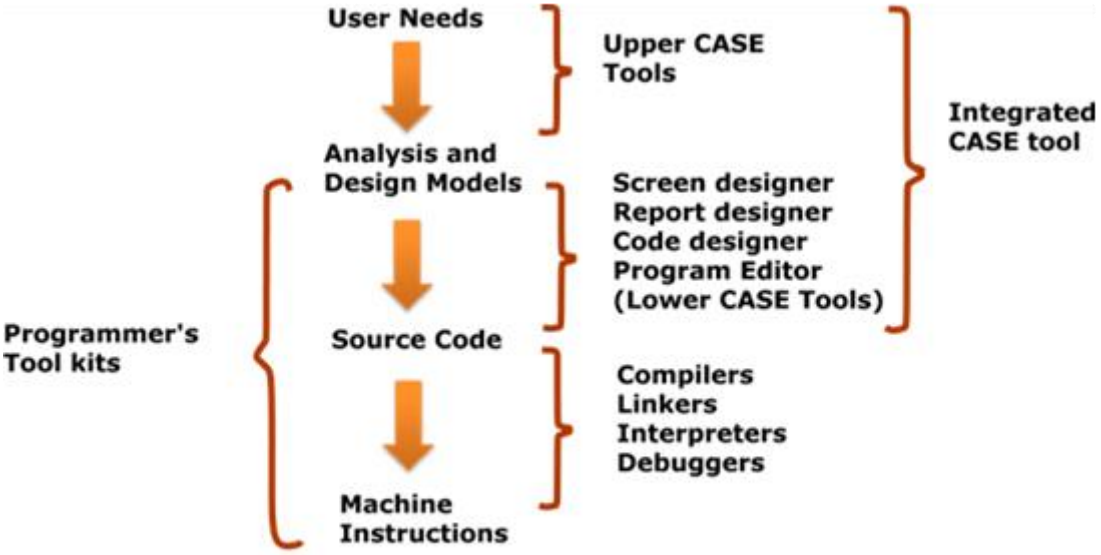
- Attempts to make systems development less of an art and more of a science are usually referred to as systems engineering or software engineering

CASE Tools

- Full form: Computer-Aided Software Engineering tools
- Diagramming tools/program enable graphical representation and automate the SDLC activities.
- Computer displays and report generators help prototype how systems "look and feel".
- Documentation generators standardize technical and user documentation.
- Code generators enable automatic generation of programs and database code directly from design documents, diagrams, forms, and reports.
- Supports a wide variety of SDLC activities: project identification and selection, project initiation and planning, analysis, design, and implementation and maintenance.

Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:



CASE Tool And Application Development

a. 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.

b. Upper Case Tools

Upper CASE tools are used in planning, analysis and design stages of SDLC.

c. 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.

Types of CASE 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.

Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project. Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain,

Adobe RoboHelp for documentation.

Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

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

Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For

Example, Fossil, Git, Accu REV.

Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change

tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built Modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

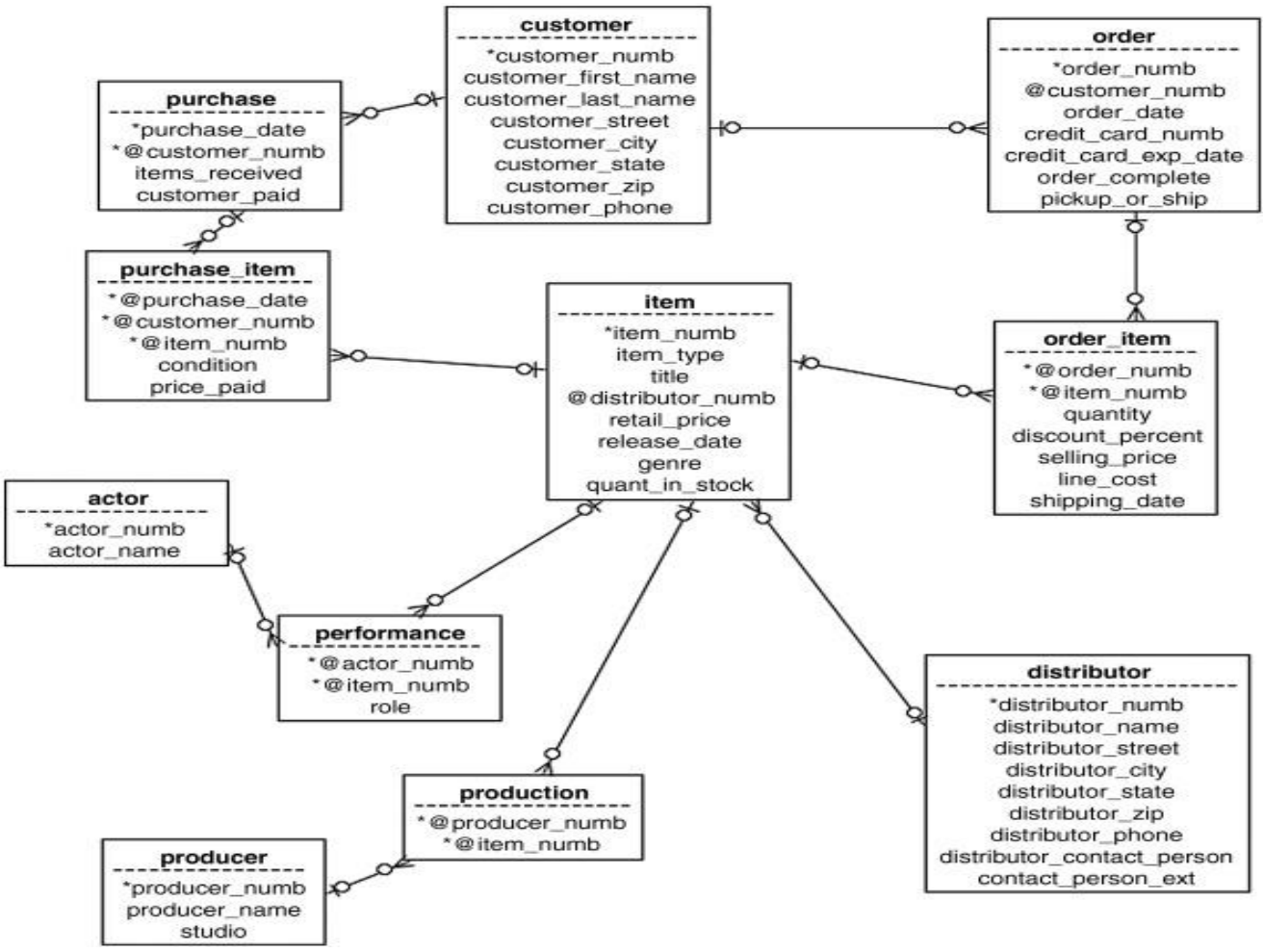


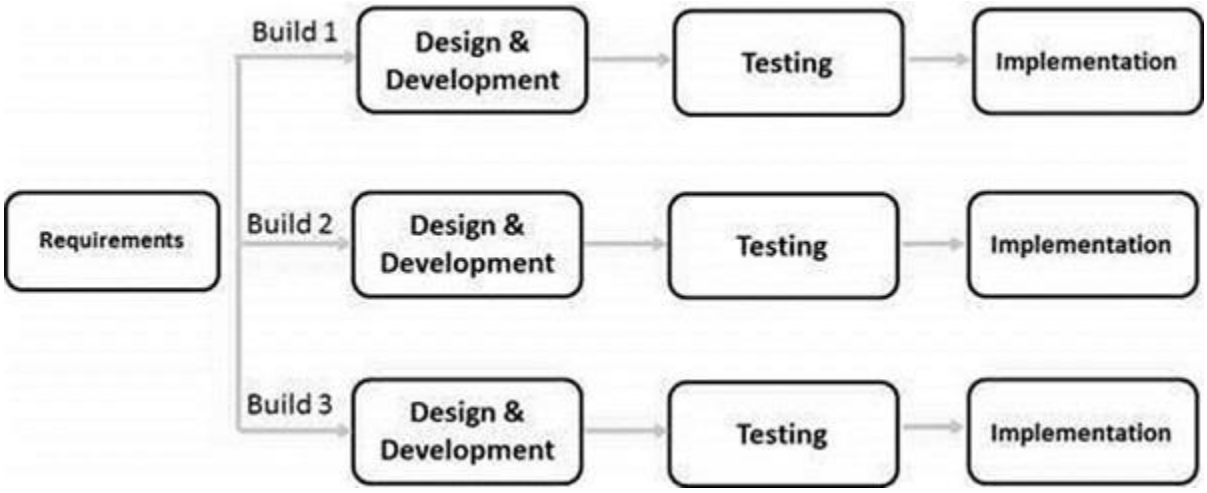
TABLE 1-2 Examples of CASE Usage within the SDLC

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)

9. SDLC Models

Iterative/Incremental Model:

- Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented.
- At each iteration, design modifications are made and new functional capabilities are added.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).



Advantages

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the life cycle, software is produced early which facilitates customer evaluation and feedback.

Disadvantages

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which risk is.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

Spiral Model

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Advantages

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

Disadvantages

- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex

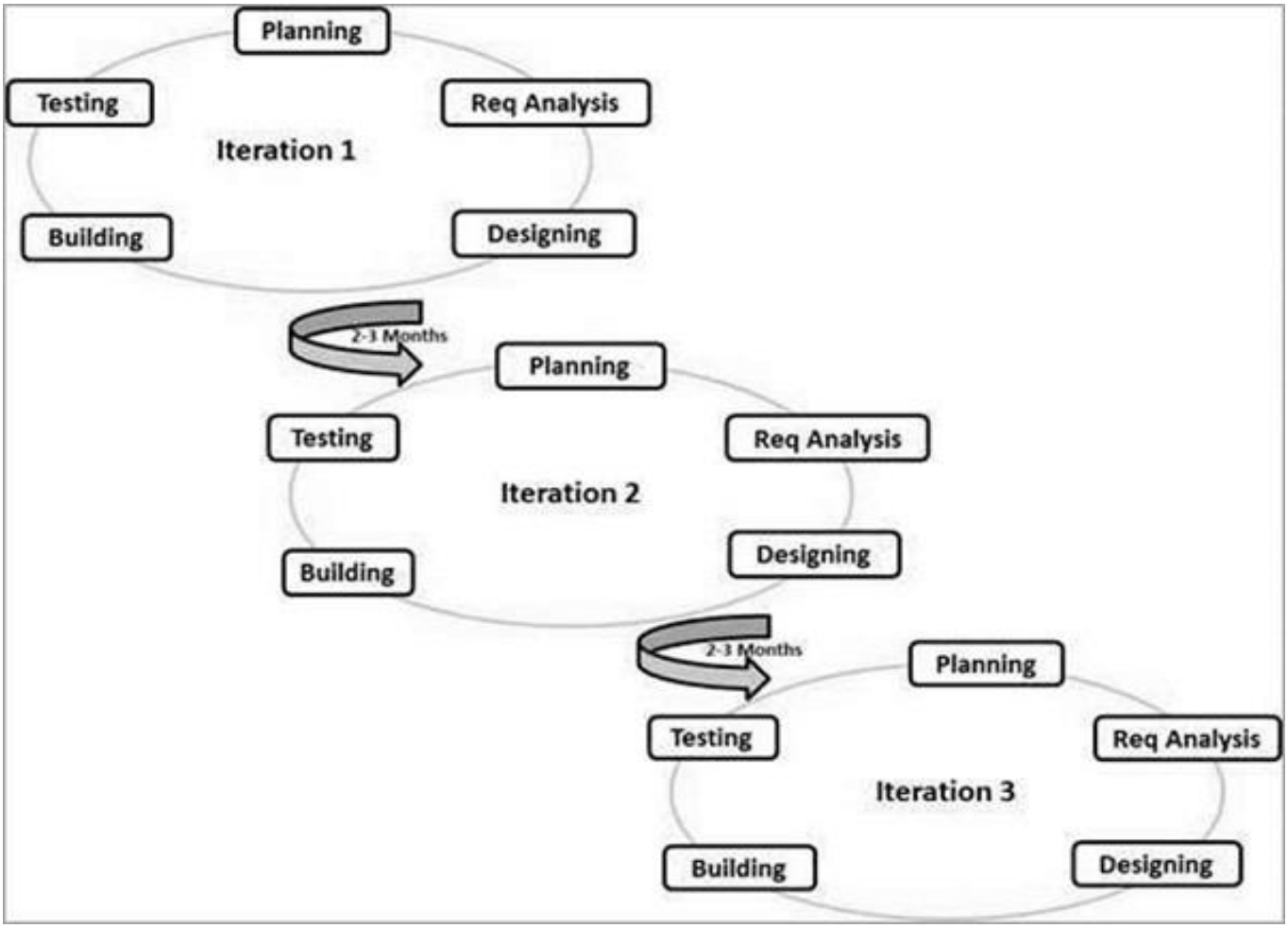
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

Agile Model

- Agile means able to move quickly and easily.
- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds.
- These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.

Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.



Advantages:

- Is a very realistic approach to software development?
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

Disadvantages

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

When to Use

- Unpredictable or dynamic requirements
- Responsible and motivated developers
- Customers who understand the process and will get involved

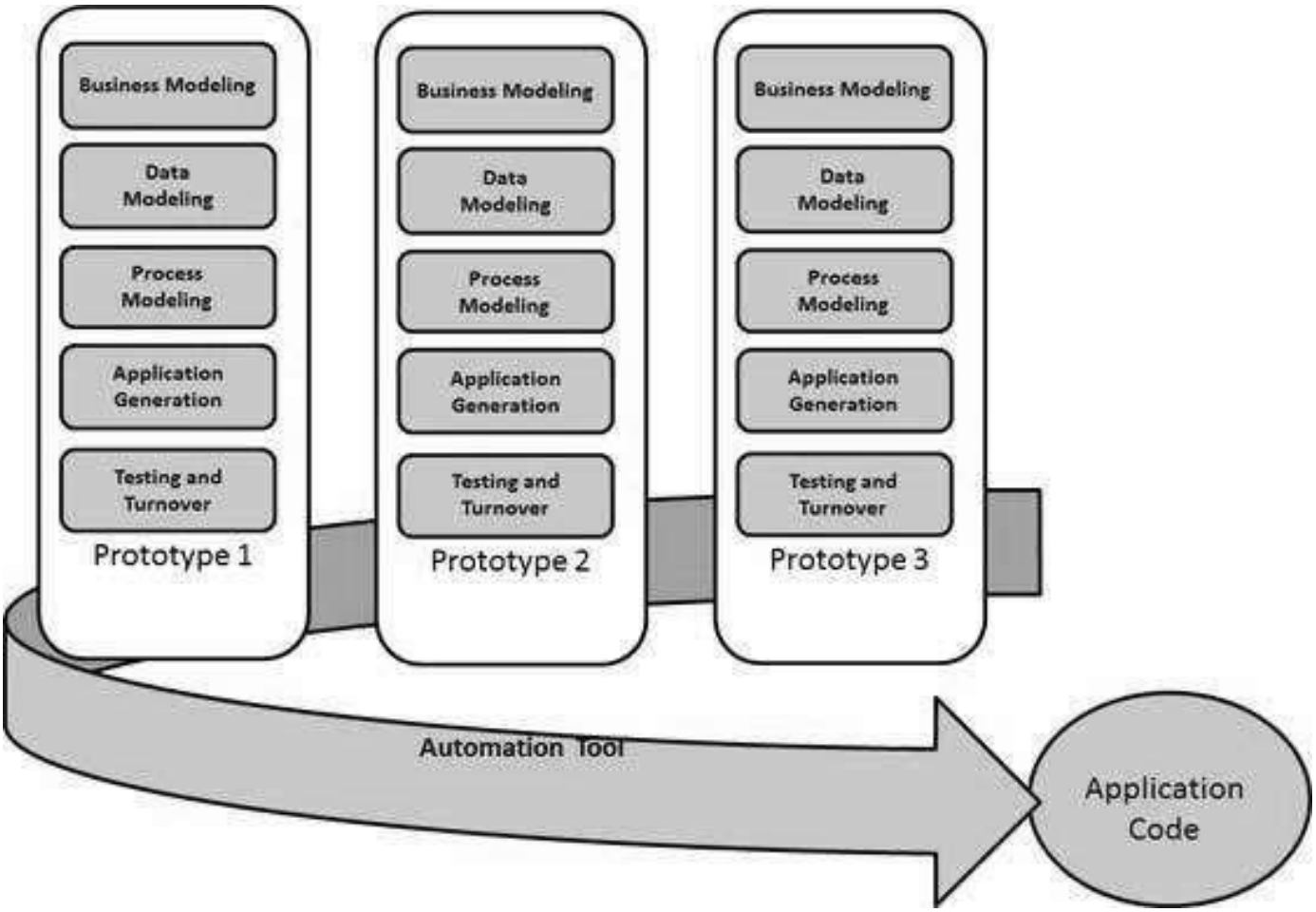
RAD Model

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.



Advantages:

- Changing requirements can be accommodated.
- Progress can be measured.
- Iteration time can be short with use of powerful RAD tools.
- Productivity with fewer people in a short time.
- 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:

- Dependency on technically strong team members for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/designers.
- High dependency on modelling skills.
- Inapplicable to cheaper projects as cost of Modelling and automated code generation is very high.
- Management complexity is more.

- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.
- Suitable for project requiring shorter development times.

When to Use:

- RAD should be used only when a system can be modularized to be delivered in an incremental manner.
- It should be used if there is a high availability of designers for Modelling.
- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

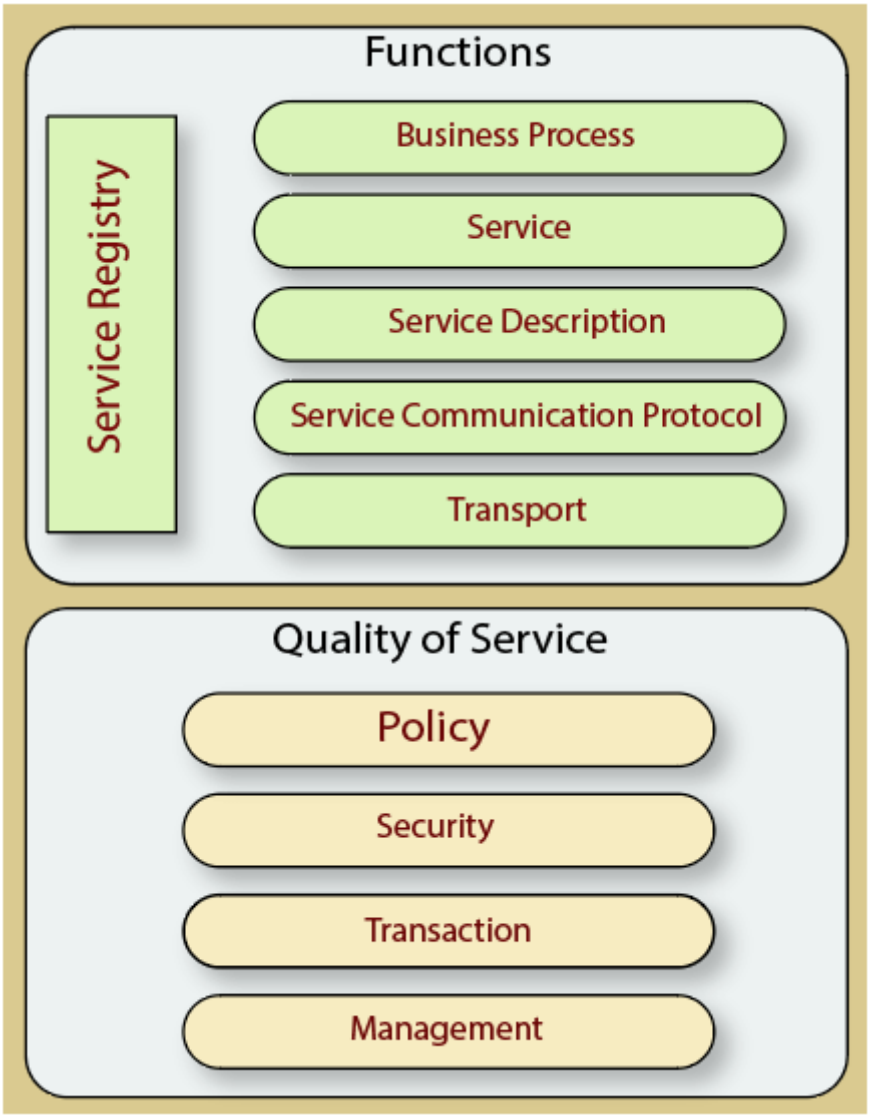
Service-Oriented Architecture (SOA)

- A service is a well-defined, self-contained function that represents a unit of functionality.
- A Service-Oriented Architecture or SOA is a design pattern which is designed to build distributed systems that deliver services to other applications through the protocol.
- Each service executes one action. Each service can be used in other application within the organization or even in other organizations.
- It is only a concept and not limited to any programming language or platform.

Characteristics:

- They are loosely coupled.
- They support interoperability.
- They are location-transparent
- They are self-contained.

Components of service-oriented architecture



Functional aspects

- Transport - It transports the service requests from the service consumer to the service provider and service responses from the service provider to the service consumer.
- Service Communication Protocol - It allows the service provider and the service consumer to communicate with each other.
- Service Description - It describes the service and data required to invoke it.
- Service - It is an actual service.
- Business Process - It represents the group of services called in a particular sequence associated with the particular rules to meet the business requirements.
- Service Registry - It contains the description of data which is used by service providers to publish their services.

Quality of Service aspects

- Policy - It represents the set of protocols according to which a service provider make and provide the services to consumers.
- Security - It represents the set of protocols required for identification and authorization.
- Transaction - It provides the surety of consistent result. This means, if we use the group of services to complete a business function, either all must complete or none of the complete.
- Management - It defines the set of attributes used to manage the services.

Advantages of SOA

- Easy to integrate - In a service-oriented architecture, the integration is a service specification that provides implementation transparency.
- Manage Complexity - Due to service specification, the complexities get isolated, and integration becomes more manageable.
- Platform Independence - The services are platform-independent as they can communicate with other applications through a common language.
- Loose coupling - It facilitates to implement services without impacting other applications or services.
- Parallel Development - As SOA follows layer-based architecture, it provides parallel development.
- Available - The SOA services are easily available to any requester.
- Reliable - As services are small in size, it is easier to test and debug them.

Extreme Programming (XP)

Extreme programming (XP) is one of the most important software development frameworks of Agile models.

It is used to improve software quality and responsiveness to customer requirements.

The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

- 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.

Working model

- 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.

Advantages:

- More (and better) communication among developers,
- Higher levels of productivity,
- Higher-quality code, and
- Reinforcement of the other practices in eXtreme Programming, such as the code and test discipline
- Test based approach to requirements and quality assurance.

Disadvantage:

- It is not for everyone and is not applicable to every project.
- Programming pairs is costly
- Skilled/specialized manpower is needed.

Object-Oriented Analysis and Design (OOAD)

Based on objects rather than data or processes

Object: a structure encapsulating attributes and behaviors of a real-world entity

Object class: a logical grouping of objects sharing the same attributes and behaviors

Inheritance: hierarchical arrangement of classes enables subclasses to inherit properties of super classes

- 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.

Advantages	Disadvantages
Focuses on data rather than the procedures as in Structured Analysis.	Functionality is restricted within objects. This may pose a problem for systems which are intrinsically procedural or computational in nature.
The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system.	It cannot identify which objects would generate an optimal system design.
The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system.	The object-oriented models do not easily show the communications between the objects in the system.
It allows effective management of software complexity by the virtue of modularity.	All the interfaces between the objects cannot be represented in a single diagram.
It can be upgraded from small to large systems at a greater ease than in systems following structured analysis.	

B. The Origin of Software

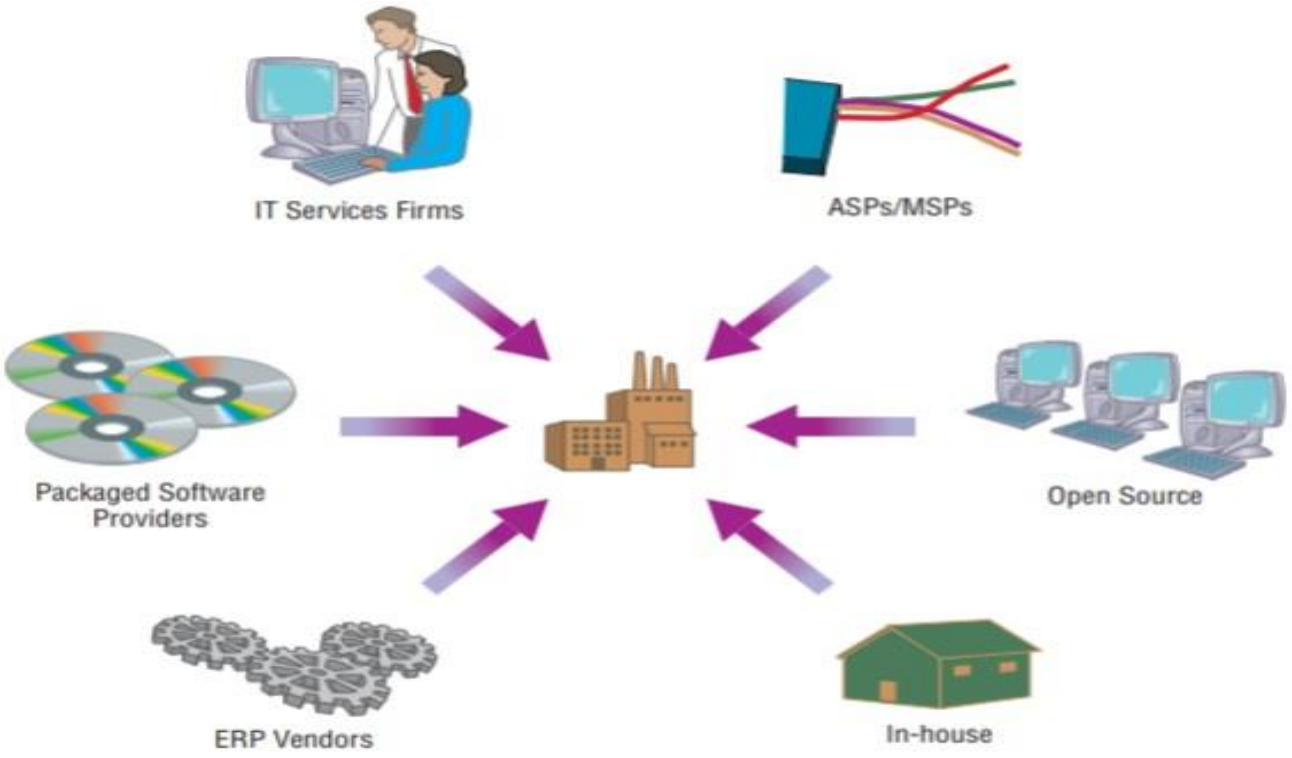
1. Introduction

- 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.

2. Software Acquisition (assets/achievement)

- 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. Another example of outsourcing would be if you hired a company to run your applications at your site on your computers. In some cases, an organization employing such an arrangement will dissolve some or all of its information systems (IS) unit and fire all of it IS employees. Often the company brought in to run the organization's computing will rehire many of the organization's original IS unit employees.

The reasons behind outsourcing are:

- freeing up internal resources,
- increasing the revenue potential of the organization,
- reducing time to market,
- increasing process efficiencies, and
- Outsourcing noncore activities.

E.g. biggest outsourcing companies are IBM and EDS.

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

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.
- 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.

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.
- Oracle is exclusively a software company known primarily for its database software, but Oracle also makes enterprise systems.
- Software companies develop what are sometimes called prepackaged or off-the-shelf systems.
- Software companies develop software to run on many different computer platforms, from microcomputers to large mainframes.

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.

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.
- A well-known example of cloud computing is Google Apps, where users can share and create documents, spreadsheets, and presentations. Another well-known example is Amazon web services, Azur, which provides customer relationship management software online.

Benefits:

- freeing internal IT staff,
- gaining access to applications faster than via internal development
- Achieving lower cost access to corporate-quality applications.

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.

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. 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.

E.g. Bank application, etc.

3. 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.

C. Managing the Information System Project

1. 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.

The project management process involves four phases:

- Initiating the project
- Planning the project
- Executing the project
- Closing down the project

a. 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:

Establishing the project initiation team:

This activity involves organizing project team members to assist in accomplishing the project initiation activities.

Establishing a relationship with the customer:

A thorough understanding of your customer builds stronger partnerships and higher levels of trust

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.

Establishing management procedures:

Successful projects require the development of effective management procedures.

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.

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.

b. 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.

Describing project scope, alternatives, and feasibility:

The purpose of this activity is to understand the content and complexity of the project.

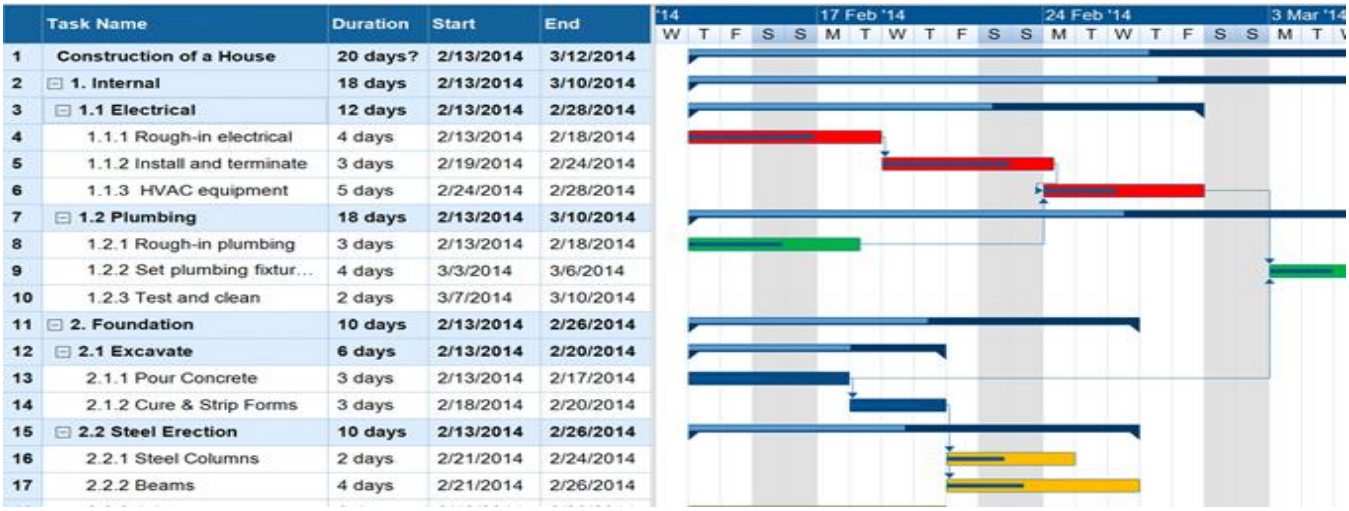
During this activity, you should reach agreement on the following questions:

- 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.

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.
- 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.



	Task Name	Duration	Start	End
1	Construction of a House	20 days?	2/13/2014	3/12/2014
2	1. Internal	18 days	2/13/2014	3/10/2014
3	1.1 Electrical	12 days	2/13/2014	2/28/2014
4	1.1.1 Rough-in electrical	4 days	2/13/2014	2/18/2014
5	1.1.2 Install and terminate	3 days	2/19/2014	2/24/2014
6	1.1.3 HVAC equipment	5 days	2/24/2014	2/28/2014
7	1.2 Plumbing	18 days	2/13/2014	3/10/2014
8	1.2.1 Rough-in plumbing	3 days	2/13/2014	2/18/2014
9	1.2.2 Set plumbing fixtur...	4 days	3/3/2014	3/6/2014
10	1.2.3 Test and clean	2 days	3/7/2014	3/10/2014
11	2. Foundation	10 days	2/13/2014	2/26/2014
12	2.1 Excavate	6 days	2/13/2014	2/20/2014
13	2.1.1 Pour Concrete	3 days	2/13/2014	2/17/2014
14	2.1.2 Cure & Strip Forms	3 days	2/18/2014	2/20/2014
15	2.2 Steel Erection	10 days	2/13/2014	2/26/2014
16	2.2.1 Steel Columns	2 days	2/21/2014	2/24/2014
17	2.2.2 Beams	4 days	2/21/2014	2/26/2014

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.

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

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.

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.

Identifying and assessing risk:

- The goal of this activity is to identify sources of project risk and estimate the consequences of those risks.
- 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.

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.

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.

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.

c. 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.

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.

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.
- Monitoring progress 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.

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

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.

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.

d. Closing Down the project

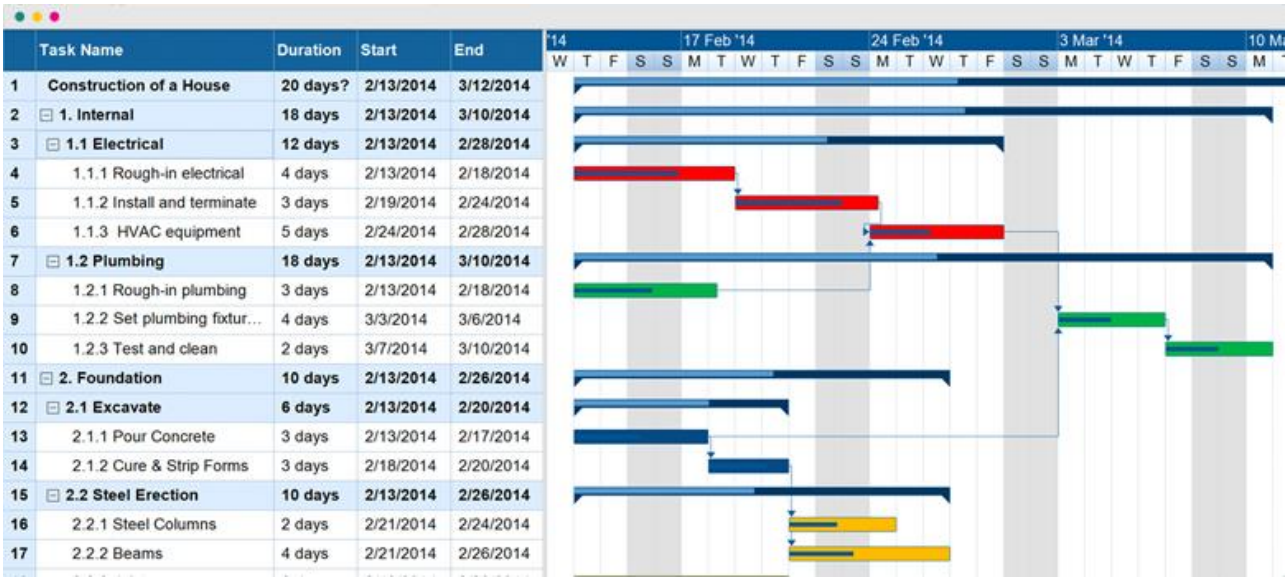
- The focus of project closedown is to bring the project to an end. Projects can conclude with a natural or unnatural termination.
- A natural termination occurs when the requirements of the project have been met—the project has been completed and is a success.
- An unnatural termination occurs when the project is stopped before completion.
- 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 (lack), 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.

2. Representing and Scheduling Project Plans

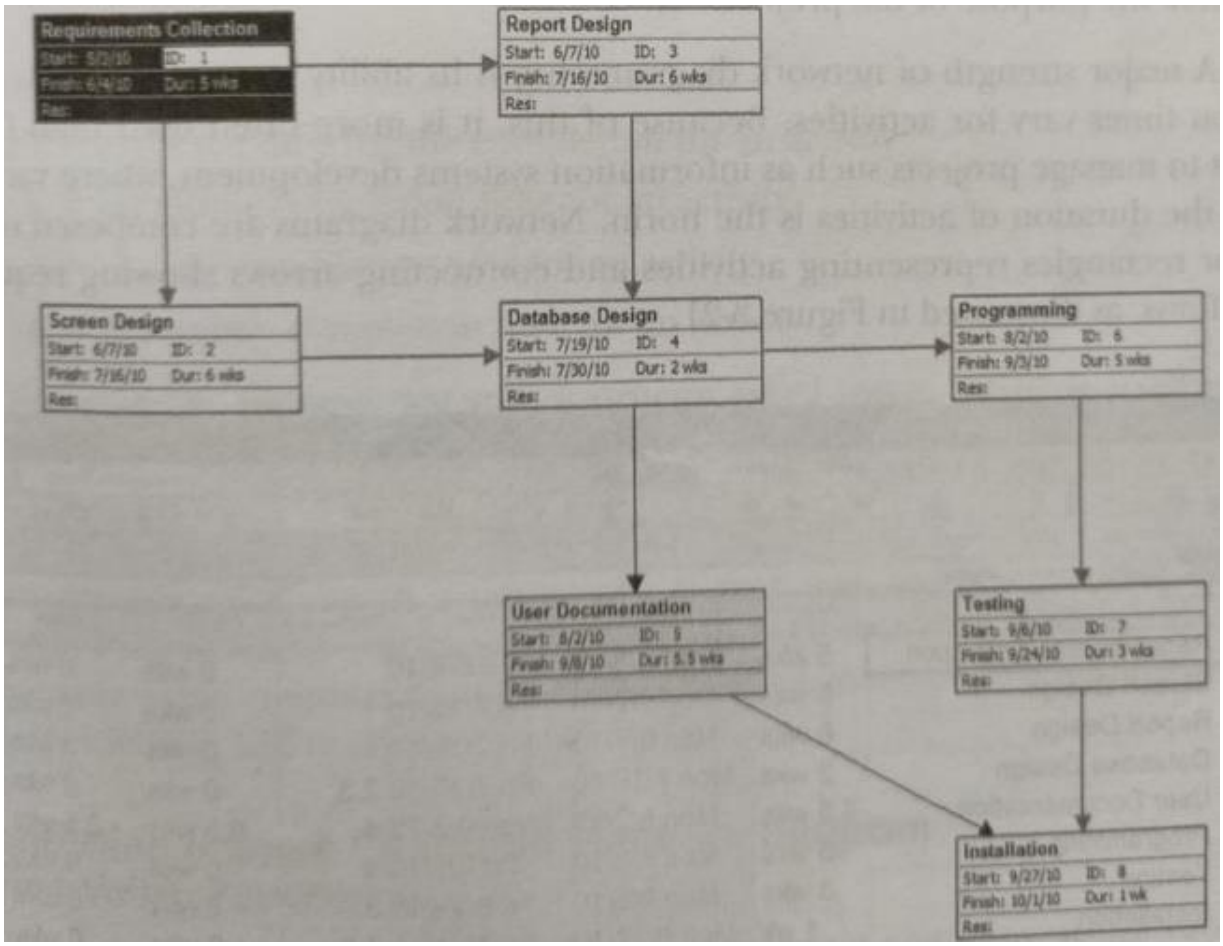
- A project manager has a wide variety of techniques available for depicting (to represent or show something in a picture or story) and documenting project plans.
- These planning documents can take the form of graphical or textual reports, although graphical reports have become most popular for depicting 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.
- Graphical diagrams that depict project plans:

A Gantt char



	Task Name	Duration	Start	End
1	Construction of a House	20 days?	2/13/2014	3/12/2014
2	1. Internal	18 days	2/13/2014	3/10/2014
3	1.1 Electrical	12 days	2/13/2014	2/28/2014
4	1.1.1 Rough-in electrical	4 days	2/13/2014	2/18/2014
5	1.1.2 Install and terminate	3 days	2/19/2014	2/24/2014
6	1.1.3 HVAC equipment	5 days	2/24/2014	2/28/2014
7	1.2 Plumbing	18 days	2/13/2014	3/10/2014
8	1.2.1 Rough-in plumbing	3 days	2/13/2014	2/18/2014
9	1.2.2 Set plumbing fixtur...	4 days	3/3/2014	3/6/2014
10	1.2.3 Test and clean	2 days	3/7/2014	3/10/2014
11	2. Foundation	10 days	2/13/2014	2/26/2014
12	2.1 Excavate	6 days	2/13/2014	2/20/2014
13	2.1.1 Pour Concrete	3 days	2/13/2014	2/17/2014
14	2.1.2 Cure & Strip Forms	3 days	2/18/2014	2/20/2014
15	2.2 Steel Erection	10 days	2/13/2014	2/26/2014
16	2.2.1 Steel Columns	2 days	2/21/2014	2/24/2014
17	2.2.2 Beams	4 days	2/21/2014	2/26/2014

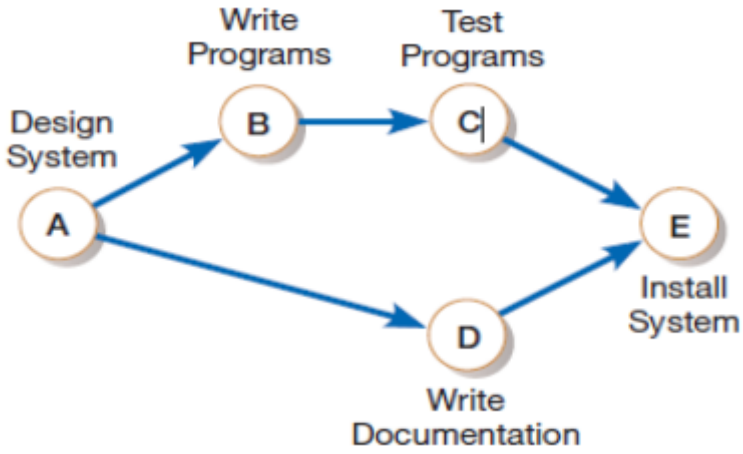
A network diagram



- 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



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:

$$ET = (o+4r+p)/6$$

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

3. 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

Functions:

- **Plan Projects** – Easily prepare projects while taking previous experience into account.
- **Keeping Track of Completion, Time and Cost of the Project** – Alert the right people when things go off course.
- **Manage Time and Schedules** – Manage time on work items and take people's work schedules into consideration.
- **The Allocation of Resources** – Ensuring that the right people work on the right things at the right time.
- **Estimated Project Budget, Including Personnel Costs** – Ensuring real-time monitoring of not only the time but also the budget as well.
- **Collaboration and Communication** – Post comments and concerns, communicate with external stakeholders while keeping a complete historical record of all actions.
- **Files & Documentation** – Easily document requirements and specifications directly or via a file.
- **Simple to Use** – The software enables users to complete tasks, and it's simple to use.