Information Systems Development

By the end of the topic, you should be able to:

- Describe the motivation for a system development process in terms of the Capability Maturity Model (CMM) for quality management.
- Differentiate between the system life cycle and a system development methodology.
- Describe 10 basic principles of system development.
- Define problems, opportunities, and directives - the triggers for systems development projects.
- Describe the PIECES framework for categorizing problems, opportunities, and directives.

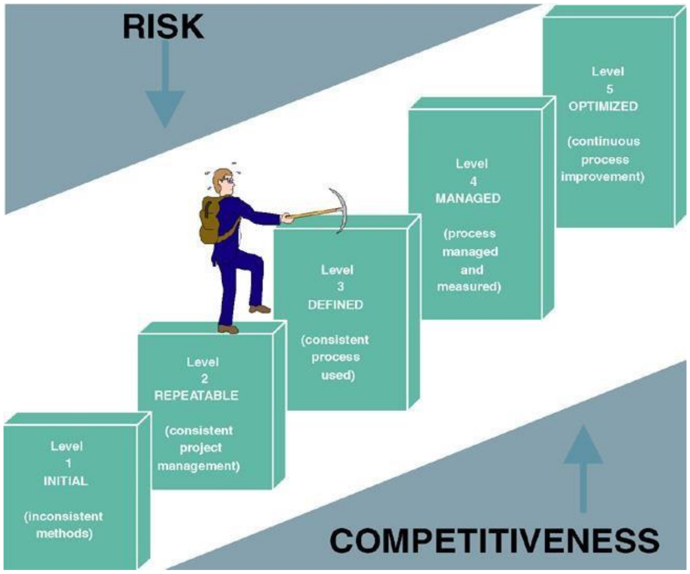
Process of System Development

- **System development process** is a set of activities, methods, best practices, deliverables, and automated tools that stakeholders use to develop and continuously improve information systems and software.
- Organizations embrace standardized processes to develop information systems because information systems are a complex product; an information system includes data, process, and communications building blocks and technologies that must serve the needs of a variety of stakeholders.
- Because of the complexity of information systems 70% of information systems development projects have failed to meet expectations, cost more than budgeted, and are delivered much later than promised.
- Increasingly, organizations have no choice but to adopt and follow a standardized systems development process.
    - First, using a consistent process for system development creates efficiencies that allow management to shift resources between projects.
    - Second, a consistent methodology produces consistent documentation that reduces lifetime costs to maintain the systems.
    - Finally, some governments such as the U.S.A have mandated that any organization seeking to develop software for the government must adhere to certain quality management requirements. A consistent process promotes quality.
- Many organizations have aggressively committed to total quality management goals in order to Increase their competitiveness advantage. In order to realize quality and productivity improvements, many organizations have turned to project and process management frameworks such as the Capability Maturity Model.
- There are many variations of system development process.
- Using a consistent process for system development:
    - Create efficiencies that allow management to shift resources between projects
    - Produces consistent documentation that reduces lifetime costs to maintain the systems
    - Promotes quality

CMM Process Management Model

- **Capability Maturity Model** (CMM) is a standardized framework for assessing the maturity level of an organization’s information system development and management processes and products. It consists of five levels of maturity:

1. **Level 1 - Initial**: System development projects follow no prescribed process.
2. **Level 2 - Repeatable**: Project management processes and practices established to track project costs, schedules, and functionality.
3. **Level 3 - Defined**: Standard system development process (methodology) is purchased or developed. All projects use a version of this process.
4. **Level 4 - Managed**: Measurable goals for quality and productivity are established.
5. **Level 5 - Optimizing**: The standardized system development process is continuously monitored and improved based on measures and data analysis established in Level 4.


RISK
Level 5 OPTIMIZED (continuous process improvement)
Level 4 MANAGED (process managed and measured)
Level 3 DEFINED (consistent process used)
Level 2 REPEATABLE (consistent project management)
Level 1 INITIAL (inconsistent methods)
COMPETITIVENESS

- It is very important to recognize that each level is a prerequisite for the next level. Currently, many organizations are working hard to achieve at least CMM Level 3 (sometimes driven by a government or organizational mandate). A central theme to achieving Level 3 (Defined) is the use of a standard process or methodology to build or integrate systems.

Level 1 - Initial (Chaotic)

- Processes at this level are typically undocumented and in a state of dynamic change, tending to be driven in an *ad hoc*, uncontrolled and reactive manner by users or events. This provides a chaotic or unstable environment for the processes.
- System development projects follow no prescribed process. Each developer uses his/her own tools and methods.
- Success or failure is usually a function of the skills and experience of the project team.
- The process is unpredictable and not repeatable.
- A project encounters many crises and is frequently over budget and behind schedule.
- Documentation is sporadic and inconsistent from one project to another. This creates problems for those who must maintain a system over life time.
- Almost all projects begin here.

Level 2 - Repeatable

- At this level, project management process and practices have been established to track project costs, schedules and functionalities
- The focus is on project management and not systems development.
- A systems development methodology is followed but this varies from project to project.
- Process discipline is unlikely to be rigorous, but where it exists, it may help to ensure that existing processes are maintained during times of stress.
- Effort is made to repeat earlier project successes; some processes are repeatable, possibly with consistent results.
- Success or failure is still a function of the skills and experience of the project team.

Level 3 - Defined

- Processes at this level have sets of defined and documented standard processes established and subject to some degree of improvement over time.
- These standard processes are in place and used to establish consistency of process performance across the organization.

Level 4 - Managed

- Processes at this level use well established process metrics, and management can effectively control the processes such as software development.
- Management can identify ways to adjust and adapt the process to particular projects without measurable losses of quality or deviations from specifications.
- Process Capability is established from this level.

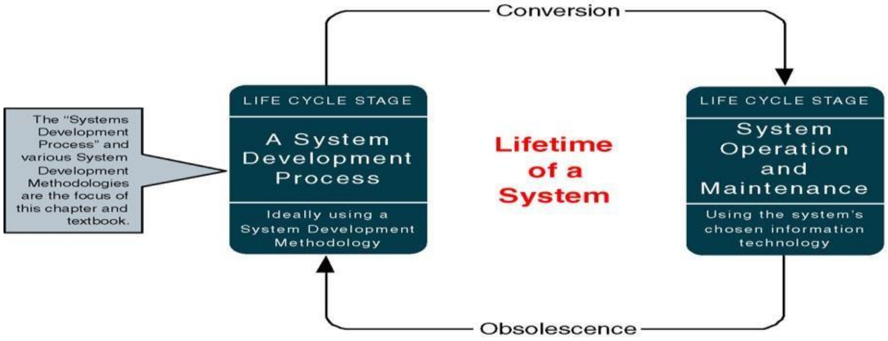
Level 5 - Optimizing

- Processes at this level focus on continually improving process performance through both incremental and innovative technological changes/improvements.
- Processes are concerned with addressing statistical *common causes* of process variation and changing the process (for example, to shift the mean of the process performance) to improve process performance. This would be done at the same time as maintaining the likelihood of achieving the established quantitative process-improvement objectives.
- There are only a few companies in the world that have attained this level 5.

System Life Cycle versus System Methodology

- **System life cycle** is the factoring of the lifetime of an information system into two stages
  (1) Systems development
  (2) Systems operation and maintenance
- First you build it, and then you use and maintain it. Eventually, you cycle back to redevelopment of a new system.
- The terms **system life cycle** and **system development methodology** are frequently and incorrectly interchanged. Most system development processes are derived from a natural system life cycle. The system life cycle just happens.

- In System life cycle there are two key stages that trigger a change from one stage to the other:
  - When a system cycles from development to operation and maintenance, a conversion must take place.
  - At some point in time, obsolescence occurs (or is imminent) and a system cycles from operation and maintenance to redevelopment.
- A system may be in more than one stage at the same time. For example, one system may be in operation and support while the next version is in development.


Conversion
LIFE CYCLE STAGE
A System Development Process
Ideally using a System Development Methodology
Lifetime of a System
LIFE CYCLE STAGE
System Operation and Maintenance
Using the system's chosen information technology
Obsolescence
The "Systems Development Process" and various System Development Methodologies are the focus of this chapter and textbook.

- **System development methodology** is a formalized approach to the systems development process; a standardized development process that defines (as in CMM Level 3) a set of activities, methods, best practices, deliverables, and automated tools that system developers and project managers are to use to develop and continuously improve information systems and software.
- A systems development methodology executes the systems development stage of the system life cycle.
- Each individual Information system has its own system life cycle.
- The methodology is the standard process to build and maintain that system and all other information systems through their life cycles.
- Consistent with the goals of the CMM, methodologies ensure that:
  - A consistent, reproducible approach is applied to all projects.
  - There is reduced risk associated with shortcuts and mistakes.
  - Complete and consistent documentation is produced from one project to the next.
  - Systems analysts, designers, and builders can be quickly reassigned between projects because all use the same process.
  - As development teams and staff constantly dialogue, the results of prior work can be easily retrieved and understood by those who follow.
- Methodologies can be purchased or home-grown. Why purchase a methodology?
  - Many information system organizations can't afford to dedicate staff to the development and continuous improvement of a home-grown methodology.
  - Methodology vendors have a vested interest in keeping their methodologies current with the latest business and technology trends.
  - Home-grown methodologies are usually based on generic methodologies and techniques that are well documented in books and on web sites.

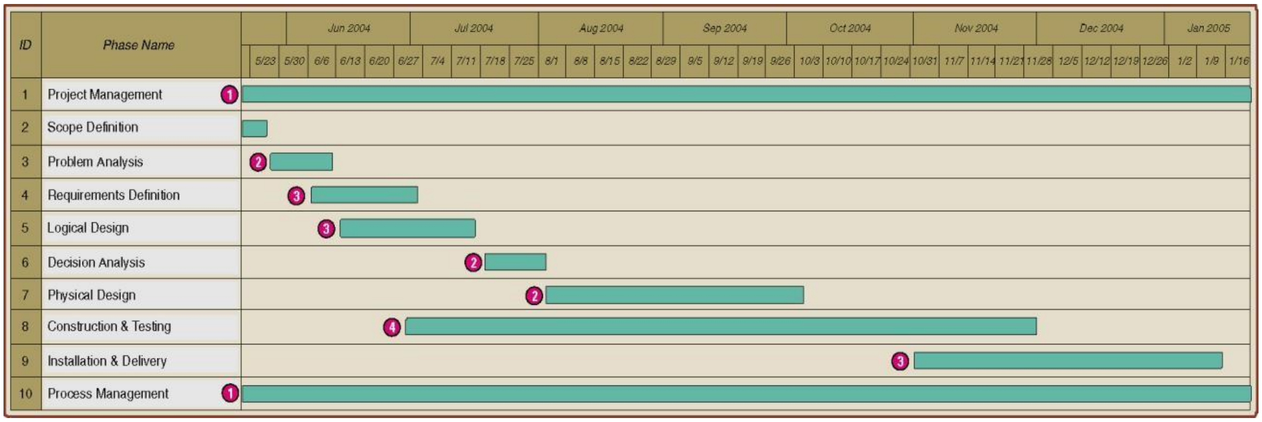
Principles of System Development

The following are the general principles that should underlie all systems development methodologies.

1. **Get the system users involved**
   - Without users/owner involvement, it is possible that the technological solutions don’t address the real organization problems.
   - Therefore, owner and user involvement are necessary to minimize organization or technical problems.
   - For successful systems development, system development should go along with user and owner’s participation.

2. **Use a problem-solving approach**
   - Problem in system development includes real problems, opportunities for improvement, and directives from management.
   - Using some sort of problem-solving approach to all projects increases efficiency and productivity in system development.
   - Classical Problem-solving approach
     1. Study and understand the problem, its context, and its impact.
     2. Define the requirements that must be met by any solution.
     3. Identify candidate solutions that fulfill the requirements, and select the “best” solution.
     4. Design and/or implement the chosen solution.
     5. Observe and evaluate the solution’s impact, and refine the solution accordingly.

3. **Establish phases and activities**
   - The classical system life cycle consists of four phases; System planning, Analysis, System design and System implementation
   - At the beginning of the development, early concerns are with the business and then, as development progresses, later concerns become more technology-driven.
   - Each phase can also be subdivided into smaller activities and tasks, which can be more easily managed and accomplished.



- Every project is different depending on the size, complexity, and development methodology or route. It’s important to note that the phases occur in parallel.

4. **Document through development**

 Documentation should be a working by-product of the entire systems development effort. Documentation should go parallel with the project.

5. **Establish standards**

- Establishing standards promote good communication between users and information systems professionals.
- Sometimes, users can be changed. With the standards and documentation, the project ensures consistent systems development.
- These standards describe activities, responsibilities, documentation guidelines or requirements and quality checks.

6. **Divide and conquer**

- Virtually all systems, which are part of super system, contain smaller systems (subsystems).
- Problem-solving process can be simplified by dividing this larger system into more easily managed pieces.

7. **Design systems for growth and change**

- Systems that are designed to meet only current requirements are difficult to modify in response to new requirements. However, it’s very important to build systems that can grow and change as requirements grow and change.
- **Entropy** is the term describing the natural and inevitable decay of all systems.
- System entropy can be managed. During the support phase, the analyst encounters the need for change directing the analyst and programmer to rework former phases of the life cycle.
- Eventually, the cost of maintenance exceeds the costs of starting over, the system has become obsolete

8. **Manage the Process and Projects**

- Most organizations have a system development process or methodology, but they do not always use it consistently on projects. Both the process and the projects that use it must be managed.
- **Process management** ensures that an organization's chosen process or management is used consistently on and across all projects.
- Process management also defines and improves the chosen process or methodology over time.
- **Process management** is an ongoing activity that documents, manages, oversees the use of, and improves an organization's chosen methodology (the "process") for system development. Process management is concerned with phases, activities, deliverables, and quality standards should be consistently applied to all projects.
- Project management ensures that an information system is developed at minimum cost, within a specified time frame, and with acceptable quality (using the standard system developmental process or methodology).
- **Project management** is the process of scoping, planning, staffing, organizing, directing, and controlling a project to develop an information system at a minimum cost, within a specified time frame, and with acceptable quality.
- Effective project management is essential to achieving CMM Level 2 success. CMM Level 2 deals with project management while CMM Level 3 deals with process management. Use of a repeatable process gets us to CMM Level 3. Most organizations pursuing the CMM are targeting Level 3; that is, consistently using a standardized process or methodology to develop all systems.
- CMM Levels 4 and 5 require effective process management.
- Project management can occur without a standard process, but in mature organizations all projects are based on a standardized and managed process.
- Process management and project management are influenced by the need for quality management.
- Quality standards are built into a process to ensure that the activities and deliverables of each phase will contribute to the development of a high-quality Information system. They reduce the likelihood of missed problem and requirements, as well as flawed designs and program errors (bugs). Standards also make the IT organization more agile. As personnel changes occur, staff can be relocated between projects with the assurance that every project is following an understood and accepted process.

9. **Justify Information Systems as Capital Investments**

- Information systems are capital investments, just like a fleet of trucks or a new building.
- System owners commit to this investment. Notice that the initial commitment occurs early in a project, when system owners agree to sponsor and fund the project. Later (during the phase called decision analysis), system owners recommit to the more costly technical decisions. In considering a capital investment, two issues must be addressed:

(1) For any problem, there are likely to be several possible solutions. The systems analyst and other stakeholders should not blindly accept the first solution suggested. The analyst who falls to look at alternatives may be doing the business a disservice.

(2) After identifying alternative solutions, the systems analyst should evaluate each possible solution for feasibility, especially for cost-effectiveness. Cost-effectiveness is measured using a technique called cost-benefit analysts.

- Like project and process management, cost-benefit analysis is performed throughout the system development process.
- A significant advantage of the phased approach to systems development is that it provides several opportunities to re-evaluate cost-effectiveness, risk, and feasibility.
- There is often a temptation to continue with a project only because of the investment already made. In the long num, cancelled projects are usually much less costly than implemented disasters. This is extremely important for young analysts to remember.
- Most system owners want more from their systems than they can afford or are willing to pay for. Furthermore, the scope of most information system projects increases as the analyst learns more about the business problems and requirements as the project progresses. Unfortunately, most analysts fail to adjust estimated costs and schedules as the scope increases. As a result, the analyst frequently and needlessly accepts responsibility for cost and schedule overruns.
- Because information systems are recognized as capital investments, system development projects are often driven by enterprise planning. Many contemporary information technology business *units* create and maintain a strategic information systems plan. Such a plan identifies and prioritizes Information system development projects.
- Ideally, a strategic information system plan is driven by a strategic enterprise plan that charts a course for the entire business.
  - **Cost-effectiveness** is the result obtained by striking a balance between the lifetime costs of developing, maintaining, and operating an information system and the benefits derived from that system. Cost-effectiveness is measured by a cost-benefit analysis.
  - **Strategic information systems plan** is a formal strategic plan (3-5 years) for building and improving an information technology infrastructure and the information system applications that use that infrastructure.
  - **Strategic enterprise plan** is a formal strategic plan (3-5 years) for an entire business that defines its mission, vision, goals, strategies, benchmarks, and measures of progress and achievement. Usually, the strategic enterprise plan is complemented by strategic business unit plans that define how each business unit will contribute to the enterprise plan. The information systems plan is one of those unit-level plans.

10. Don’t Be Afraid to Cancel or Revise Scope

- Don’t be afraid to cancel a project or revise scope regardless of how much money has been spent so far - cut your losses.
- **Creeping commitment approach** to systems development is advocated for. **Creeping commitment** is a strategy in which feasibility and risks are continuously reevaluated throughout a project. Project budgets and deadlines are adjusted accordingly.
- With the creeping commitment approach, multiple feasibility checkpoints are built into any systems development methodology. At each checkpoint feasibility is reassessed. All previously expended *costs* are considered sunk (meaning not recoverable). They are, therefore, irrelevant to the decision.

- Thus, the project should be re-evaluated at each checkpoint to determine if it remains feasible to continue investing time, effort, and resources into the project.
- At each checkpoint, the analyst should consider the following options:
  - Cancel the project if it is no longer feasible.
  - Re-evaluate and adjust the costs and schedule if project scope is to be increased.
  - Reduce the scope if the project budget and schedule are frozen and not sufficient to cover all project objectives.
- In addition to managing feasibility throughout the project, risks must be managed. Risk management seeks to balance risk and reward. **Risk management** is the process of identifying, evaluating, and controlling what might go wrong in a project before it becomes a threat to the successful completion of the project or implementation of the information system. Risk management is drive by risk analysis or assessment.
- Different organizations are more or less averse to risk, meaning that some are willing to rake greater risks than others in order to achieve greater rewards.

A Systems Development Process

- In this section we'll examine a logical process for systems development.

Where Systems Development Projects Come From?

- System owners and system users initiate most projects. The drive for most projects is some combination of problems, opportunities, and directives.
  - **A problem** is an undesirable situation that prevents the organization from fully achieving its purpose, goals, and/or objectives.
  - **An opportunity** is a chance to improve the organization even in the absence of an identified problem.
  - **A directive** is a new requirement that is imposed by management, government, or some external influence.

The PIECES Problem-Solving Framework

- **Problem solving** refers to solving problems, exploiting opportunities, and fulfilling directives
- James Wetherbe developed a useful framework for classifying problems called **PIECES** because the letters of each of the six categories, when put together, spell the word “PIECES".
- This is a checklist for identifying problems with an existing information system.

  P the need to improve **performance**
  I the need to improve **information** (and data)
  E the need to improve **economics**, control costs, or increase profits
  C the need to improve **control** or security
  E the need to improve **efficiency** of people and processes
  S the need to improve **service** to customers, suppliers, partners, employees, etc.
- **Performance** o Throughput o Response Time

- **Information (and Data)**
  - Outputs
    - Lack of any information
    - Lack of necessary information
    - Lack of relevant information
    - Too much information – information overload
    - Information that is not in a useful format
    - Information that is not accurate
    - Information that is difficult to produce
    - Information that is not timely to its subsequent use
  - Inputs
    - Data is not captured
    - Data is not captured in time to be useful
    - Data is not accurately captured – contains errors
    - Data is difficult to capture
    - Data us captured redundantly – same data is captured more than once
    - Too much data is captured
    - Illegal data is captured
  - Stored Data
    - Data is stored redundantly in multiple files and/or databases
    - Stored data is not accurate
    - Data is not secure from accident or vandalism
    - Data is not well organized
    - Data is not flexible – not easy to meet new information needs from stored data
    - Data is not accessible
- **Economics**
  - Costs
    - Costs are unknown
    - Costs are untraceable
    - Costs are too high
  - Profits
    - New markets can be explored
    - Current marketing can be improved
- **Control (and Security)**
  - Too little security or control
    - Input data is not adequately edited
    - Crimes (e.g., fraud, embezzlement) are (or can be) committed against the data
    - Ethics are breached on data or information – refers to data or information getting to unauthorized people
    - Redundantly stored data is inconsistent in different files or databases
    - Data privacy regulations or guidelines are being (or can be) violated

        - Processing errors are occurring (either by people, machines, or software)
        - Decision- making errors are occurring o Too much control or security
        - Bureaucratic red tape slows the system
        - Controls inconvenience customers or employees
        - Excessive controls cause processing delays
- **Efficiency**
    - People, machines, or computers waste time
        - Data is redundantly input or copied
        - Data is redundantly processed
        - Information is redundantly generated
    - People, machines, or computers waste materials and suppliers
        - Effort required for tasks is excessive
        - Materials required for tasks is excessive
- **Service**
    - The system produces inaccurate results
    - The system produces inconsistent results
    - The system produces unreliable results
    - The system is not easy to learn
    - The system is not easy to use
    - The system is awkward to use
    - The system is inflexible to new or exceptional situations
    - The system is inflexible to change
    - The system is incompatible with other systems
    - The system is not coordinated with other systems

.

Planned vs Unplanned Projects

Projects can be either planned or unplanned.

- A planned project is the result of one of the following:
    - An **information systems strategy plan** has examined the business as a whole to identify those system development projects that will return the greatest strategic (long-term) value to the business.
    - A **business process redesign** has thoroughly analyzed a series of business processes to eliminate redundancy and bureaucracy and to improve efficiency and value added. Not it is time to redesign the supporting information system for those redesigned business processes.
- Unplanned projects are those that are triggered by a specific problem, opportunity, or directive that occurs in the course of doing business.
- Most Organizations have many unplanned projects. Anyone can submit a proposed project based on something that is happening in the business. The number of unplanned project proposals can easily overwhelm the largest information systems organization; therefore, they are frequently screened and prioritized by a steering committee of system owners and IT managers to determine which requests get approved. Those requests that are not approved are backlogged until resources become available (which sometimes never happen).

  - **Steering committee** is an administrative body of system owners and information technology executives that prioritizes and approves candidate system development projects.
  - **Backlog** is a repository of project proposals that cannot be funded or staffed because they are a lower priority than those that have been approved for system development.
- Both planned and unplanned projects go through the same essential system development process.

System Development Life Cycle

By the end of the topic, you should be able to:

- Describe the essential phases of system development. For each phase, describe its purpose, inputs, and outputs.
- Describe cross life cycle activities that overlap multiple system development phases.
- Describe various system development methodologies • Describe various automated tools for system development.

System Development

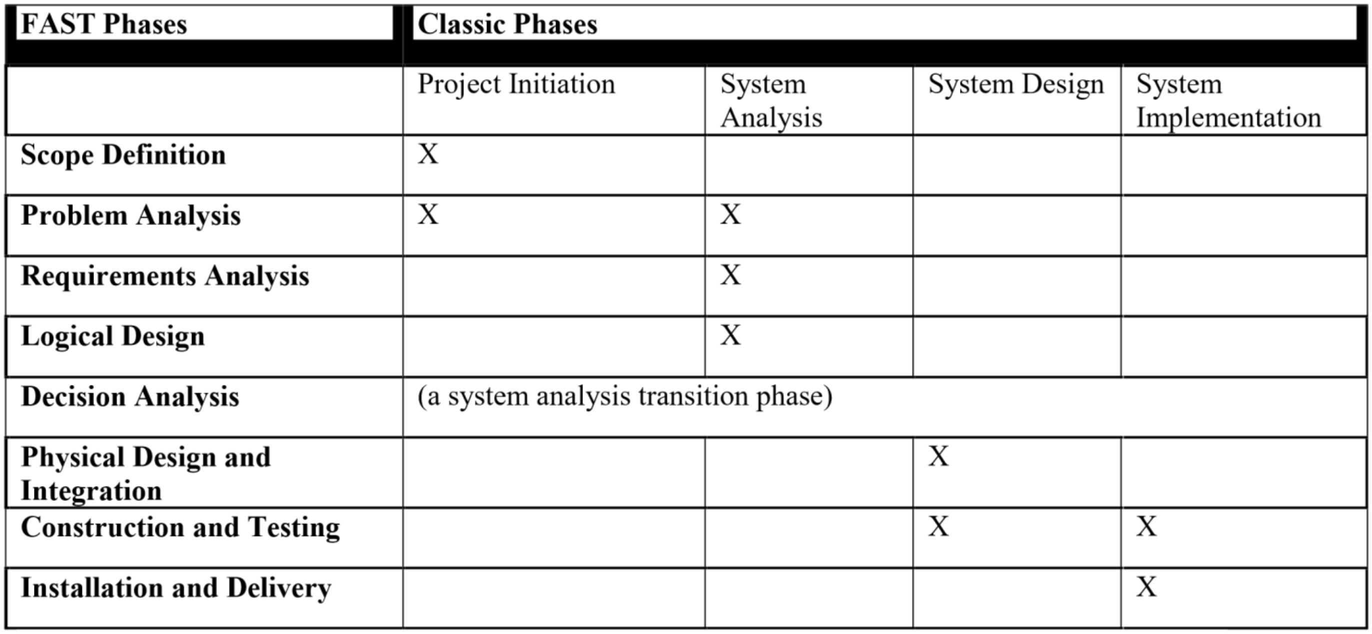
- Systems development refers to the process of examining a business situation with the intent of improving it through better procedures and methods.
- Systems development can generally be thought of as having two major components: systems analysis and systems design.
- Systems design is the process of planning a new business system or one to replace or complement an existing system. But before this planning can be done, one must thoroughly understand the old system and determine how computers can be used (if at all) to make its operation more effective.

System Development Life Cycle

- System life cycle is an organizational process of developing and maintaining information systems.
- The systems development life cycle (SDLC) is a problem-solving process that describes a set of steps that produces a new information system. Each step in the process delineates a number of activities.
- The system development life cycle method can be thought of as a set of activities that analysts, designers and users carry out to develop and implement an information system.
- Performing these activities in the order prescribed by the SDLC will bring about a solution to the business situation.
- The process of system development starts when management or sometimes system development personnel realize that a particular business system needs improvement.
- System life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle.

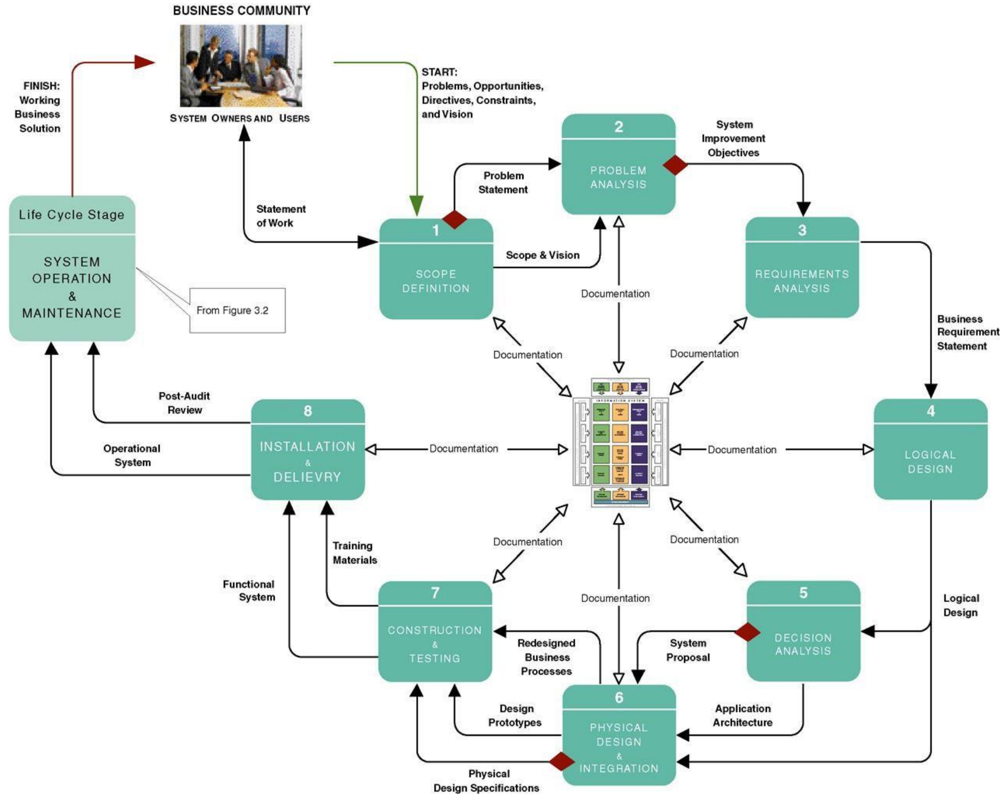
FAST (Framework for the Application of Systems Thinking) Phases

- There are various systems development methodologies defined and designed which are employed during development process of system. These approaches are also referred as "System Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development.
- FAST like most methodologies, consists of phases. The number of phases will vary from one methodology to another.
- The FAST methodology employs eight phases to better define periodic milestones and the deliverables. The final deliverable of the methodology is the production system.



FAST Phases	Classic Phases			
	Project Initiation	System Analysis	System Design	System Implementation
Scope Definition	X			
Problem Analysis	X	X		
Requirements Analysis		X		
Logical Design		X		
Decision Analysis	(a system analysis transition phase)			
Physical Design and Integration			X	
Construction and Testing			X	X
Installation and Delivery				X

- The figure below illustrates the phases of the FAST methodology. Each phase produces deliverables that are passed to the next phase. And documentation accumulates as you complete each phase. Iconic representation of the building blocks to symbolize this accumulation of knowledge and work-in-process artifacts during system development.
- Notice that a project starts with some combination of PROBLEMS, OPPORTUINITIES, and DIRECTIVES from the user community (the green arrow) and finishes with a WORKING BUSINESS SOLUTION (the red arrow) for the user community.



- *FAST* is a repository-based methodology. This means that phases (and activities included in phases) communicate across a shared repository. Thus, the phases and activities are not really sequential. Work in one phase can and should overlap work in another phase, so long as the necessary information is already in the repository. This accelerates development and allows *FAST* to live up to its name. Furthermore, this model permits the developer to backtrack when an error or omission is discovered.

Scope Definition Phase

- The purpose of the scope definition phase is twofold:
  - It answers the question, "Is this problem worth looking at?
  - Assuming the problem worth looking at**,** establish the size and boundaries of the project, the project vision, any constraints or limitations, the required project participants, and finally, the budget and schedule.
- This phase includes OWNERS, PROJECT MANAGERS, and SYSTEMS ANALYSTS. System users are generally excluded because it is too early to get into the level of detail they will eventually bring to the project.
- The scope definition phase is triggered by some combination of PROBLEMS, OPPORTUNITIES and DIRECTIVES
- There are several deliverables or outcomes of a scope definition
  - One important outcome is a PROBLEM STATEMENT, a concise overview of the problems, opportunities, and/or directives that triggered the project. The PIECES framework provides an excellent outline for a problem statement.

  - **Problem statement** is a statement and categorization of problems, opportunities, and directives; may also include constraints and an initial vision for the solution. Synonyms include *preliminary study* and *feasibility assessment*.
- The goal here is not to solve the problems, opportunities, and directives but only to catalogue and categorize them.
- We should also identify any constraints that may impact the proposed project. Examples of constraints include budget limits, deadlines, human resources available or not available, business policies or government regulations and technology standards.
- Finally, the system owners should be asked for at least a high-level vision for the system improvements they are seeking.
- Given a basic understanding of problems, opportunities, directives, constraints and vision, there is need to establish initial scope. Thus, an initial SCOPE STATEMENT is another important outcome of this phase.
- Scope defines how big we think the project is. Your Information system building blocks provide a useful framework for defining scope.
- Scope can and frequently does change during a project. But by documenting initial scope, you establish a baseline for controlling scope creep on both the budget and the schedule. Scope creep is a common phenomenon wherein the requirements and expectations of a project increase, often without regard to the impact on budget and schedule.
- Given the Initial problem, scope statements for the project, the analyst can staff the project team, estimate the budget for system development, and prepare a schedule for the remaining phases. Ultimately this phase concludes with a “go or no go" decision from system owners. Either the system owners agree with the proposed scope, budget and schedule for the project or they must reduce scope (to reduce costs and time) or cancel the project.
- The final and most important deliverable is a **STATEMENT OF WORK**. A statement of work is a contract or agreement to develop the information system. It consolidates the problem statement, scope statement and schedule and budget for all parties who will be involved in the project.

Problem Analysis Phase

- There is always an existing system, regardless of whether it currently uses Information technology.
- The participants include the SYSTEM OWNERS however this phase begins to actively involve the SYSTEM USERS. The system users are the business subject matter experts in any project. The system users’ perspectives overlap many phases. Remember principle 1: “Get the system users Involved.". The PROJECT MANAGERS and SYSTEM ANA£YSTS are always involved in all phases of a project
- The prerequisites for the problem analysis phase are the SCOPE and PROBLEM STATEMENTS as defined and approved in the scope definition phase.
- The deliverable of the problem analysis phase is a set of SYSTEM IMPROVEMENT OBJECTIVES derived from a thorough understanding of the business problems. These objectives do not define inputs, outputs, or processes. Instead, they define the business criteria on which any new system will be evaluated. For instance. we might define a system improvement objective as any of the following:

  - o Reduce the time between order processing and shipping by three days. o Reduce bad credit losses by 45 percent.
  - o Comply with new financial aid federal qualification requirements by January 1.
- Think of system improvement objectives as the gratifying criteria for evaluating any new system that you might eventually design and implement.
- System improvement objectives may be presented to system owners and users as a written recommendation or an oral presentation.
- Depending on the complexly of the problem and the project schedule, the team may or may not choose to formally document the existing system. Such documentation frequently occurs when the business processes are considered dated or overly bureaucratic. Documentation of the existing system is sometimes called an “AS IS” BUSINESS Model. The as-is model may be accompanied by analysis demonstrating inefficiencies, bottlenecks, or other problems related to the business processes.
- Every existing system has its own terminology, history, culture, and nuances. Learning those aspects of the system is an important by-product of this phase. From all of the information gathered, the project team gains a better understanding of the existing system's problems and opportunities. After reviewing the findings, the system owners will either agree or disagree with the recommended system improvement objectives.
- And consistent with the creeping commitment principle, another go or no-go feasibility checkpoint is usually at the end of the phase. The project can be either:
  - − Cancelled if the problems are deemed no longer worth solving.
  - − Approved to continue to the next phase.
  - − Reduced or expanded in scope (with budget and schedule modifications) and then approved to continue to the next phase.

Requirements Analysis Phase

- This phase is concerned with
  - − What capabilities should the new system provide for its users?
  - − What data must be captured and stored?
  - − What performance level is expected?
- This requires decisions about what the system must do, not how it should do those things. □ The REQUIREMENT ANALYSIS phase defines and prioritizes the business requirements. The analyst approaches the users to find out what they need or want out of the new system, carefully avoiding any discussion of technology or technical implementation.
- This is perhaps the most important phase of systems development. Errors and omissions in requirements analysis result in user dissatisfaction with the final system and costly modifications.
- The participants primarily include SYSTEM USERS (which may include owners who will actually use the system), SYSTEMS ANALTSTS and PROJECT MANAGERS SYSTEM DESIGNERS are omitted from this phase in order to prevent premature attention to technology solutions.
- The building blocks can themselves provide the framework for defining many business requirements, including BUSINESS DATA REQUIREMENTS, BUSINESS PROCESS

REQUIREMENTS and BUSINESS AND SYSTEM INTERFACE REQUIREMENTS. Because the business requirements are intended to solve problems, the PIECES framework can also provide a useful outline for a requirements statement.

- The IMPROVEMENT OBJECTIVES from the problem analysis phase are the prerequisites to the requirement analysis phase. The deliverable is a BUSINES REQUIREMENT STATEMENT. Again, this requirements statement does not specify any technical possibilities or solutions. The requirements statement may be a document as small as a few pages, or it may be extensive with a page or more of documentation per requirement.
- To produce a business requirements statement, the systems analyst works closely with system users to identify needs and priorities. This Information is collected by way of Interviews, questionnaires, and facilitated meetings.
- The challenge to the team is to validate those requirements. The system improvement objectives provide the “grading key" for business requirements: Does each requirement contribute to meeting one or more system improvement objectives
- Typically, requirements must also be prioritized. Priorities serve two purposes. First, if project timeline become stressed, requirements priorities can be used to re-scope the project. Second, priorities can frequently be used to define iterations of design and construction to create staged releases or versions of the final product
- The requirements analysis phase should never be skipped or short-changed. One of the most common complaints about new systems and applications is that they don't really satisfy the users' needs. This usually happens where system designers and builders become preoccupied with a technical solution before fully understanding the business needs. System designers and builders are dependent on competent systems analysts to work with users to define and document complete and accurate business requirements before applying any technology.

The Logical Design Phase

- Logical design is the translation of business user requirements into a system model that depicts only the business requirements and not any possible technical design or implementation of those requirements.
- Business requirements are usually expressed in words. Systems analysts have found it useful to translate those words into pictures called system models to validate the requirements for completeness and consistency. System model is a picture of a system that represents reality or a desired reality.
- System models facilitate improved communication between system users, system analysts, system designers, and system builders.
- The LOGICAL DESIGN PHASE translates business requirements into system models. The term logical design should be Interpreted as "technology independent; meaning the pictures illustrates the system independent of any possible technical solution- hence, they model business requirements that must be fulfilled by any technical solution we might want to consider.
- Different methodologies require or recommend different amounts and degrees of system modelling or logical design.

- The participants include SYSTEM ANALYSTS (who draw the models) and system USERS (who validate the models). PROJECT MANAGERS are always included to ensure that modelling meets standards and does not deter overall project progress. We can draw
  (1) LOGICAL DATA MODELS that depict data and information requirements,
  (2) LOGICAL PROCESS MODELS that depict business processes requirements, and (3) LOGICAL INTERFACE MODELS that depict business and system Interface requirements.
- The deliverables of logical design are the LOGICAL SYSTEM MODELS and SPECIFICATIONS.
- We should note that the SCOPE DEFINITION, PROBLEM ANALYSIS, REQUIREMENT ANALYSIS, and LOGICAL DESIGN phases are collectively recognized by most experts as **system analysis**.

Decision Analysis Phase

- Given business requirements and the logical system models, there are usually numerous alternative ways to design a new information system to fulfil those requirements. Some of the pertinent questions include the following:
  1. How much of the system should be automated with information technology?
  2. Should we purchase software or built it ourselves (called the make-versus buy decision)?
  3. Should we design the system for an internal network, or should we design a Web based solution?
  4. Which information technologies (possibly emerging) might be useful for this application?
- These questions are answered in the DECISION ANALYSIS phase of the methodology
- The purpose of this phase is to:
  1. Identify candidate technical solutions
  2. Analyze those candidate solutions for feasibility, and
  3. Recommend a candidate system as the target solution to be designed.
- Candidate solutions evaluated in terms of:
  - **Technical feasibility** – Is the solution technically practical? Does our staff have the technical expertise to design and build this solution?
  - **Operational feasibility** – Will the solution fulfill the users' requirements? To what degree? How will the solution change the users' work environment? How do users feel about such a solution?
  - **Economic feasibility** – Is the solution cost-effective?
  - **Schedule feasibility** – Can the solution be designed and implemented within an acceptable time?
  - **Risk feasibility** – What is the probability of a successful implementation using the technology and approach?

Physical Design & Integration Phase

- **Physical design** is the translation of business user requirements into a system model that depicts a technical implementation of the users' business requirements.

- The purpose of the design phase is to transform the business requirements from the definition phase into a set of technical design blueprints for construction.
- Two extreme philosophies of physical design include:
  - *Design by specification* – physical system models and detailed specification are produced as a series of written (or computer-generated) blueprints for construction.
  - *Design by prototyping* – Incomplete but functioning applications or subsystems (called *prototypes*) are constructed and refined based on feedback from users and other designers.
- This phase is to design the new system and to integrate the new system with the existing system. It specifies the technical requirements for the target solution □ FAST encourages an iterative design and construct strategy.
- System analysts, database specialists, network specialists, computer specialists are involved in this design phase.
- Two triggers are business requirements and design requirements. This views the system from the perspectives of the system designers.
- These days, the design and construction phases are merged together. The basic idea of RAD is to actively involve system users in the design process, to accelerate the definition/design/construction process by catching errors and omissions earlier in the process, and to reduce the amount of time that passes before the users begin to see a working system.
- The final deliverable is a **technical set of design specifications**. These specifications can take several forms, but the most common approach is modelling.
- General design models will picture the structure of the database, the structure of the overall application, the overall look and feel of the user interface, the structure of the computer network and the design structures for any completes software to be written.

The Construction and Testing Phase

- Design specification is the key input to the construction phase.
- After several iterations of the design/construction loop that implements rapid application development, functional system is implemented.
- The purpose of this phase is to build and test a functional system that fulfils business and design requirements and to implement the interfaces between the new system and existing production systems.
- The phase builds and tests the actual solution or interim prototypes of the solution.
- Important aspect of application programming is testing.
  - **Unit tests**: ensure that the applications programs work properly when tested in isolation from other applications programs.
  - **System tests**: ensure that applications programs written in isolation work properly when they are integrated into the total system.
- The final deliverable is the functional system

.

The Installation and Delivery Phase

- Functional key information system is the input to the delivery phase.
- This purpose of this phase is to install, distribute and place the new system into operation or production.

- Every information workers are active in this phase.
- Tests are conducted to ensure that the new system works properly.
- The final deliverable of the phase is the **production system** for the system users. The other outputs are **training** and **supports**.
- **System support** is an ongoing maintenance of a system after it has been placed into operation. This includes program maintenance and system improvements. These activities include fixing software bugs, recovering the system, assisting users and adapting the system to new requirements

Installation and Delivery Phase

- Deliver the system into operation (production)
- Deliver User training
- Deliver completed documentation
- Convert existing data

System Operation & Maintenance

System support is the ongoing technical support for users of a system, as well as the maintenance required to deal with any errors, omissions, or new requirements that may arise.

Cross lifecycle activities

- These are activities that overlap different phases of the methodology.
- They are influenced or performed in conjunction with several phases.
- Cross life cycle activities include fact-finding, documentation and presentations, estimation and measurement, feasibility analysis and project and process management.

Fact-finding

- Information about the systems, requirements, and preferences is collected using research, interviews, meetings, questionnaires, sampling, and other techniques.
- Important activities performed on survey, study, and definition phases of FAST projects.
- Project team learns about the businesses and system's vocabulary, problems, opportunities, constraints, requirements, and priorities.

Documentation and Presentations

- Documentation and Presentation are common communication skills for successful project completion.
- **Documentation** is an activity of recording facts and specifications for a system.
- **Presentation** is written or verbal activity of formally packaging documentation for review by interested users and managers. These activities have to be supported by all the phases in FAST.

Estimation and measurement

- These are commonly performed to address the quality and productivity of systems.
- **Estimation** is an activity of approximating the time, effort, costs, and benefits of developing systems.
- **Measurement** is an activity of measuring and analyzing developer productivity and quality.
- Software and systems metrics provide a guide of techniques and tools that can both simplify the estimation process and provide a statistical database of estimates versus performance.

Feasibility Analysis

- Feasibility analysis is performed several times in creeping commitment approach in system development life cycle.
- **Feasibility** is a measure of how beneficial the development of an information system would be to an organization.

Project and Process Management

- Project management is the ongoing activity, by which an analyst plans, delegates, directs and controls progress to develop an acceptable system within the allotted time and budget.
- Similarly, process management is ongoing activity that establishes standards for activities, methods, tools, and deliverables of the life cycles.

Computer-Aided Systems Engineering (CASE)

- CASE is the application of information technology to systems development activities, techniques, and methodologies.
- CASE tools are software that automates or support one or more phases of a systems development life cycle. This accelerates the process of developing systems and to improve the quality of the system.
- CASE is not the methodology but it is a technology supporting a methodology.
- There are various CASE tools:
  - **A case Tool Framework Upper-Case**: tools support the survey, study, definition, and design phases of the methodology.
  - **Lower-Case:** tools support the design, construction, and delivery phases of the methodology (Power builder). There can be overlap between upper- and lower-case tools.

Case Tool Architecture

- CASE tools are built around an automated repository. Around that repository is a collection of tools or facilities to create documentation or other system components. CASE tools are able to use and update some other tool’s repository.
- **CASE repository** is a developers’ database. It is a place where the developers can store diagrams, descriptions, specifications, and other by-products of systems development.

Facilities and Functions

- Representative CASE tools provide some of the following facilities:
  - diagramming tools
  - description tools
  - prototyping tools
  - inquiry and reporting tools
  - quality management tools
  - decision support tools
  - documentation organization tools
  - design generation tools
  - code generator tools
  - testing tools
  - data sharing tools
  - version control tools and
  - housekeeping tools

Automated Tools and Technology

These include:

- Computer-aided systems engineering (CASE)
- Application development environments (ADEs)
- Process and project managers

Computer-aided systems engineering (CASE)

- **Computer-aided systems engineering** (CASE) is automated software tools that support the drawing and analysis of system models and associated specifications. Some CASE tools also provide prototyping and code generation capabilities.
  - **CASE repository** is system developers' database where developers can store system models, detailed descriptions and specifications, and other products of system development. Synonyms: *dictionary* and *encyclopedia*.
  - **Forward engineering** is CASE tool capability that can generate initial software or database code directly from system.
  - **Reverse engineering** is CASE tool capability that can generate initial system models from software or database code.

Application Development Environments

- **An application development environment (ADEs)** is an integrated software development tool that provides all the facilities necessary to develop new application software with maximum speed and quality. A common synonym is *integrated development environment* (IDE). ADE facilities may include:
  - Programming languages or interpreters

- Interface construction tools
- Middleware
- Testing tools
- Version control tools
- Help authoring tools
- Repository links

Process and Project Managers

- **Process manager application** is an automated tool that helps document and manage a methodology and routes, its deliverables, and quality management standards. An emerging synonym is *method ware*.
- **Project manager application** is an automated tool to help plan system development activities (preferably using the approved methodology), estimate and assign resources (including people and costs), schedule activities and resources, monitor progress against schedule and budget, control and modify schedule and resources, and report project progress.

Software Development Methodologies (Models)

- A system development methodology refers to the framework that is used to structure, plan, and control the process of developing an information system.
- A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses.
- One system development methodology is not necessarily suitable for use by all projects. Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations.
- The models specify the various stages of the process and the order in which they are carried out.
- The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of planned testing, influence regression testing and largely determines which test techniques to use.
- There are various Software development models or methodologies. They are as follows:
  1. Waterfall model
  2. Prototype model
  3. V model
  4. Incremental model
  5. RAD model
  6. Agile model
  7. Iterative model
  8. Spiral model

Waterfall Model

- The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.

- It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin.
- This type of model is suitable for a project which is small and there are no uncertain requirements.
- At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model the testing starts only after the development is complete.
- In **waterfall model phases** do not overlap.
- A number of variants of this model exist, with each one quoting slightly different labels for the various stages. In general, however, the model may be considered as having six distinct phases.

Waterfall Model Phases

Requirement's analysis phase

- This phase involves gathering information about what the users' needs and defining in the clearest possible terms the problem that the new system is expected to solve. Analysis includes understanding the users' needs and constraints, the functions the new system must perform, the performance levels it must adhere to, and the external systems it must be compatible with. Some of the techniques used to obtain this information include interviews, questionnaires and observations.
- The output of this phase is **formal requirements specification**, which serves as input to the next step.

Design phase

- The task in this phase is defining the hardware and software architecture, components, modules, interfaces, and data to satisfy specified requirements.
- It involves defining the hardware and software architecture, specifying performance and security parameters, designing data storage containers and constraints, choosing a programming language, and indicating strategies to deal with issues such as exception handling, resource management and interface connectivity.
- This is also the stage at which user interface design is addressed, including issues relating to navigation and accessibility.
- The output of this stage is one or more **design specifications**, which are used in the next stage of coding.

Coding phase

- This stage involves the actual construction of the system as per the design specification(s) developed in the previous stage.
- This step is performed by a development team consisting of programmers, interface designers and other specialists, using tools such as compilers, debuggers, interpreters and media editors.
- The output of this phase is one or more **system components**, built according to a predefined coding standard and debugged, tested and integrated to satisfy the system design requirements.

Testing phase

- In this stage, both individual components and the integrated whole are tested to ensure that they are error-free and fully meet the requirements outlined in the requirements stage. An independent quality assurance team defines "test cases" to evaluate whether the product fully or partially satisfies the requirements outlined in the first stage.
- Defects, if found, are logged and feedback provided to the development team to enable correction. This is also the stage at which system documentation such as a user manual is prepared, reviewed and published.

Implementation phase

- This step is undertaken once the system has been tested and certified as fit for use. It involves preparing the system for installation and use at the user site.

Maintenance phase

- This step takes place after installation. It involves making modifications to the system or an individual component to alter attributes or improve performance. These modifications arise either due to change requests initiated by the user, or defects uncovered during live use of the system.
- Every change made to the system during the maintenance cycle is recorded and a new product release (called a "maintenance release" and exhibiting an updated revision number) is performed to enable the user to gain the benefit of the update.

Basic Principles of Waterfall Model

1. Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
2. Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
3. Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

Advantages of waterfall model

1. This model is simple and easy to understand and use.
2. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
3. In this model phases are processed and completed one at a time. Phases do not overlap.
4. Waterfall model works well for smaller projects where requirements are very well understood.
5. Ideal for supporting less experienced project teams and project managers, or project teams whose composition fluctuates.
6. The orderly sequence of development steps and strict controls for ensuring the adequacy of documentation and design reviews helps ensure the quality, reliability, and maintainability of the developed software.
7. Progress of system development is measurable.
8. Conserves resources.

Disadvantages of waterfall model

1. Inflexible, slow, costly and cumbersome due to significant structure and tight controls.
2. Project progresses forward, with only slight movement backward.
3. Little room for use of iteration, which can reduce manageability if used.
4. Depends upon early identification and specification of requirements, yet users may not be able to clearly define what they need early in the project.
5. Requirements inconsistencies, missing system components, and unexpected development needs are often discovered during design and coding.
6. Problems are often not discovered until system testing. Once an application is in the testing stage, it is very difficult to go back and change something that was not well thought out in the concept stage.
7. System performance cannot be tested until the system is almost fully coded, and under capacity may be difficult to correct.
8. Difficult to respond to changes. Changes that occur later in the life cycle are more costly and are thus discouraged.
9. Produces excessive documentation and keeping it updated as the project progresses is time consuming.
10. Written specifications are often difficult for users to read and thoroughly appreciate.
11. Promotes the gap between users and developers with clear division of responsibility.
12. Very little user interaction action is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.
13. Not suitable for the projects where requirements are at a moderate to high risk of changing.
14. In case of any failure once the system has been deployed then the costs of fixing such issues are very high, because of the need to update all the stages.

Situations where most appropriate

1. Project is for development of a mainframe-based or transaction-oriented batch system.
2. Project is large, expensive, and complicated.
3. Project has clear objectives and solution.
4. Pressure does not exist for immediate implementation.
5. Project requirements can be stated unambiguously and comprehensively.
6. Project requirements are stable or unchanging during the system development life cycle.
7. Users are fully knowledgeable in the business and application.
8. Team members may be inexperienced.
9. Team composition is unstable and expected to fluctuate.
10. Project manager may not be fully experienced.
11. Resources need to be conserved.
12. Strict requirement exists for formal approvals at designated milestones.

Situations where least appropriate

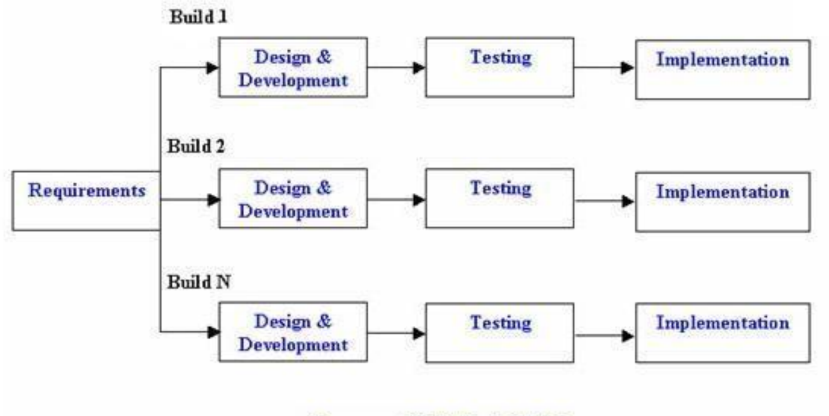
1. Large projects where the requirements are not well understood or are changing for any reasons such as external changes, changing expectations, budget changes or rapidly changing technology.
2. Web Information Systems (WIS) primarily due to the pressure of implementing a WIS project quickly; the continual evolution of the project requirements; the need for experienced, flexible

team members drawn from multiple disciplines; and the inability to make assumptions regarding the users' knowledge level.

3. Real-time systems.
4. Event-driven systems.
5. Leading-edge applications

Incremental Model

- In incremental model the whole requirement is divided into various builds.
- Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle.
- Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, coding and testing phases.
- A working version of software is produced during the first module, so you have working software early on during the software life cycle.
- Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.



Incremental Life Cycle Model

Advantages of Incremental model

1. Generates working software quickly and early during the software life cycle.
2. This model is more flexible – less costly to change scope and requirements.
3. It is easier to test and debug during a smaller iteration.
4. In this model users can respond to each built.
5. Lowers initial delivery cost.
6. Easier to manage risk because risky pieces are identified and handled during its iteration.

Disadvantages of Incremental model 1. Needs good planning and design.

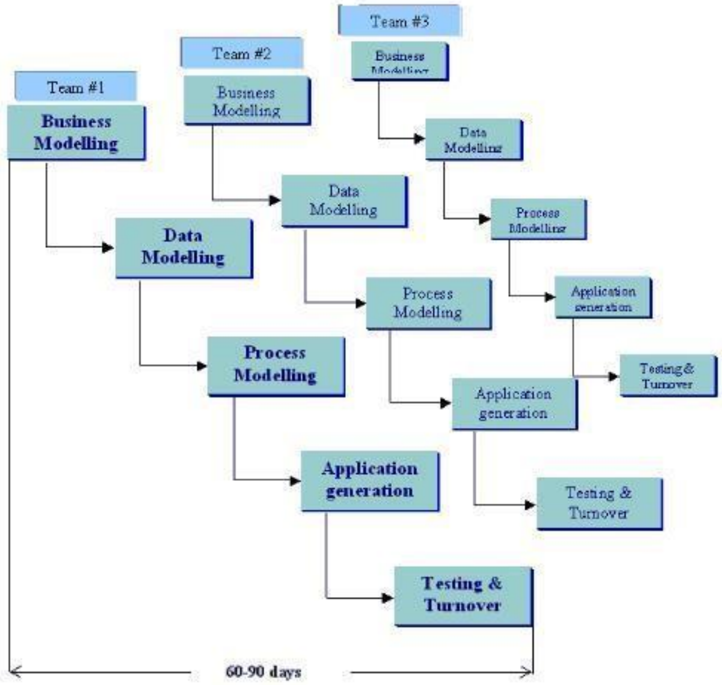
2. Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
3. Total cost is higher than waterfall.

When to use the Incremental model

1. This model can be used when the requirements of the complete system are clearly defined and understood.
2. Major requirements must be defined; however, some details can evolve with time.
3. There is a need to get software to the market early.
4. A new technology is being used
5. Resources with needed skill set are not available
6. There are some high-risk features and goals.

Rapid Application Development (RAD) model

- RAD model 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 user something to see and use and to provide feedback regarding the delivery and their requirements.



The phases in the rapid application development (RAD) model

Business modelling phase

□ The information flow is identified between various functions.

Data modelling

□ Information gathered from business modelling is used to define data objects that are needed for the business.

Process modelling

- Data objects defined in data modelling are converted to achieve the business information flow to achieve some specific business objective.
- Description is identified and created.

Application generation

- Automated tools are used to convert process models into code and the actual system.

Testing and turnover

- Test new components and all the interfaces.

Advantages of the RAD model

1. Reduced development time.
2. Increases reusability of components
3. Quick initial reviews occur
4. Encourages user feedback
5. Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model

1. Depends on strong team and individual performances for identifying business requirements.
2. Only system that can be modularized can be built using RAD
3. Requires highly skilled developers/designers.
4. High dependency on modelling skills
5. Inapplicable to cheaper projects as cost of modelling and automated code generation is very high.

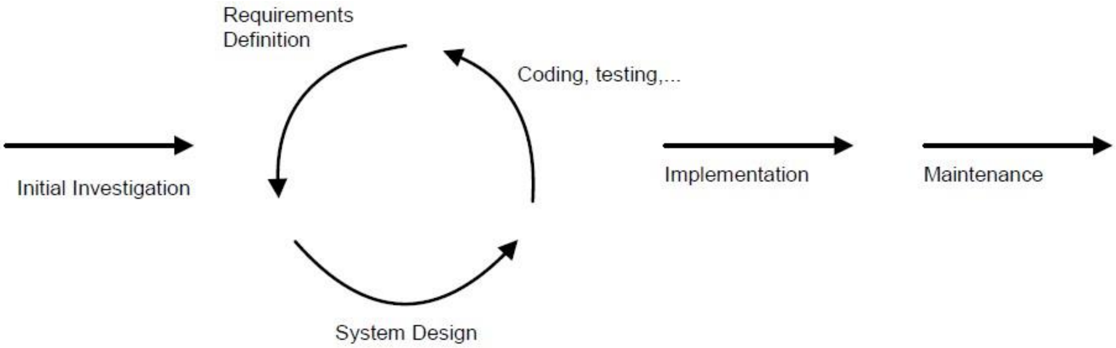
When to use RAD model

1. RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
2. It should be used if there's high availability of designers for modelling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
3. 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).

Prototype Model

- **Software prototyping** is an activity during certain software development. It is the creation of incomplete versions (prototypes) of the software program being developed.
- A prototype typically simulates only a few aspects of the features of the eventual program, and may be completely different from the eventual implementation.
- The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements.
- By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.

- Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.
- The prototypes are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.



The process of prototyping involves the following steps

1. **Identify basic requirements:** Determine basic requirements including the input and output information desired.
2. **Develop Initial Prototype**: The initial prototype is developed that includes only user interfaces.
3. **Review**: The customers, including end-users, examine the prototype and provide feedback on additions or changes.
4. **Revise and Enhance the Prototype:** Using the feedback both the specifications and the prototype can be improved. Negotiation about what is within the scope of the software may be necessary. If changes are introduced then a repeat of steps #3 and #4 may be needed.

Advantages of Prototype model

1. Users are actively involved in the development
2. The software designer and implementer can obtain feedback from the users early in the project.
3. The users and the developers can compare if the software made matches the software specification, according to which the software program is built.
4. It also allows the analyst some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met.
5. Prototyping can also avoid the great expense and difficulty of changing a finished software product.
6. Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
7. Errors can be detected much earlier.
8. Quicker user feedback is available leading to better solutions.
9. Missing functionality can be identified easily
10. Confusing or difficult functions can be identified
11. Requirements validation, Quick implementation of, incomplete, but functional, application.

Disadvantages of Prototype model

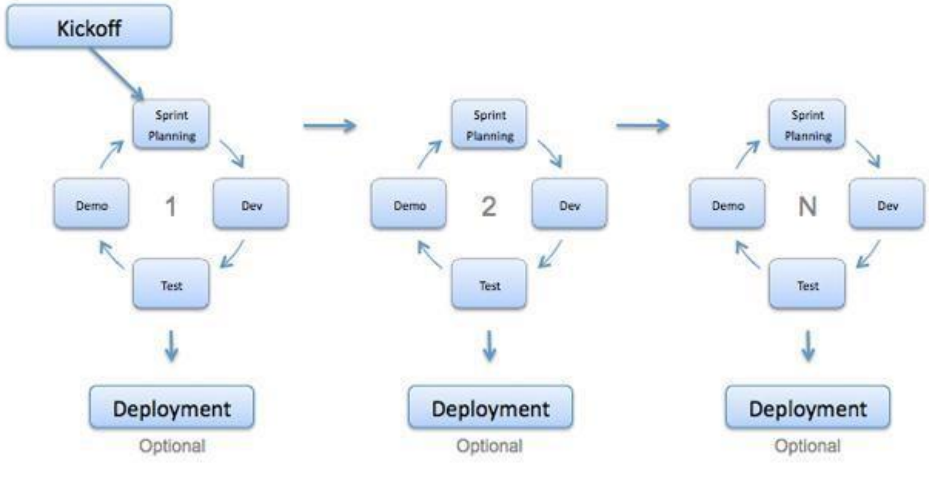
1. Leads to implementing and then repairing way of building systems.
2. Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
3. Incomplete application may cause application not to be used as the full system was designed
4. Incomplete or inadequate problem analysis.

When to use Prototype model

1. Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
2. Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
3. Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

Agile Development Model

- Agile development model is also a type of Incremental model.
- Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality.
- Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications.
- Extreme Programming (XP) is currently one of the most well-known agile development life cycle model.



Advantages of Agile model

1. User satisfaction by rapid, continuous delivery of useful software.
2. People and interactions are emphasized rather than process and tools. Users, developers and testers constantly interact with each other.

3. Working software is delivered frequently.
4. Face-to-face conversation is the best form of communication.
5. Close daily cooperation between business people and developers.
6. Continuous attention to technical excellence and good design.
7. Regular adaptation to changing circumstances.
8. Even late changes in requirements are welcomed

Disadvantages of Agile model

1. In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
2. There is lack of emphasis on necessary designing and documentation.
3. The project can easily get taken off track if the users' representative is not clear what final outcome that they want.
4. Only senior programmers are capable of taking the kind of decisions required during the development process.

When to use Agile model

1. When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
2. To implement a new feature the developers, need to lose only the work of a few days, or even only hours, to roll back and implement it.
3. Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
4. Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

An Iterative Life Cycle Model

- An iterative life cycle model does not attempt to start with a full specification of requirements.
- Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

Advantages of Iterative model

1. In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on, we can design and built a skeleton version of that, and then evolved the design based on what had been built.

2. In iterative model we are building and improving the product step by step. Hence, we can track the defects at early stages. This avoids the downward flow of the defects.
3. In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
4. In iterative model less time is spent on documenting and more time is given for designing.

Disadvantages of Iterative model

1. Each phase of an iteration is rigid with no overlaps
2. Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle

When to use iterative model

1. Requirements of the complete system are clearly defined and understood.
2. When the project is big.
3. Major requirements must be defined; however, some details can evolve with time.

Spiral Model

- The spiral model is similar to the incremental model, with more emphasis placed on risk analysis.
- The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model).
- The baseline spiral starts in the planning phase where requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral.

Phases in Spiral model

Planning Phase

□ Business requirements are gathered during the planning phase.

Risk Analysis

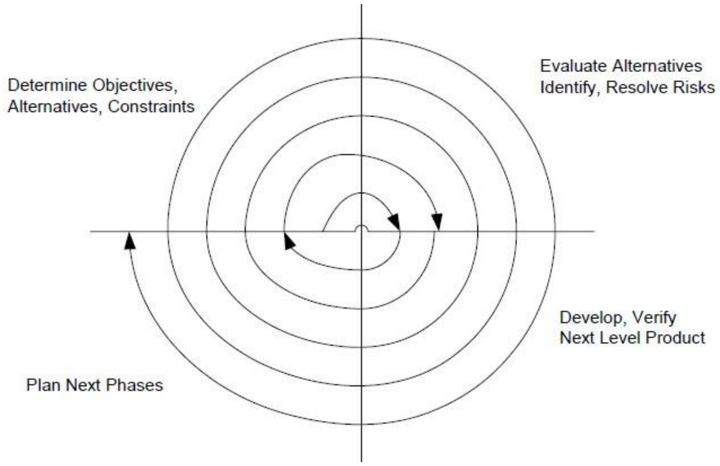
- In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions.
- A prototype is produced at the end of the risk analysis phase.
- If any risk is found during the risk analysis, then alternate solutions are suggested and implemented.

Engineering Phase

□ In this phase software is **developed**, along with testing at the end of the phase. Hence in this phase the development and testing is done.

Evaluation phase

□ This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.



Advantages of Spiral model

1. High amount of risk analysis hence, avoidance of Risk is enhanced.
2. Good for large and mission-critical projects.
3. Strong approval and documentation control.
4. Additional Functionality can be added at a later date.
5. Software is produced early in the software life cycle.

Disadvantages of Spiral model

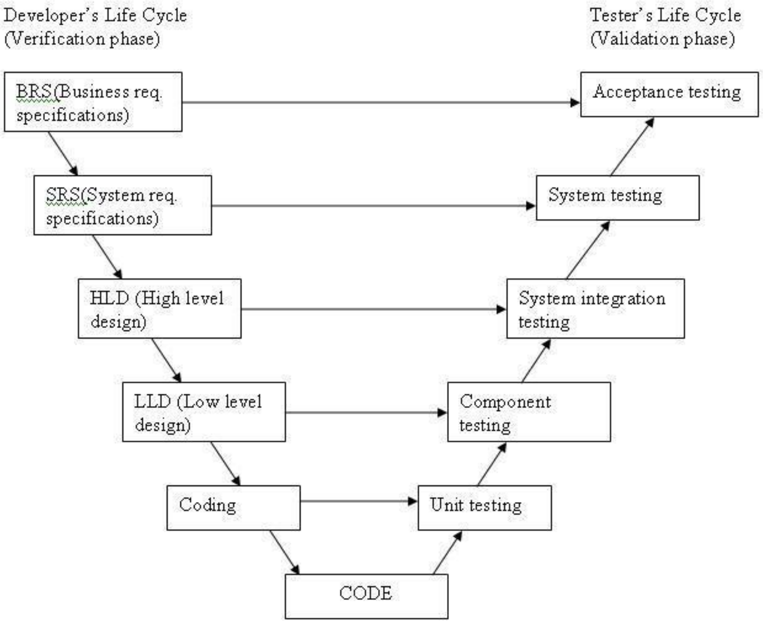
1. Can be a costly model to use.
2. Risk analysis requires highly specific expertise.
3. Project's success is highly dependent on the risk analysis phase.
4. Doesn't work well for smaller projects.

When to use Spiral model

1. When costs and risk evaluation is important
2. For medium to high-risk projects
3. Long-term project commitment unwise because of potential changes to economic priorities
4. Users are unsure of their needs
5. Requirements are complex
6. New product line
7. Significant changes are expected (research and exploration)

V-Model (Verification and validation)

- Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins.
- Testing of the system is planned in parallel with a corresponding phase of development.



Phases of V-Model (Verification and validation) model

Requirements phase

- In this model before development is started, a system test plan is created.
- The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design (HLD) phase

- This phase focuses on system architecture and design.
- It provides overview of solution, platform, system, product and service/process.
- An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase

- This is where the actual software components are designed.
- It defines the actual logic for each and every component of the system.
- Class diagram with all the methods and relation between classes comes under LLD.
- Component tests are created in this phase as well.

Coding phase

 This is at the bottom of the V-Shape model. Module design is converted into code by developers.

Implementation phase

 Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Advantages of V-model 1. Simple and easy to use.

2. Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.

3. Proactive defect tracking – that is defects are found at early stage.
4. Avoids the downward flow of the defects.
5. Works well for small projects where requirements are easily understood.

Disadvantages of V-model

1. Very rigid and least flexible.
2. Software is developed during the implementation phase, so no early prototypes of the software are produced.
3. If any changes happen in midway, then the test documents along with requirement documents has to be updated.

When to use the V-model

1. The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
2. The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.
3. High confidence of user is required for choosing the V-Shaped model approach.