Introduction to Software Engineering

By:Er. Akanksha Yadav,
Assistant Professor,
Department of computer science and engineering,foet,
University of Lucknow.

Books

TEXT Books:

- Pressman R S, *Software Engineering: A Practitioners Approach*, McGraw Hill.
- Datta S, *Software Engineering: Concepts and Applications,* Oxford University Press, 2010.

UNIT-1

Outline

- Software Engineering Overview
  - Definitions
  - Characteristics
  - Components
- Evolution and Significance of Software Engineering
- Challenges in SE
- Software development methodologies
- Software Development Process and Role of people in development

Software

A set of machine-readable instructions (programs) that directs a computer's processor to perform specific operations.

OR

A collection of instructions that enable a user to interact with the computer or have the computer perform specific tasks for them.

Characteristics of Software

- It is soft or intangible part of the computer system.
- Software *deteriorates. So,* Software is constantly subject to change (*alterations are required with respect to time).*
- Software is not manufactured but developed and customized.

Characteristics of Software (cont..)

- Software is costly to maintain.
- Software is inherently Complex.

(Complex $\neq$ complicated

Complex = composed of many simple parts related to one another.

Complicated = not well understood, or explained)

Software Component

- A software component is a modular building block for computer software
  - It is a modular, deployable, and replaceable part of a system that encapsulates implementation.
- A component communicates and collaborates with
  - Other components
  - Entities outside the boundaries of the system

Software Engineering

- Software engineering is an engineering discipline which is concerned with all aspects of software production.
- It is a process of solving customer's problem by the systematic development and evolution of large, high quality software systems within cost, time and other constraints.

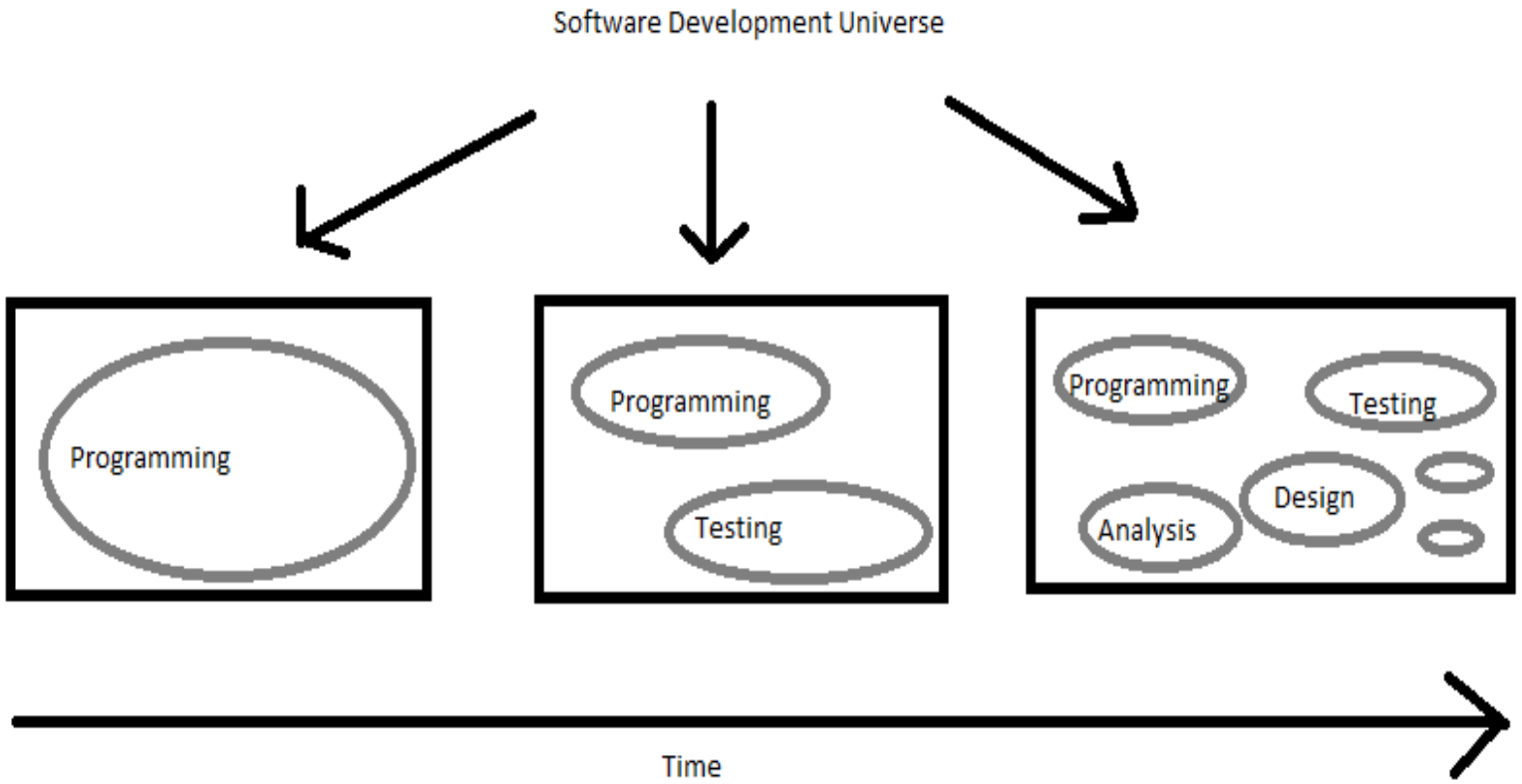
Evolution of Software Engineering

- Programming to Software Engineering
- Hardware- Software: From Coupling to Congress
- Advent of High-Level languages
- Advent of the Personal computer
- Global Software Development
- Return of Open Source

Programming to Software Engineering

- Programming plays a significant role in software development.
- This produces predictable results or spectacular results at times. Programming is more about creativity, about the elegance and beauty of algorithms.
- From a pre-eminently central position, programming has moved to be one of the concerns for the software developer.

Repositioning of Programming in the Software Development Universe



Hardware-Software: From Coupling to Congress

- Earlier Software came with hardware; almost no one sold or brought software by itself.
- Compare that with the present; probably the operating system is the only piece of Software now that comes with the hardware we buy.
- Every other application software is acquired and installed separately , often downloaded from web for free.

Coupling between hardware and software has now loosened.

Advent of High-Level languages

- What has changed over the past few decades is how we communicate with computer.
- What has brought about this change is the vehicle of communication that is LANGUAGE.
- One of the main objectives of any language is to support levels of abstraction in communication.

- High-level abstractions usually hide more details and make it easier for the human mind to broadly grasp an idea.

- High-level programming languages let one be a programmer without being able to communicate in machine or assemble language

Advent of Personal Computer

- In the 1950s and 60s, a computer was a considerably piece of equipment in price and size. Academic departments or large corporations owned them; it was really unthinkable for individual users to have their own computers
- But now a days, it is estimated that more than one billion PCs are in use in the world now.

Global Software Development

- As the web's presence increased throughout the 1990s and then into the new millennium, software engineering became a truly global enterprise.
- Global software development has deep economic as well as social implications.
- Global software development has put demands of cultural sensitivity, professional flexibility and political awareness on the new breed of the global software engineer.

Open Source

- Open source is about sharing the source code of software developed by a particular group or individual for free, so that others can use.
- Open source was nothing special in the initial era of computing.
- Companies sold hardware, and the software with it for free.
- When software become a commercial commodity in its own right, large corporations made every effort for the free flow of code.

Challenges in Software Engineering

Heterogeneity, Delivery and Trust

- Heterogeneity
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- Delivery
  - Developing techniques that lead to faster delivery of software;
- Trust
  - Developing techniques that demonstrate that software can be trusted by its users.

Software Development Methodologies/ Software Development Life Cycle Models

DEVELOPMENT LIFE CYCLE MODELS

•A SDLC model covers the entire lifetime of a product.

•From birth of a commercial idea to final installation of last release.

•A SDLC model is a descriptive and diagrammatic representation of the software life cycle.

•Represents all the activities required to make a software product transit through its life cycle phases.

•It also captures the order in which these activities are to be undertaken.

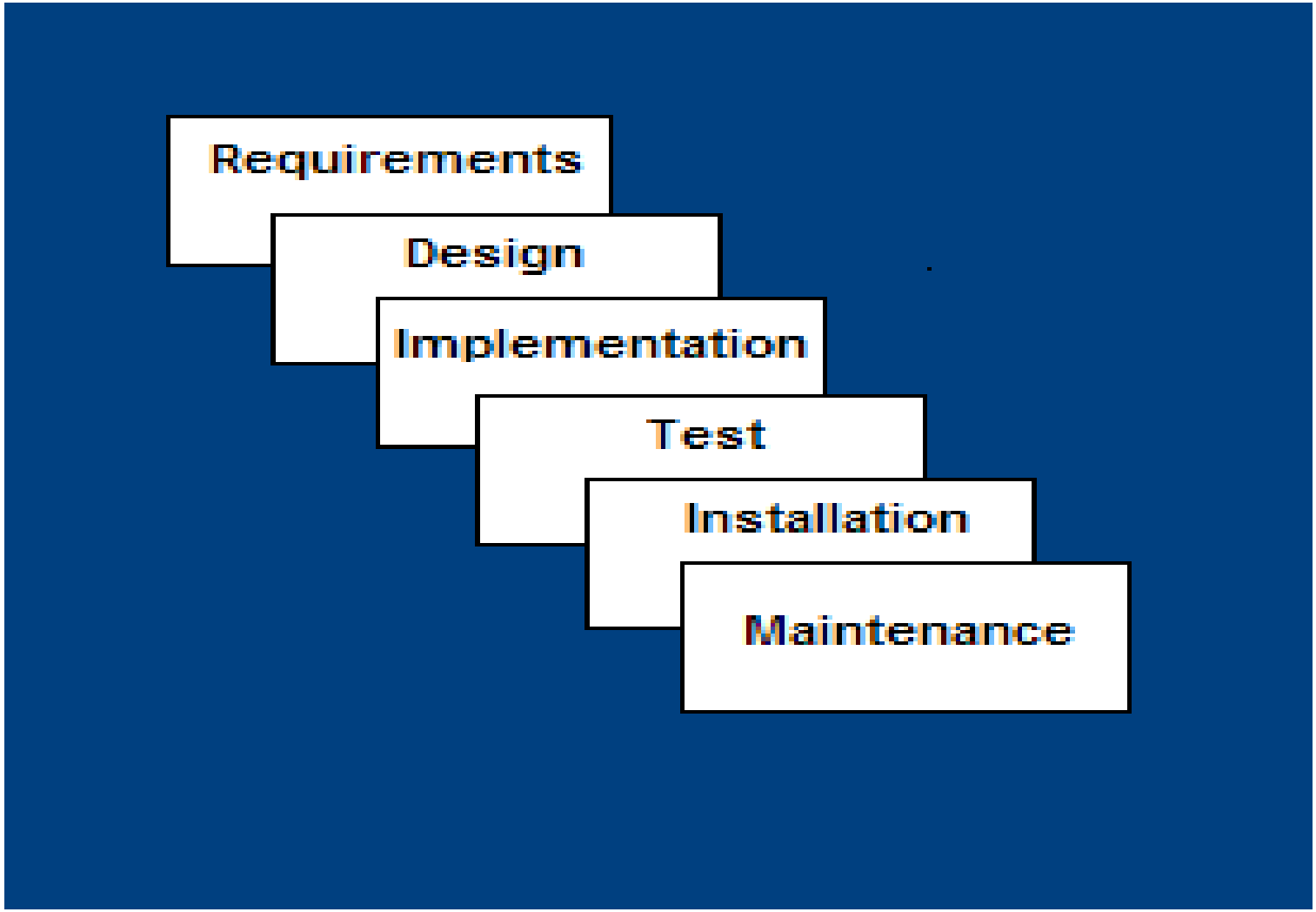
Software development methodologies:

- Water fall
- Prototyping
- Spiral Development
- Iterative & Incremental Development
- Agile Development.

Waterfall Model

- Provides a systematic approach to software development.
- The process of software development is represented by a sequence of steps.
- The sequential phases are what make this model linear, simple and systematic in nature.
- Each phase must be completed before you can move to next phase.
- This model is also known as the linear sequential model or classical life cycle .

Waterfall Model



Phases of Waterfall Model

Software requirement analysis:

- In this Phase , the requirements for the software are established through discussion with client and are then documented.
- Constraints are identified.
- Requirements are analyzed.
- Finally, a requirement specification document is created which serves the purpose of guideline for the next phase of the model.

Designing:

- If the first phase gets successfully completed and a well thought out plan for the software development has been laid then the next step involves formulating the basic design of the software on paper

- After the basic design gets approved, then a more elaborated technical design can be planned. Here the functions of each of the part are decided.

System Design:

- The system has to be properly designed before any implementation is started.
- This involves an architectural design which defines and describes the main blocks and components of the system, their interfaces and interactions.
- Hardware and Software components are identified
- E.g. this involves the definition or selection of a computer platform, an operating system, other peripheral hardware, etc. The software components have to be defined to meet the end user requirements and to meet the need of possible scalability of the system.

Software Design:

- Based on the system architecture which defines the main software blocks the software design will break them further down into code modules.

- All necessary system states like startup, shutdown, error conditions and diagnostic modes have to be considered and the activity and behaviour of the software has to be defined.

- The output of this phase is a Software Design Document which is the base of the following implementation work.

Implementation(Coding):

In this phase the actual coding of the software is done. The design of the previous phase is converted into the code.

Testing:

- In this phase, the output generated is checked to ensure that it matches the requirements.
- The programs developed in the previous phase are checked for the logical and syntax errors.

Maintenance and Support:

- The software developed needs maintenance and support.
- To ensure that the system will continue to perform as desired
- This refers to the changes as well as new requirements in the software after delivery.

Advantages of Waterfall Model

The Linear Sequential model offers the following advantages:

- It is easy to understand and implement.
- It prohibits skipping any phase in the sequence.
- As everything is documented a new team member can easily understand what's to be done.
- It is ideal for small projects and the requirements and goals of the project are well established in advance.

Disadvantages of Waterfall Model

The following are the disadvantages using Linear sequential model:

- If requirements change the Waterfall model may not work.
- Difficult to estimate time and cost for each stage of the development process.
- The working version of the software is available to the customer after testing. Therefore, if there is any major error it will remain till end of the testing.
- Due to linear nature if any phase is not completed , the software analyst and developers cannot proceed further.

Prototyping Model

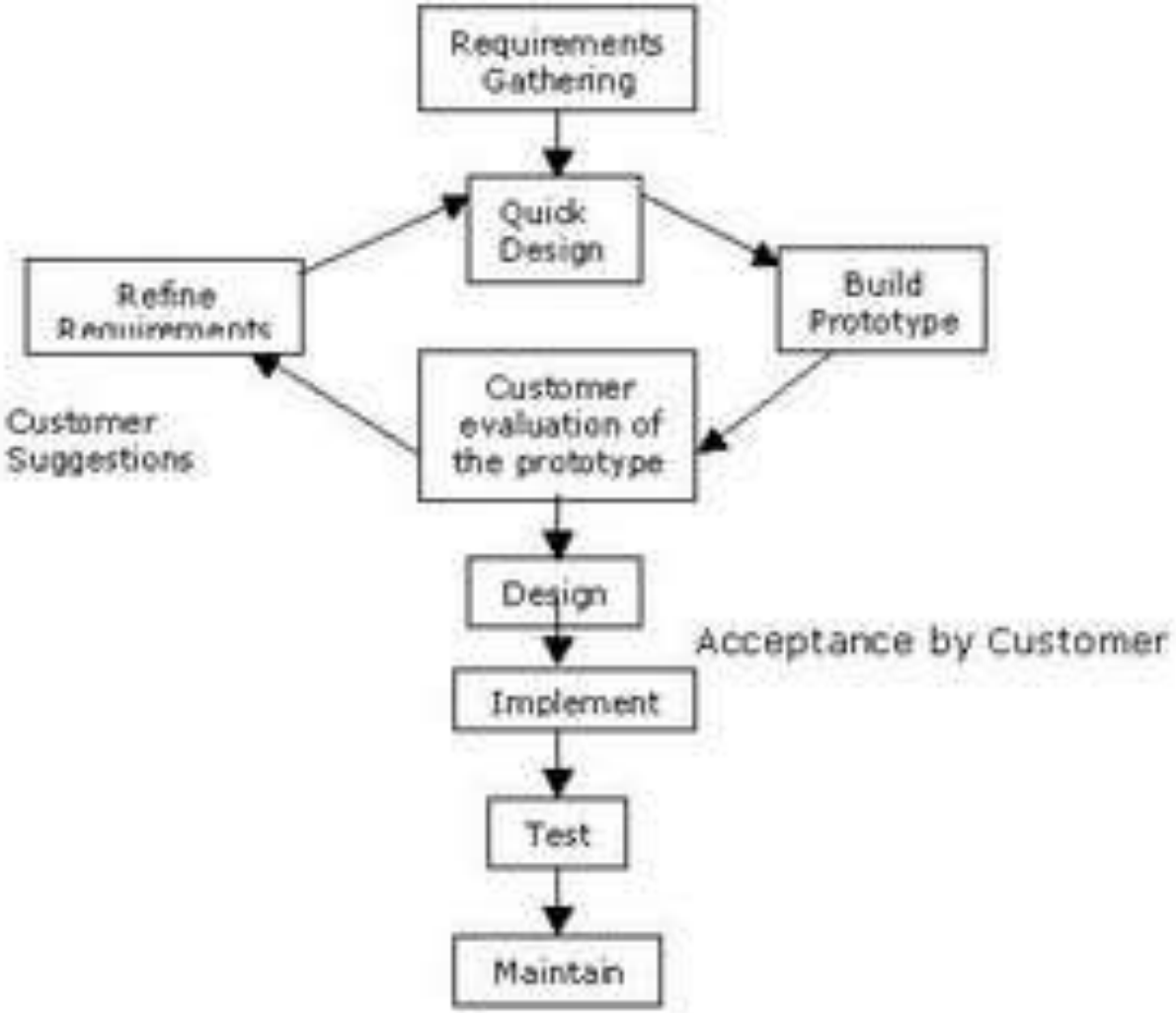
- In this model the developer and client interact to establish the requirements of the software.
- Define the broad set of objectives.
- This is follow up by the quick design, in which the visible elements of the software, the input and the output are designed..
- The final product of the design is a prototype.
- The client then evaluates the prototype and provides its recommendations and suggestion to the analyst.
- The process continues in an iterative manner until all the user requirements are met.

Need for a prototype in software development

This is a valuable mechanism for gaining better understanding of the customer's needs:

- How the screens might look like
- How the user interface would behave
- How the system would produce outputs

Phases of Prototyping Model



Advantages of Prototyping Model

The following are the advantages of Prototyping model:

- Due the interaction between the client and developer right from the beginning , the objectives and requirements of the software is well established.
- Suitable for the projects when client has not clear idea about his requirements.
- The client can provide its input during development of the prototype.
- The prototype serves as an aid for the development of the final product.

Disadvantages of Prototyping Model

The prototyping model has the following disadvantages.

- The quality of the software development is compromised in the rush to present a working version of the software to the client.
- The client look at the working version of the product at the outset and expect the final version of the product to be deliver immediately. This cause additional pressure over the developers to adopt shortcut in order to meet the final product deadline.

Iterative & Incremental Development Model

- **This** is a combination of both iterative method and incremental build model for development.
- *Iteration* refers to the cyclic nature of a process in which activities are repeated in a structured manner. And *increment* refers to the quantifiable outcome of each iteration.
- The basic idea behind this method is to develop a system through repeated cycles (Iterative) and in smaller portions (increment).

Iterative & Incremental Development Model(cont..)

- Each **iteration** consists of all of the standard Waterfall **phases.**
- At each iteration, design modifications are made and new functional capabilities are added.

OR

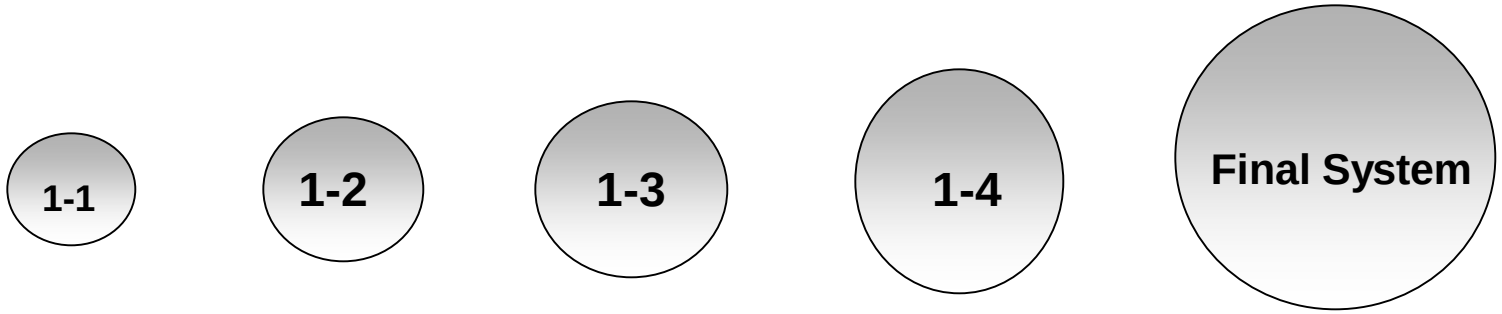
- Incremental development slices the system functionality into increments (portions) by prioritizing requirements. And, deliver a slice of functionality in each increment.

Iterative & Incremental Development Model(cont..)

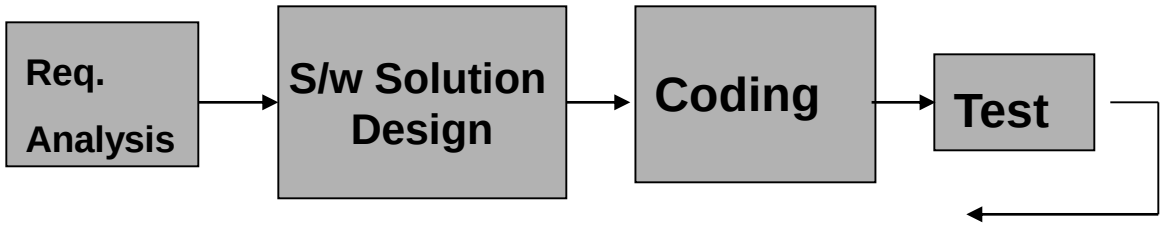
- In this model, the product is designed, implemented, integrated and tested as a series of incremental builds.

- Two key things: iterative refinement, where the process improves what already exists and is being done, and incremental development, where the process results in progress towards project objectives.

Incremental Model (INM)



Basic process



Next Increment

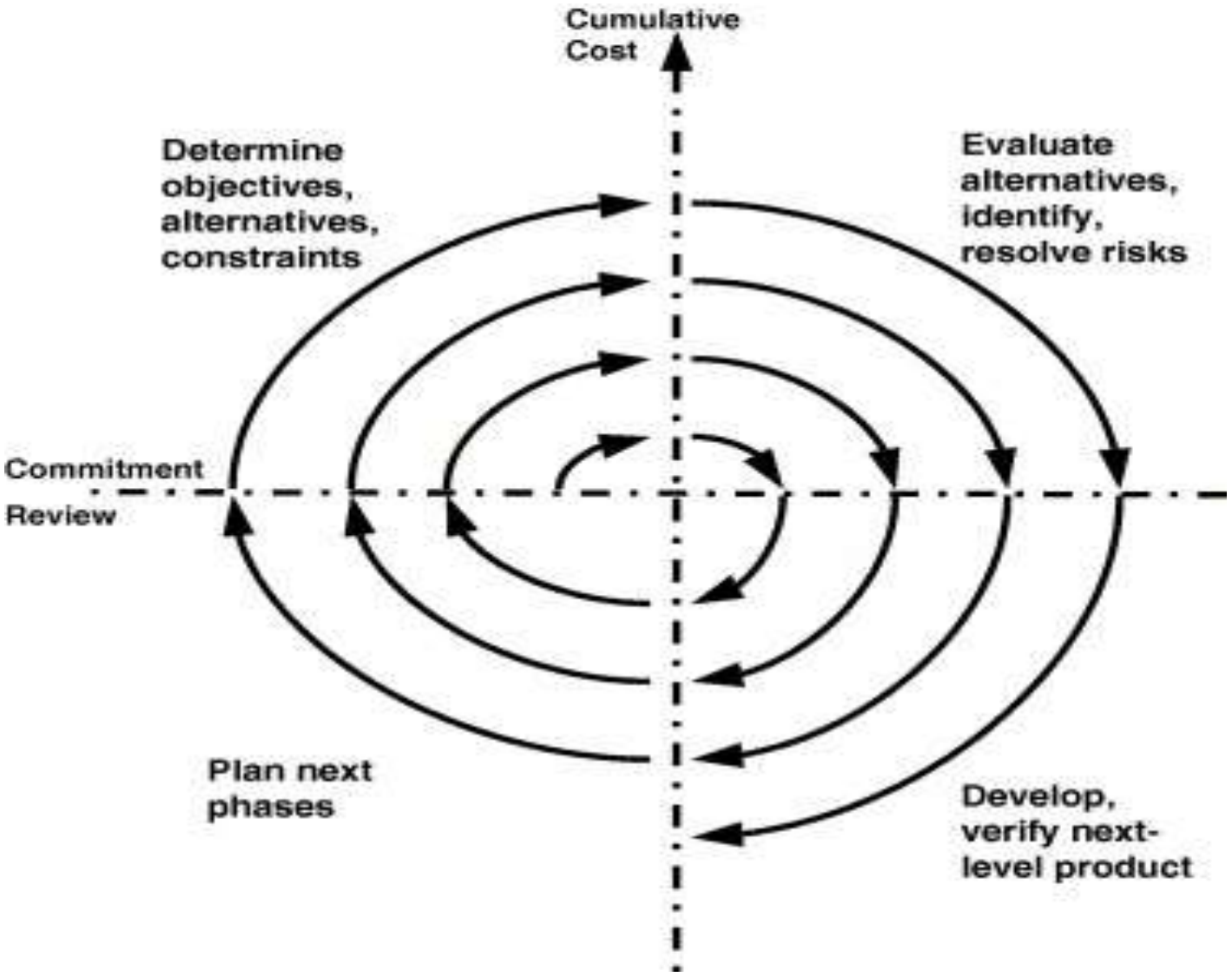
Iterative and Incremental Model Strengths

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses “divide and conquer” breakdown of tasks
- Initial product delivery is faster
- Customers get important functionality early
- Risk of effect of changing requirements is reduced

Iterative and Incremental Model Weaknesses

- Proper planning and designing is required.
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Total cost of the complete system is higher than waterfall.

Spiral model



- Each loop in a spiral represents a development phase (and we can have any number of loops according to the project). Each loop has four sections or quadrants :

  1. *To determine the objectives, alternatives and constraints.*

  2. *Risk analysis and evaluation of alternatives.*

  3. *Execution of that phase of development.*

  4. *Planning the next phase.*

- **First quadrant (Objective Setting)**
  - We try to understand the product objectives, alternatives in design and constraints imposed because of cost, technology, schedule, etc.
- **Second Quadrant (Risk Assessment and Reduction)**

- A detailed analysis is carried out for each identified project risk.
  - We try to find which other approaches can be implemented in order to fulfil the identified constraints. Risk mitigation is in focus in this phase

- **Third Quadrant (Development and Validation)**

  - In this phase we develop the planned product. Testing is also done. In order to do development, waterfall or incremental approach can be implemented.

Fourth Quadrant (Review and Planning)

- Review the results achieved so far with the customer and plan the next iteration around the spiral.

- Progressively more complete version of the software gets built with each iteration around the spiral.

Why spiral model is called meta model?

- Spiral model is also called as meta-model because in a way it comprises of other models of SDLC.
- Here we do software development systematically over the loops (adhering to waterfall approach) and at the same time we make a prototype and show it to user after completion of various phase (just in case of prototype model).
- This way we are able to reduce risks as well as follow systematic approach.

Advantages of Spiral Model

1) Spiral Life Cycle Model is one of the most flexible SDLC models in place.
2) Project monitoring is very easy and effective. Each phase, as well as each loop, requires a review from concerned people. This makes the model more transparent.
3) Risk management is one of the in-built features of the model, which makes it extra attractive compared to other models.
4) Changes can be introduced later in the life cycle as well.
5) Project estimates in terms of schedule, cost etc become more and more realistic.
6) It is suitable for projects where business needs may be unstable.
7) A highly customized product can be developed using this.

Disadvantages of Spiral Model

1) Cost involved in this model is usually high.
2) Skills required, to evaluate and review project from time to time, need expertise.
3) Rules and protocols should be followed properly to effectively implement this model. Doing so, through-out the span of project is tough.
4) It is not suitable for low risk projects.
5) Amount of documentation required in intermediate stages makes management of project very complex affair.

Software Development Process

- A process may be said to be a set of pre-defined activities recommended to team(s) of practitioners with the intention of fulfilling an objective.
- It is a set of prescribed steps towards performing some pre-defined task.

Different Software Development Processes

- Personal Software Process
- Team Software Process
- Unified Software Process

Personal Software Process

- Self Improvement Process for individual software Engineers to control, manage and improve the way they work.
- NOT a set of specific guidelines.
- Provides data and techniques to choose technologies and methods that are most effective, and make routine activities more predictable and efficient.

- Historical data can facilitate the estimation of program size and development time as well as give valuable inputs into quality improvement.

Process Elements

- Scripts:

  Describe how a process is ‘enacted’ and to point to standards, forms, guidelines and measures whenever necessary.

- Forms:

  Provide a placeholder for recording data.

Personal Process Elements(Cont.)

- Standards

  To Guide activities and give a base for verification of product and process.

- Process Improvement

  Facilitate the improvement of an existing process through a Process Improvement Proposal.

Two Key points for creating a new process

1. Consider the Current process as well.

2. Recognize the ‘incremental’ nature of process development needs.

Team Software Process

- Teams play a vital role in the production of large-scale, industrial strength software systems.
- Using TSP, individuals who have practiced PSP can come together to form effective teams.

What happens in TSP?

- As Team members learn best when a defined process is followed which gives rapid feedback.

  TSP *scripts* and *forms* offer a defined, measured and repeatable framework for teams, enabling teams to deliver products over a number of short development life cycles and evaluate results after each cycle.

What happens in TSP? (Cont.)

- As productive team work comes out of clear goals, supporting work environment and proper coaching and leadership.

TSP provides supporting environment.

Unified Software Development Process

- It is more than a single process
- Generic process framework that can be specialized for a very large class of software systems, for different application areas, different type of organizations and project sizes.
- An end-to-end process template for building software systems.
- Helps perform the workflows (activities that are associated with software development, Requirement analysis, design, Implementation, Testing) and phases of software development in a consistent and repeatable way.

- Disclaimer:

“This content is solely for the purpose of e-learning by students and any commercial use is not permitted. The author does not claim originality of the content and it is based on the following references”.

- REFERENCES:
- www.tutorialspoint.com
- www.wikipedia.com
- Books:
- Pressman R S, *Software Engineering: A Practitioners Approach*, McGraw Hill.
- Datta S, *Software Engineering: Concepts and Applications,* Oxford University Press, 2010.