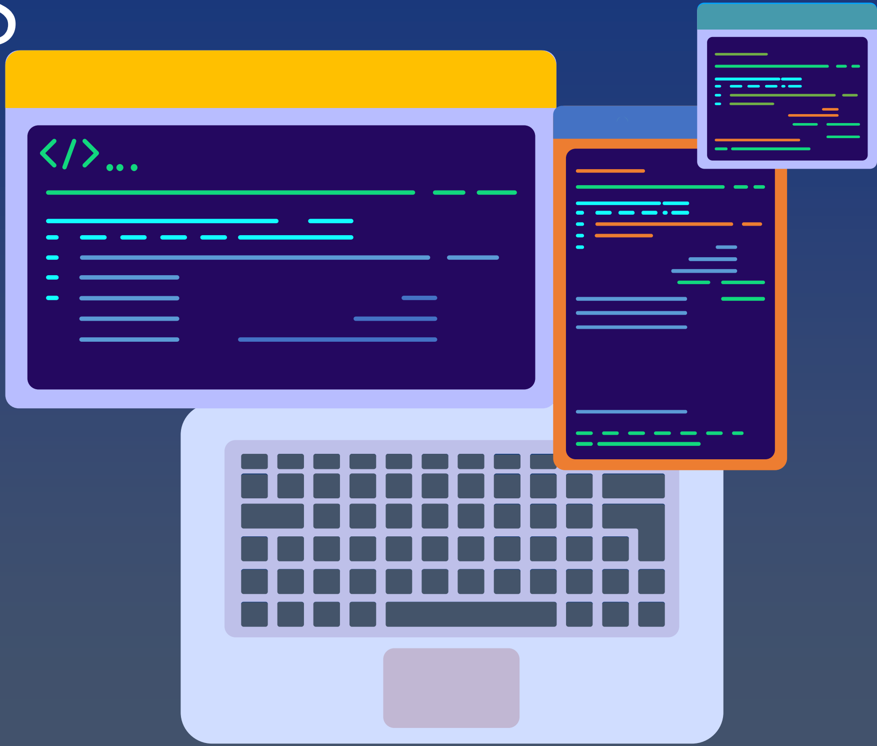
FUNDAMENTALS OF SOFTWARE ENGINEERING



INTE-E3-SYSTEMS TESTING AND AUTOMATION

TOPIC OUTLINE

01 Understand the fundamentals of software engineering.

02 Discuss various software lifecycles used in software development.

03 Identify the activities in the waterfall model, including requirements gathering and specification, software design, implementation, testing, and maintenance.

04 Discuss the several testing activities in a software testing life cycle.

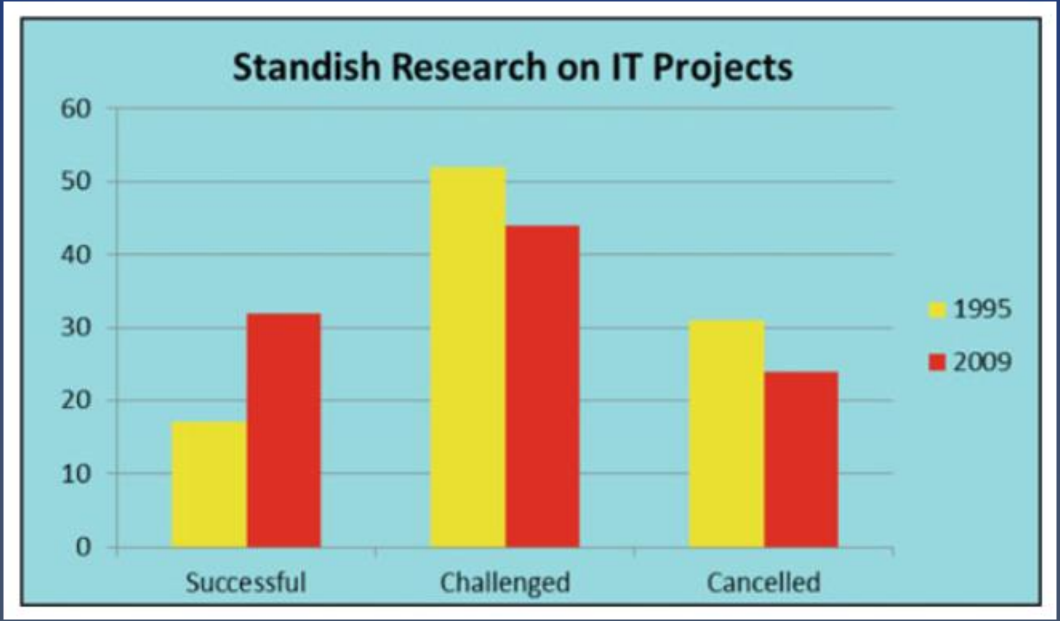
INTRODUCTION

- The approach to software development in the 1950s and 1960s has been described as the "***Mongolian Hordes Approach***" by Brooks (1975).
- The "method" was applied to projects that were running late, and it involved adding a large number of programmers to the project, with the expectation that this would enable the project schedule to be recovered.

The completed code will always be full of defects.
The coding should be finished quickly to correct these defects.
Design as you code approach.

—Software Development Philosophy during 1950/60s

Standish report—results of 1995 and 2009 Survey



What is Software Engineering?

- According to IEEE 610.12:
  - Software engineering is the application of a ***systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software***; that is, the application of engineering to software, and the study of such approaches.

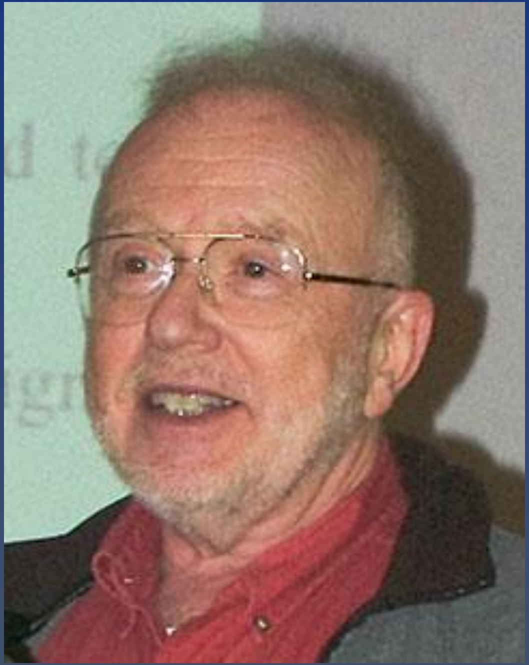
Software engineering includes as follows:

1. Methodologies to design, develop, and test software to meet customers’ needs.
2. Software is engineered.
3. Quality and safety are properly addressed.
4. Mathematics.
5. Sound project management and quality management practices are employed.
6. Support and maintenance of the software are properly addressed.

What is Software Engineering? Cont.

- Software engineering is not just programming.
- It requires the software engineer to state precisely the requirements that the software product is to ***satisfy*** and then to ***produce*** designs that will meet these requirements.
- The project needs to be ***planned and delivered on time and budget***.
- The requirements must provide a precise description of the problem to be solved, i.e. ***it should be evident from the requirements what is and what is not required***.

According to David Parnas…



- Computer scientists should be educated as engineers to enable them to apply appropriate scientific principles to their work.

- ***Software engineer*** should be able to use mathematics to assist in the modelling or understanding of the behaviour or properties of the proposed software system.

Challenges in Software Engineering

- The challenge in software engineering is to deliver high-quality software on time and on budget to customers.
- The accurate estimation of project cost, effort, and schedule is a challenge in project management.
- Risk management is an important part of project management, and the objective is to identify potential risks early and throughout the project and to manage them appropriately.
- Software quality needs to be properly planned to enable the project to deliver a quality product.

Software Processes and Lifecycles

- The software process assets in an organization generally consist of as follows:
- A software development policy for the organization
- Process maps that describe the flow of activities
- Procedures and guidelines that describe the processes in more detail
- Checklists to assist with the performance of the process
- Templates for the performance of specific activities (e.g. design, testing)
- Training materials.

Software Processes and Lifecycles cont.

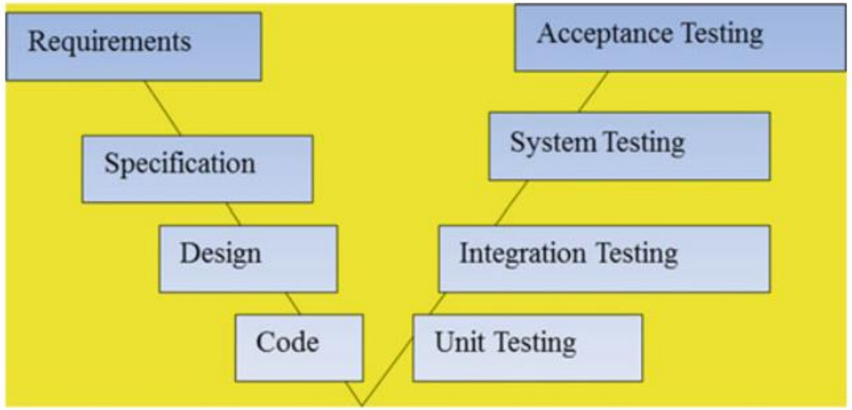
- The processes used to develop high-quality software generally include as follows:
- Project management process
- Requirements process
- Design process
- Coding process
- Peer review process
- Testing process
- Supplier selection and management processes
- Configuration management process
- Audit process
- Measurement process
- Improvement process
- Customer support and maintenance processes

Several well-known software lifecycles:

- Waterfall model (Royce 1970)
- Spiral model (Boehm 1988)
- Rational Unified Process (Rumbaugh 1999)
- Agile methodology (Beck 2000)

Waterfall lifecycle

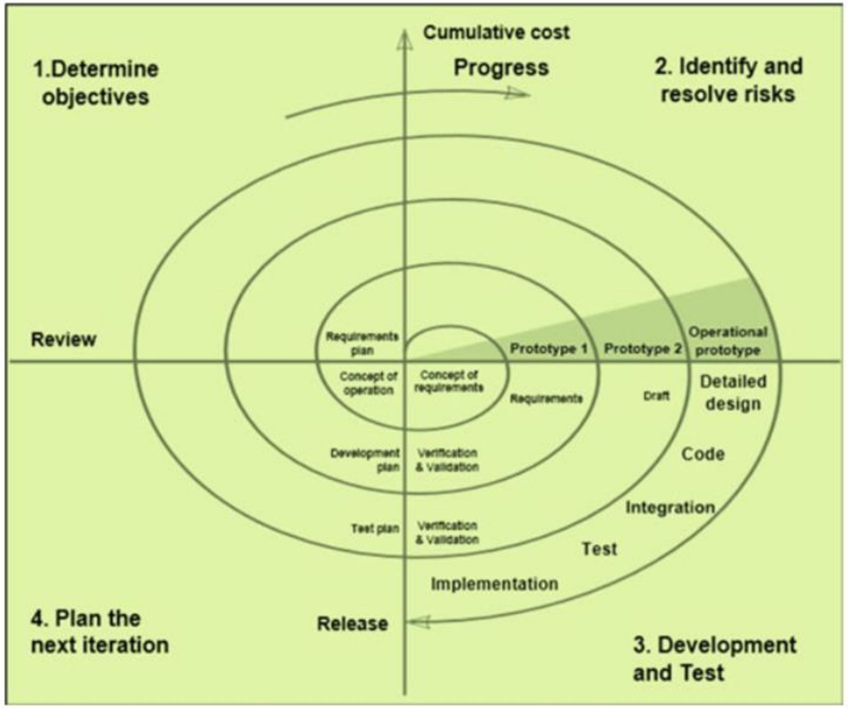
- Requirements gathering and definition
- System Specification(functional or non-functional)
- Design and implementation
- Testing(Unit, System, and User Acceptance)



Waterfall V lifecycle model

Spiral Lifecycle

- Developed by ***Barry Boehm***(1980s)
- Determining objectives
- Analysing the risks
- Updates to the requirements
- Design
- Code
- Testing
- User review of the particular iteration or spiral



Variations of Spiral Model

- Rapid Application Development(RAD)
- Joint Application Development(JAD)
- Dynamic Systems Development(DSDM)
- Agile Development(Popular)
- Other lifecycle models such as the iterative development process that combines the waterfall and spiral lifecycle model.

Cleanroom Methodology

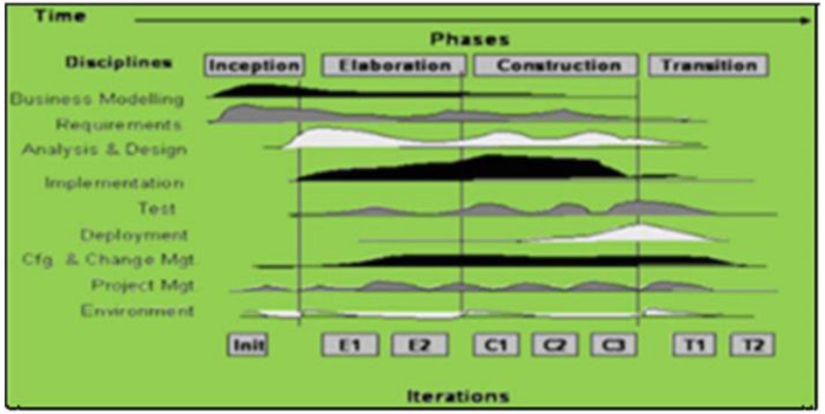
- Developed by Harlan Mills at IBM
- Its approach to software testing is based on the predicted usage of the software product, which allows a ***software reliability measure to be calculated***.
- The Rational Unified Process (RUP) was developed by IBM Rational.

Rational Unified Process

- Developed at the Rational Corporation (now part of IBM) in the late 1990s.
- It uses the Unified Modelling Language (UML) as a tool for ***specification and design***.
- ***UML is a visual modelling language*** for software systems that provides a means of specifying, constructing, and documenting the object-oriented system.
- James Rumbaugh, Grady Booch, and Ivar Jacobson developed it to facilitate the understanding of the architecture and complexity of a system.

Rational Unified Process cont.

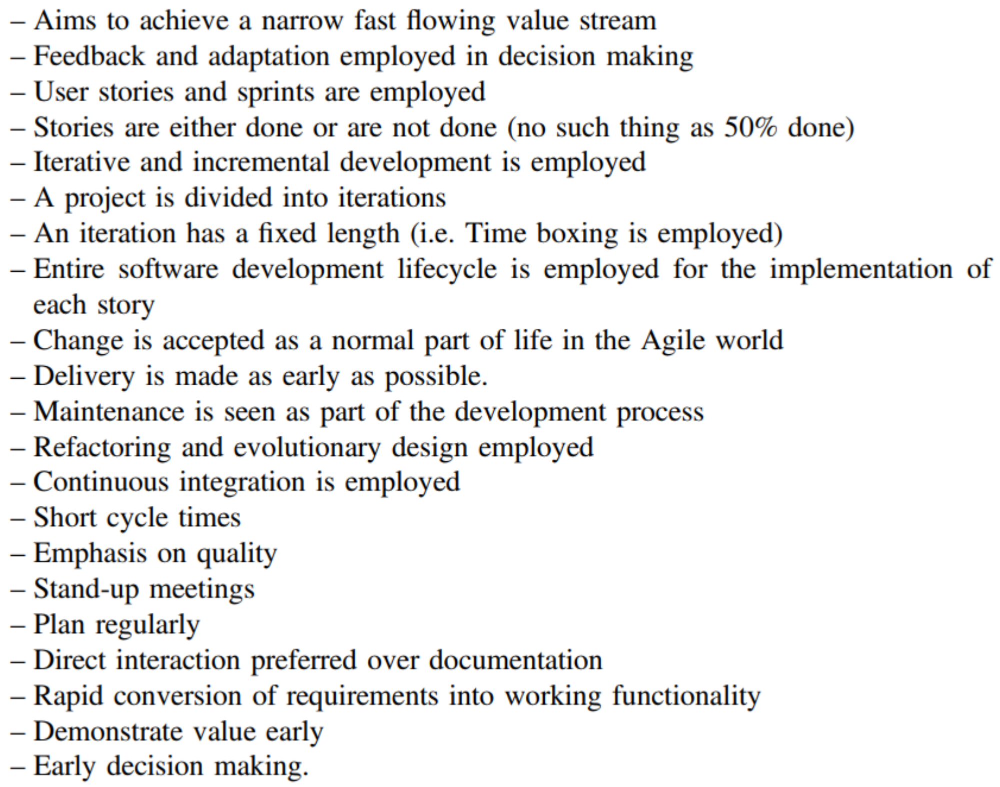
- RUP is ***use case driven, architecture centric, iterative and incremental***, and it includes cycles, phases, workflows, risk mitigation, quality control, project management, and configuration control.



Agile Development

- This is a software development methodology that is ***more responsive to customer needs than traditional methods such as the waterfall model***.
- The ***waterfall development model is similar to a wide and slow moving value stream***, and ***halfway through the project 100% of the requirements are typically 50% done***.
- However, for Agile development, ***50% of requirements are typically 100% done halfway through the project.***

Agile has a strong collaborative style of working including:



- Aims to achieve a narrow fast flowing value stream
- Feedback and adaptation employed in decision making
- User stories and sprints are employed
- Stories are either done or are not done (no such thing as 50% done)
- Iterative and incremental development is employed
- A project is divided into iterations
- An iteration has a fixed length (i.e. Time boxing is employed)
- Entire software development lifecycle is employed for the implementation of each story
- Change is accepted as a normal part of life in the Agile world
- Delivery is made as early as possible.
- Maintenance is seen as part of the development process
- Refactoring and evolutionary design employed
- Continuous integration is employed
- Short cycle times
- Emphasis on quality
- Stand-up meetings
- Plan regularly
- Direct interaction preferred over documentation
- Rapid conversion of requirements into working functionality
- Demonstrate value early
- Early decision making.

Specification of System Requirements

- ***Design***
- The design of the system consists of engineering activities to describe the architecture or structure of the system, as well as activities to describe the algorithms and functions required to implement the system requirements.
- It is a creative process concerned with how the system will be implemented, and its activities include architecture design, interface design, and data structure design.

Specification of System Requirements cont.

- ***Implementation***
- This phase is concerned with implementing the design in the target language and environment (e.g. C++ or Java) and involves writing or generating the actual code.
- The development team divides up the work to be done, with each programmer responsible for one or more modules. The coding activities generally include code reviews or walkthroughs to ensure that quality code is produced and to verify its correctness.
- The code reviews will verify that the source code conforms to the coding standards and that maintainability issues are addressed.
- They will also verify that the code produced is a valid implementation of the software design, and that it is fit for purpose.

Software Testing

- Unit Testing
- Integration Test
- System Test
- Performance Test
- User Acceptance Test

Support and Maintenance

- This phase continues after the release of the software product to the customer.
- Software systems often have a long lifetime, and the software needs to be continuously enhanced over its lifetime to meet the evolving needs of the customers.
- This may involve regular releases of new functionality and corrections to known defects.

Software Inspection

- There are several ***roles*** defined in the process including the moderator who chairs the inspection.
- The ***reader's*** responsibility is to read or paraphrase the particular deliverable, and
- the ***author*** is the creator of the deliverable and has a special interest in ensuring that it is correct.
- The ***tester*** role is concerned with the testing viewpoint.

Software Project Management

– Estimation of cost, effort, and schedule for the project
– Identifying and managing risks
– Preparing the project plan
– Preparing the initial project schedule and key milestones
– Obtaining approval for the project plan and schedule
– Staffing the project
– Monitoring progress, budget, schedule, effort, risks, issues, change requests, and quality
– Taking corrective action
– Replanning and rescheduling
– Communicating progress to affected stakeholders
– Preparing status reports and presentations.