CHAPTER 2 Software Engineering

LEARNING OUTCOMES

By the end of this chapter, you should be able to:

1. Discuss the main challenges in software enginnering and how can they be overcome
2. Describe various software life cycles in software engineering.
3. Discuss the benefits, any risks and disadvantages of few software development approaches i.e agile.
4. Describe the main activities that take place in software testing.

INTRODUCTION

Programmers are like engineers in that they build software products, and they require education on traditional engineering as well as the latest technologies. Computer science education often places more emphasis on the latest technologies rather than the key engineering principles of designing and building high-quality products that are safe to use.

Programmers need knowledge of sound engineering principles to enable them to build products that are safe for the public to use which includes:-

- A solid foundation on design
- Mathematics for building safe software products.

2.1 WHAT IS SOFTWARE ENGINEERING

"Software engineering is the application of a systematic, disciplined, qualitifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software, and the study of such approaches"

- The IEEE 610.12

Software engineering involves the multi-person construction of multi-version programs. Software engineering includes:-

- Methodologies to design, develop, implement and test software to meet customer's needs.
- Software is engineered – software products are properly designed, developed, and tested in accordance with engineering principles.
- Quality and safety are properly addressed.
- Mathematics may be employed to assist with the design and verification of software products – the level will depend on the safety – critical nature.
- Sound project management and quality management practices are employed.
- Support and maintenance of the software is properly addressed.

Software engineering is not just programming but requires the engineer to state precisely the requirements that the software product is to satisfy. The project needs to be planned and delivered on time and budget.

The term "engineer" is a title that is awarded on merit in classical engineering. It is generally applied only to people who have attained the necessary education and competence to be called engineers and who base their practice on classical engineering principles.

Software engineers need education on specification, design, turning designs into programs, software inspections and testing. The education should enable the software engineer to produce well-structured programs that are fit for purpose. However, there are arguments that software engineers have responsibilities as professional engineers. They are responsible for designing and implementing high-quality and reliable software that is safe to use. They are also accountable for their own decisions and actions and have responsibility to object to decisions that violate professional standards.

2.2 CHALLENGES IN SOFTWARE ENGINEERING

The challenge in software engineering is to deliver high-quality software **on time** and **on budget** to customers. Organizations and project managers need to determine how good their estimation process actually is and to make improvements as appropriate in order to get the accurate estimation of project cost, effort and schedule.

Risk management is also an important which the objective is to identify potential risks early and throughout the project and to manage them appropriately. It involves:-

- Determining the probability of each risk occurring
- Assessing the impact of each risk should it materialize
- Monitoring the risks during project execution

Actions are identified to reduce the probability of the risk occurring or reducing the impact of the risk should it materialize.

Software quality needs to be properly planned to enable the project to deliver a quality product. Flaws with poor quality software will at best cause irritation to clients, lead to a negative impact on customer satisfaction and damage the credibility of the company.

There are a number of high-profile software failures in the literature which caused embarrassment to the organizations as well as the cost of replacement and correction:-

- Millennium bug (Y2K) problem

- Floating-point bug in the Intel microprocessor
- European Space Agency Ariane – 5 disaster

These failures indicate that quality needs to be carefully considered when designing and developing software for all modern organizations. The effect of the above software failure:-

- Huge cost to correct the software (e.g. poorly designed legacy software in the case of the Y2K problem)
- Negative perception of a company and loss of market share in the case of the floating-point (e.g. Intel microprocessor problem)
- Loss of a valuable communications satellite (e.g. Ariane-5)

2.3 SOFTWARE PROCESSES AND LIFE CYCLES

The development of software involves many processes: defining requirements, project management and estimation, design, implementation and testing. It is important the process employed are fit for purpose.

Mature software companies will establish high-quality processes for the various software management and engineering activities. All affected staff will then be trained on the new processers and audits will be conducted to ensure that the process is followed. Data will be collected to improve the process. The software process assets in an organization generally consist of:-

- 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

The processes employed to develop high-quality software generally include processes for:-

- Project management process
- Requirement 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

The software development process has an associated life cycle that consists of various phases such as waterfall model, spiral model and the Rational Unified Process (RUP).

Waterfall Lice Cycles

The waterfall model starts with requirements gathering and definition. It is followed by the functional specification, the design and implementation of the software, and comprehensive testing (unit testing, system testing, integration testing and user acceptance testing).

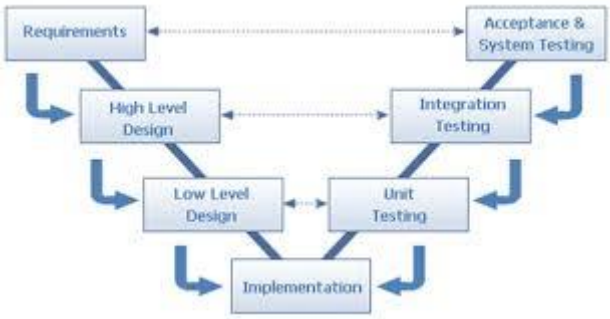

Figure 2- 1

The model, as described pictorially in Figure 2-1 is employed for projects where the requirements can be identified early in the project life cycle or are known in advance. Each phase has entry and exit criteria that must be satisfied before the next phase commences.

Most companies employ a set of templates (usually based on the IEEE standards) to enable the performance of activities in the various phases to be consistently performed.

Spiral Life Cycles

The spiral model is useful where the requirements are not fully known at project initiation. The requirements evolve as part of the development life cycle. The development proceeds in a number of spirals, where each spiral typically involves updates to the requirements, design, code, testing and a user review of the particular iteration or spiral.

The spiral is, in effect, a reusable prototype with the business analysts and the customer reviewing the current iteration and providing feedback to the development team. The feedback is then analysed and addressed in subsequent spirals.

Spiral approach is often used in joint application development for web-based software development as usability and the look and feel of the application is a key concern.

The implementation of part of the system helps in gaining a better understanding of the requirements of the system which feeds into subsequent development cycle in the spiral. The process repeats until the requirements and the software product are fully complete.

There are several variations of the spiral model:-

- Rapid Application Development (RAD)
- Joint Application Development (JAD)
- Dynamic Systems Development method (DSDM)

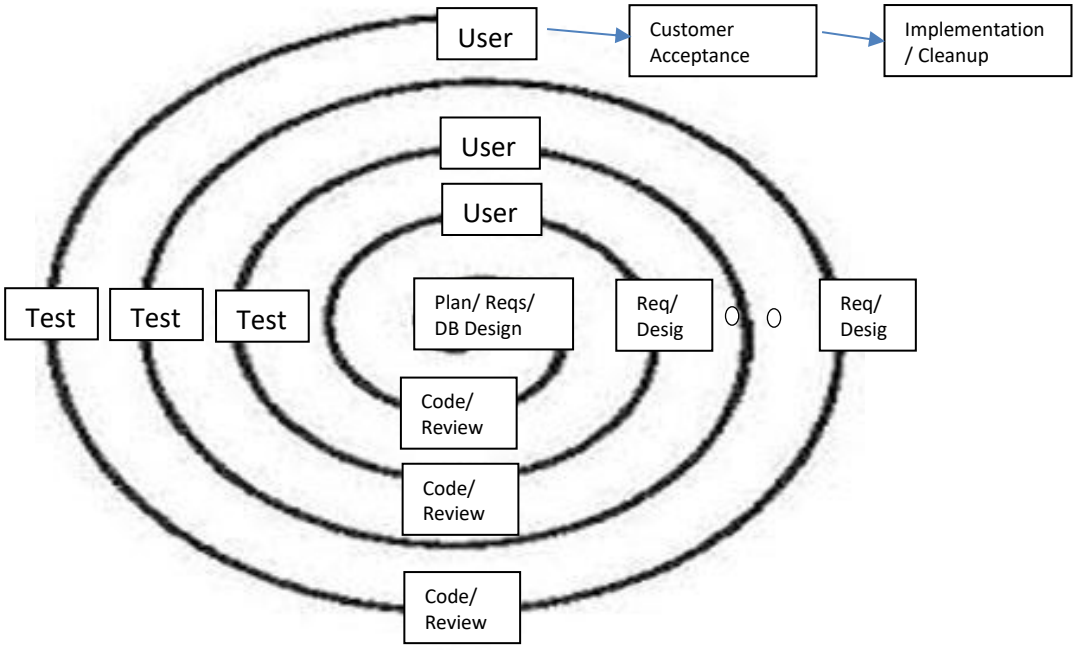

Figure 2- 2

Rational Unified Process

The Rational Unified Process (RUP) was developed at the Rational Corporation and it uses the Unified Modelling Language (UML) as a tool for specification and design. UML is a visual modelling language for software systems and provides a means of specifying, constructing, and documenting the object-oriented system which facilitates the understanding of the architecture and complexity of the system.

The unified process is use case driven, architecture centric, iterative, and incremental which includes:-

- Cycles
- Phases

- Workflows
- Risk mitigation
- Quality control
- Project management and
- Configuration control.

Requirements are gathered as use cases and the use cases describe the functional requirements from the point of view of the user of the system. They describe what the system will do at a high level and ensure that there is an appropriate focus on the user when defining the scope of the project.

Use cases also drive the development process, as the developers create a series of design and implementation models that realize the use cases. The developers review each successive model for conformance to the use case model, and the test team verifies that the implementation correctly implements the use cases.

RUP approach is used to decompose the work into smaller slices or mini-projects, where each mini-project is an iteration that results in an increment to the product (Figure 2-3: Iteration in Rational Unified Process). RUP also is a way to mitigate risk in software engineering.

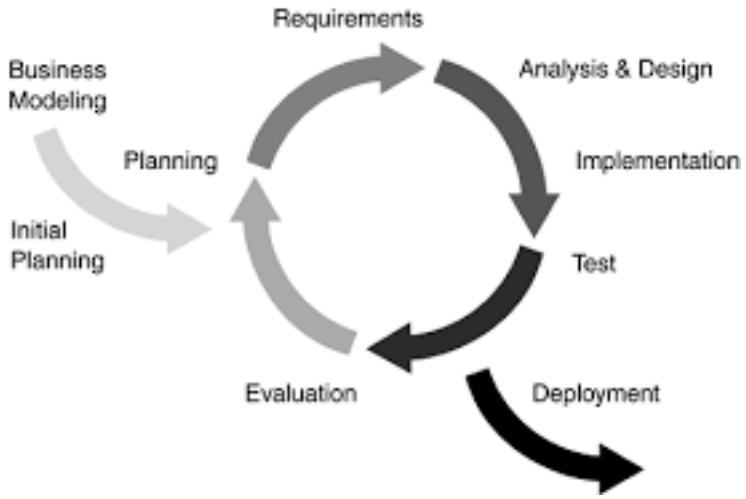

Figure 2- 3

The unified process consists of four phases: inception, elaboration, construction and transition (Figure 2-4). Each phase consists of one or more iterations and an iteration consists several workflows or disciplines: requirements, analysis, design, implementation and test. Each phase terminates in a milestone with one or more project deliverables.

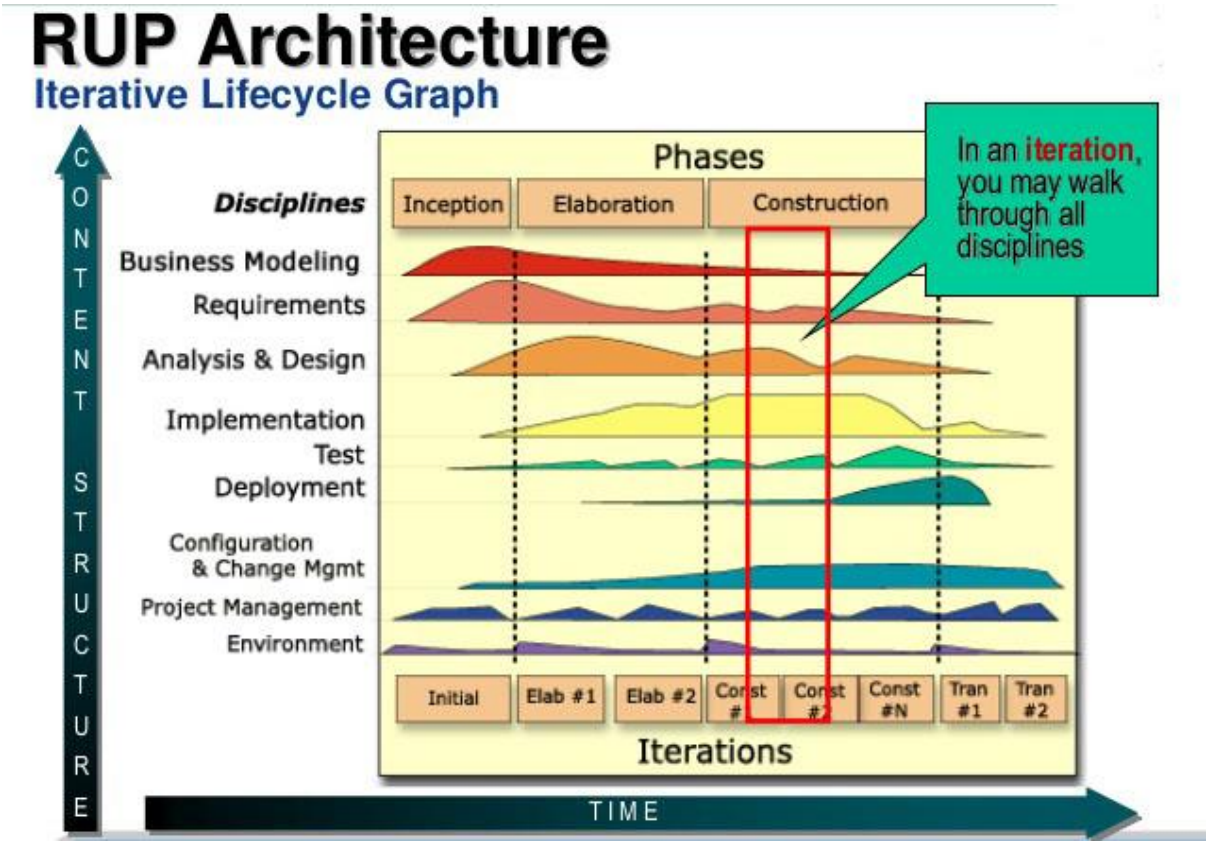

Figure 2- 4

Phases	Description
Inception	Identifies and prioritizes the most important risks. It is concerned with the initial project planning and cost estimation and initial work on the architecture and functional requirements for the product.
Elaboration	Specifies most of the use cases in detail, and the system architecture is designed.
Construction	Concerned with building the product. At the end of this phase, the product contains all of the use cases that management and the customer have agreed for the release.
Transition	Covers the period during which the product moves into the customer site and includes activities such as training customer personnel, providing on-site assistance, and correcting defects found after delivery.

Table 1 - 1

Agile Development

Agile is a software development methodology that claims to be more responsive to customer needs than traditional methods such the waterfall model. In Agile development, 50% of requirements are typically 100% done halfway through the project.

Agile methodology has a strong collaborative style of working and its approach includes the following:-

- Aim 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 not done
- Iterative and incremental development is employed
- A project is divided into iterations
- An iteration has a fixed length
- Entire software development life cycle is employed for the implementation of each story
- Change is accepted as a normal part of life in the Agile word
- 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.

In Agile world, ongoing changes to requirements are considered to be normal and it is believed to be more realistic to change requirements regularly throughout the project rather than attempting to define all of the requirements at the start of the project. This methodology allows and supports:-

- Controls to manage changes to the requirements
- Good communication
- Early regular feedback as part of the process

In Agile, stories often build upon other stories and the entire software development life cycle is employed for the implementation of each story. Stories are either done or not done, i.e. there is no such thing as a story being 80% done. The story is complete only when it passes its acceptance tests. Stories are prioritized based on a number of factors:-

- Business value of story
- Mitigation of risk
- Dependencies on other stories

Sprint planning is performed before the start of an iteration. The goal is to assign stories to the iteration to fill the available time. The estimates for each story and their priority are determined, and the prioritized stories are assigned to the iteration.

During the iteration, a short morning stand-up meeting would be held daily which attended by the project manager and the project team. It discusses the progress made the previous day, problem reporting and tracking, and the work planned the day ahead. A separate meeting is conducted for issues that require more detailed discussion.

Once the iteration is complete the latest product increment to be demonstrated to the audience including the product owner – feedback received and new requirements identified.

Retrospective meeting also conducted by the team to identify what went well and what went poorly during the iteration – improvement actions for future iterations identified.

With the philosophy of "two heads are better than one", agile employs pair programming and a collaborate style of working – allows multiple perspectives in decision making and a broader understanding of the issues.

As software testing is very important, agile employs automated testing for unit, acceptance, performance and integration testing. Tests are run often and aim to catch programming errors early – usually run on a separate build server to ensure all depencies are checked. Tests are re-run before making a release. Agile employs test-driven development with tests written before the code - the developers write code to make a test pass with ideally developers only coding against failing tests. This approach forces the developer to write testable code.

In agile, refactoring is employed as a design and coding practice with the objective to change how the software is written without changing what it does. Refactoring is a tool for evolutionary design where the design is regularly evaluated and improvements implemented as they are identified.

Continuous integration allows the system to be built with every change. Early and regular integration allows early feedback to be provided – problems identified earlier.

2.4 LIFE CYCLE PHASES

The waterfall software development life cycle consists of various phases:-

- Business requirements definition
- Specification of system requirements
- Design
- Implementation
- Unit testing
- System testing
- UAT testing

- Support and maintenance

Business requirements definition

In system development, if the requirements are incorrect then the implemented system would be incorrect as well. The requirements:-

- Specify what the customer wants
- define what the software system is required to do
- How the above two to be done

Prototyping may be employed to assist in the definition and validation of the requirements. The prototype will include key parts of the system and the users may examine the prototype to clarify the requirements and to give early feedback. The prototype may be thrown away once the prototyping is complete or it may be reused in the development of the system.

CHARACTERISTICS OF GOOD REQUIREMENTS

1. The requirements are numbered to facilitate traceability

2. Each requirement
    a. is clear and unambiguous
    b. is testable
    c. has a priority to indicate its importance
    d. may be implemented
    e. is broken down as fully as possible
    f. is consistent with the project's objectives
    g. is necessary
    h. is stated as a stakeholder need (i.e. premature design/ solution or implementation is not included)

3. The requirements
    a. have been analysed and any conflicts between the requirements are resolved
    b. are complete and consistent
    c. are correct
    d. are traceable between business requirements to the system requirements and vice versa.

4. The technical constraints have been identified.

The specification of the requirements needs to be unambiguous to ensure that all parties involved in the development of the system understand fully what is to be developed and tested. The implications of the proposed set of requirements also need to be

considered, as the choice of a particular requirement may affect the choice of another requirement. Therefore, feature interactions need to be identified and investigated at the requirement phase to determine how interactions should be resolved. This can be achieved by the following activities or processes:-

- Requirement gathering and prototyping
- Requirement consolidation and documentation
- Requirements review and validation
- Technical analysis
- Developer/ client contract
- Changes to requirements
- Requirements specification notations

Specification of System Requirements

This phase involves a detailed specification of the software requirements of the product, and it is essentially a statement of what the organization will provide to meet the business requirements.

The detailed business requirements is a statement of what the customer wants, whereas the specification of the system requirement is a statement of what will be delivered by the developer or software house. It is important to ensure that business requirements are addressed by the system requirements, and the system requirements are reviewed by the stakeholders.

There are two categories of system requirements:-

Functional requirements

Define the functionality that is required of the system and it may include screen shots and report layouts.

Non-functional requirements

Generally include security, reliability, performance, and portability requirements, as well as usability and maintainability 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 which include architecture design, interface design and data structure design. There are often several possible design solutions for a particular system and the designer need to decide on the most appropriate solution. Each potential solution need to be evaluated and the chosen one may leads to certain criteria or factors i.e. the simplest and least costly.

The design may be specified in various way such as graphical notations that display the relationship between the components making up the design. The notation may include flow charts or various UML diagrams such as sequence diagrams and start charts. Program description language or pseudocode can be used to give specifications of algorithms and data structures that are the basis for implementation. Natural language is often used to express information that cannot be expressed formally.

The design activities include:-

- Architecture design of system with all sub-systems
- Abstract specification of each sub-system
- Interface design (for each sub-system)
- Component design
- Data structure design
- Algorithm design

The design method can give several views of the system:-

- Data flow view (system modelled using data transformations that take place)
- Entity-relation (describes the logical data structures being used)
- Structural view of system components and their interactions

In this phase, object oriented design often applied where the system is viewed as a collection of objects rather than functions with each object managing its own state information.

Implementation

This phase involves translating the design into the target code, and it involves writing or generating the actual code. The coding activities usually include code reviews or walkthroughs to ensure that high-quality source code is produced and to verify its correctness. This activity also verify the source code conforms to the coding standards, maintainability issues are addressed and the code produced is a valid implementation of the software design.

As software reuse has become popular and provide a way to speed up the development process, the components or objects that may be reused (developed internally or purchased off-the-shelf) need to be identified and handled accordingly.

Even software reuse helps in increasing productivity and a faster time to market, it is important to consider the risks of reuse since there are inherent risks with customized-off-the shelf (COTS) software such as the supplier may decide to no longer support the software or there is no guarantee that a component that has worked successfully in one domain will work correctly in a different domain.

Testing

Testing is conducted to verify that quality has been built into the software and that the implemented software is valid with respect to the requirements. Various types of testing that may be conducted:-

Unit Testing

- Performed by the programmer on the completed unit or module prior to its integration with other modules.
- The tests are written by the programmer with the objective to show the code satisfies the design
- Each unit test case is documented and it should include a test objective and the expected result.
- The developer executes the unit tests, records the results, correct any identified defects and re-test the software.

Integration Test

- Performed by the development team on the integrated system once all of the individual units work correctly in isolation.
- The objective is to verify that all of the modules and their interfaces work correctly together and to identify and resolve any issues – units may work correctly in isolation but may fail when integrated with other units.

Performance Test

- The purpose of this kind of testing is to ensure that the performance of the system is within the bounds specified in the non-functional requirements.
- It may include load performance testing, where the system is subjected to heavy loads over long period of time and stress testing, where the system is subjected to heavy loads during a short time interval.
- The testing often involves the simulation of many users using the system and measuring the response times for various activities.
- The scalability of the system also need to be determined in order to support future growth
- Test tools are used in simulating a large number of users and heavy loads.

User Acceptance Test (UAT)

- The objective of this type of testing is to verify whether the product satisfies the business requirements and meets the customer expectations.
- It is conducted by business analyst and customer representatives.
- Its successful completion demonstrates that the customer requirements are satisfied and the customer is happy to accept the product.

- UAT is usually performed under controlled conditions at the customer site, and it matches the intended real-life behaviour of the system where the customer can see the product in operation and can judge whether the system is fit for purpose.

Maintenance

This phases continues after the release of the software product to the customer. Problems identified at this stage are reported as the customer support and maintenance agreement.

The support issues usually require investigation - the issue may be a defect in the software, an enhancement to the software product, or due to a misunderstanding. The support and maintenance team will identify the causes of any identified defects and will implement an appropriate solution to resolve.

The presence of a maintenance phase suggests an acceptance of the reality that problems with the software will be identified post-release.

One important measure of quality is customer satisfaction with the company, and customer feedback may be obtained informally or formally. One formal approach is to define a customer satisfaction survey and to request customers to provide structured feedback. The information may be used to determine the overall level of customer satisfaction with the company. The questions in a customer survey questionnaire could include:-

- Quality and reliability of the product
- Usability of the product
- Testing effectiveness
- Customer support

Feedback of customer satisfaction survey should be used to develop appropriate action plans to address the key findings as companies wish to improve to serve their customers better.

2.5 SOFTWARE INSPECTIONS

Software inspections play a key role in building quality into software products and verifying that the products are of high quality. There are a number of well-known approaches can be applied:-

- Fagan methodology
- Gilb's approach
- Prince2's approach

There are various roles defined in the inspection process including the moderator who chairs the inspection.

The ***moderator***:-

- Is skilled in the inspection process and is responsible for ensuring that all of the participants receive the appropriate materials for the inspection and sufficient preparation can be done.
- Will chair the inspection meeting and will cancel the inspection if the inspectors have done inadequate preparation.
- Will ensure that the defects identified are recorded, and the speed of inspection does not exceed the recommended guidelines.

The ***reader's*** is responsible to read or paraphrase the particular deliverable.

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.

According to Fagan methodology, there are seven steps in inspection process:-

- Planning
- Overview
- Prepare
- Inspect
- Process improvement
- Re-work
- Follow-up

2.6 SOFTWARE TESTING

Software testing plays a key role in verifying a software product is of high quality and matches the customer's quality expectations. Testing verifies that the software is fit for purpose and that the requirements have been correctly implemented, as well as identifying defects present in the software.

There are various types of testing as discussed in the previous section. The testing needs to be planned and this includes identifying its scope as well as the test environment, support tools and resources required.

The test cases are then designed for the various types of testing from the requirements and design documents. Tests are then executed, the results logged and reported, with any defects corrected and re-tested.

The quality of the testing is dependent on the maturity of the test process. A good software test process should include several of the following activities:-

- Test planning and risk management
- Dedicated test environment and test tools
- Test case definition
- Test automation
- Formality in handover to test department
- Test execution
- Test result analysis
- Test reporting
- Measurements of test effectiveness
- Post-mortem and test process improvement

2.7 SOFTWARE PROJECT MANAGEMENT

The timely delivery of high-quality software requires good management and engineering processes. Software projects have a history of being delivered late or over-budget.

A good project management practices play a key role in the timely delivery of high-quality and reliable software. Project management includes the following activities:-

- 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.
- Re-planning and re-scheduling
- Communicating progress to affected stakeholders
- Preparing status reports and presentations.

The project plan will contain or reference several other plans such as the project quality plan, the communication plan, the configuration management plan and the test plan.

Project estimation and scheduling are difficult as often software projects are breaking new ground and differ from previous projects. Previous estimates may often not be a good basis for estimation for the current project. Unanticipated problems can arise for technically advanced projects, and the estimates may often be optimistic.

The effective management of risk during a project is essential to project success. Risks arise due to uncertainty and the risk management cycle involves:-

- Risk identification
- Risk analysis and evaluation
- Identifying responses to risks
- Selecting and planning a response to the risk
- Risk monitoring

Once the risks have been identified they are logged (e.g. in the Risk Log). The likelihood of each risk arising and its impact is then determined. The risk is assigned an owner and an appropriate response to the risk determined.

2.8 PROCESS MATURITY MODELS

The SEI and other quality experts believe that there is a close relationship between the quality of the delivered software and the quality and maturity of the processes used to create the software. Therefore, there is a need to focus on the software process as well as on the product.

The SEI applied the process improvement principles used in the manufacturing field to develop process maturity models for the software field, and they developed models such as the CMM and its successor the CMMI.

The CMMI allows organizations to benchmark themselves against other similar organizations which can be done thru formal SEI SCAMPI appraisals conducted by an authorized SCAMPI lead appraiser.

The use of the CMMI provides a solid engineering approach to the development of software. It requires high-quality processes to be in place for project management, requirement development and management, design and development, reviews and testing, independent quality audits and so on. The next chapter gives further elaboration to the CMMI.

SUMMARY

Software engineers need to be properly trained to enable them to build high-quality products that are safe to use.

Programmers are like engineers in the sense that they build products. Therefore, programmers need to receive an appropriate education in engineering as part of their training.

Software engineering involves multi-person construction of multi-version programs. It is a systematic approach to the development and maintenance of the software, and it requires a precise statement of the requirements of the software product and then the design and development of a solution to meet the requirements.

It includes methodologies to design, develop, implement, and test software as well as sound project management, quality management, and configuration management practices. Support and maintenance of the software is properly addressed.

Software process maturity models such as the CMMI have become popular in recent years. They place an emphasis on understanding and improving the software process to enable software engineers to be more effective in their work.

REFERENCES

O'Regan, G (2011).Introduction to Software Process Improvement. London, Springer.

Irfan Ul-Haq (2012). Advanced Software Engineering. Retrieved from https://www.slideshare.net/bilalhashmishah/software-process-improvement-12777417