SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

COURSE NAME : 19CS302 AGILE SOFTWARE ENGINEERING

II YEAR /III SEMESTER

Unit 1- Introduction to Software Engineering

Topic 2: Software Engineering

Brain Storming

1. What is Software?
2. How to develop software?

What is Software Engineering?

- Software engineering (SE) is concerned with developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all the requirements that customers have defined for them.

Software Engineering Practice

- Consists of a collection of concepts, principles, methods, and tools that a software engineer calls upon on a daily basis.
- Equips managers to manage software projects and software engineers to build computer programs.
- Provides necessary technical and management how to getting the job done.

Essence (Nature) of Practice

•Understand the problem (communication and analysis)

- Who has a stake in the solution to the problem?
- What are the unknowns? and what (data, function, behavior) are required to properly solve the Problem?
- Can the problem be compartmentalized? Is it possible to represent smaller problems that may be easier to understand.
- Can the problem be represented graphically? Can analysis model be created?

Plan a solution (planning, modeling and software design)

- Have you seen similar problems like this before?
- Has a similar problem been solved? If so, are the elements of the solution reusable?
- Can sub problems be defined and are solutions available for the sub problems?

Conti.....

Carry out the plan (construction; code generation)

- Does the solution conform to the plan? Is the source code traceable back to the design?
- Is each component of the solution correct? Has the design and code been reviewed, or better?

Examine the results for accuracy (testing and quality assurance)

- Is it possible to test each component of the solution? Has a reasonable testing strategy been implemented?
- Does the solution produce results that conform to the data, function, and behavior that are required?
- Has the Software been validated against all stakeholders requirement?

Types of Practice

- Communication Practice
- Planning Practice
- Modeling Practice
- Construction Practice
- Testing Practice
- Deployment Practice

Communication Practice

1. Listen to the speaker and concentrate on what is being said
2. Prepare before you meet by researching and understanding the problem
3. Someone should facilitate the meeting and have an agenda
4. Face-to-face communication is best, but also have a document or presentation to focus the discussion
5. Take notes and document decisions

Conti...

6. Strive for collaboration and consensus
7. Stay focused on a topic; modularize your discussion
8. If something is unclear, draw a picture
9. Move on to the next topic a) after you agree to something,

   b) if you cannot agree to something, or c) if a feature or function is unclear and cannot be clarified at the moment
10. Negotiation is not a contest or a game; it works best when both parties win

Planning Practice

1. Understanding the scope of the project
2. Involve the customer in the planning activity
3. Recognize that planning is iterative; things will change
4. Estimate based only on what you know
5. Consider risk as you define the plan
6. Be realistic on how much can be done each day by each person and how well
7. Adjust granularity as you define the plan
8. Define how you intend to ensure quality
9. Describe how you intend to accommodate change
10. Track the plan frequently and make adjustments as required

Modelling Practice (Analysis)

1) The information domain of a problem (the data that flows in and out of a system) must be represented and understood
2) The functions that the software performs must be defined
3) The behavior of the software (as a consequence of external events) must be represented
4) The models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion
5) The analysis task should move from essential information toward implementation detail

Modelling Practice (Design)

1) The design should be traceable to the analysis model
2) Always consider the software architecture of the system to be built
3) Design of data is as important as design of processing functions
4) Interfaces (both internal and external) must be designed with care
5) User interface design should be tuned to the needs of the end-user and should stress ease of use

Construction Practice (Before Coding)

1) Understand the problem you are trying to solve
2) Understand basic design principles and concepts
3) Pick a programming language that meets the needs of the software to be built and the environment in which it will operate
4) Select a programming environment that provides tools that will make your work easier
5) Create a set of unit tests that will be applied once the component you code is completed

Construction Practice (When Coding Begin)

1) Constrain your algorithms by following structured programming practices
2) Select data structures that will meet the needs of the design
3) Understand the software architecture and create interfaces that are consistent with it
4) Keep conditional logic as simple as possible
5) Create nested loops in a way that makes them easily testable
6) Select meaningful variable names and follow other local coding standards
7) Write code that is self-documenting
8) Create a visual layout (e.g., indentation and blank lines) that aids code understanding

Construction Practice (When Coding Ends)

1) Conduct a code walkthrough
2) Perform unit tests (black-box and white-box) and correct errors you have uncovered
3) Refactor the code

Testing Practice

1) All tests should be traceable to the software requirements
2) Tests should be planned long before testing begins
3) The Pareto principle applies to software testing
   - 80% of the uncovered errors are in 20% of the code
4) Testing should begin “in the small” and progress toward testing “in the large”
   - Unit testing --> integration testing --> validation testing --> system testing

Deployment Practice

1) Customer expectations for the software must be managed
   - Be careful not to promise too much or to mislead the user
2) A complete delivery package should be assembled and tested
3) A support regime must be established before the software is delivered
4) Appropriate instructional materials must be provided to end users
5) Buggy software should be fixed first, delivered later

Video links for SDLC

https://www.youtube.com/watch?v=DRDD7UWX2y4

https://www.youtube.com/watch?v=i-QyW8D3ei0

Technology in SDLC

Confluence: Online tool used for capturing and sharing information
Jira: Online Bug/Issue/Task/Project Tracking system
Jira Agile: Add-on plugin for Jira which provides enhanced capability for agile based projects
Git: Source Code Repository/Version Control system
Stash: Online Git Repository Manager
Source Tree: Git client for Macintosh and Windows computers

Assessment 1

1. What is Software Engineering?

Ans : _____

2. What are all the practices of Software Engineering?

Ans : _____

References

1.Roger S.Pressman, Software engineering- A practitioner‘s Approach, 10th Edition, McGraw-Hill, 2017.

2.Ken Schawber, Mike “Agile Software Development with Scrum” Pearson Education, 2nd Edition, 2015.

Thank You