10. Software Maintenance and Maintenance Process Model

CSE333: Software Engineering

Learning Outcomes

- Understand the stages of testing from testing, during development to acceptance testing by system customers.
- have been introduced to techniques that help you choose test cases that are geared to discovering program defects.
- Understand test-first development, where you design tests before writing code and run these tests automatically.

Software Maintenance

- Software Maintenance is the process of modifying a software product after it has been delivered to the customer.
- The main purpose of software maintenance is to modify and update software application after delivery to correct faults and to improve performance.
- The purpose of **software maintenance** is to preserve the value of software over time, which can accomplished by:
  - Expanding the customer base.
  - Enhancing software's capabilities.
  - Omitting obsolete capabilities.
  - Employing newer technology.

Categories Software Maintenance

- Basic software maintenance includes optimization, error correction, and enhancement of existing features, which combine together to make the software abreast with the latest changes and demands of the software industry.

1. **Corrective Maintenance:**
   - Corrective Maintenance is a reactive process that is **focused on fixing failures in the system.**
   - It refers to the modification and enhancement done to the coding and design of a software to fix errors or defects detected by the user or concluded by error user report.
   - Corrective Maintenance is further divided into two types:
     - **Emergency Repairs.**
     - **Scheduled Repairs.**

Categories Software Maintenance(Cont..)

2. Adaptive Maintenance: Adaptive Maintenance is initiated as a consequence of internal needs, like moving the software to a different hardware or software platform compiler, operating system or new processor and to match the external completion and requirements.

3. Perfective Maintenances: The process of perfective maintenance includes making the product faster, cleaner structured, improving its reliability and performance, adding new features, and more.

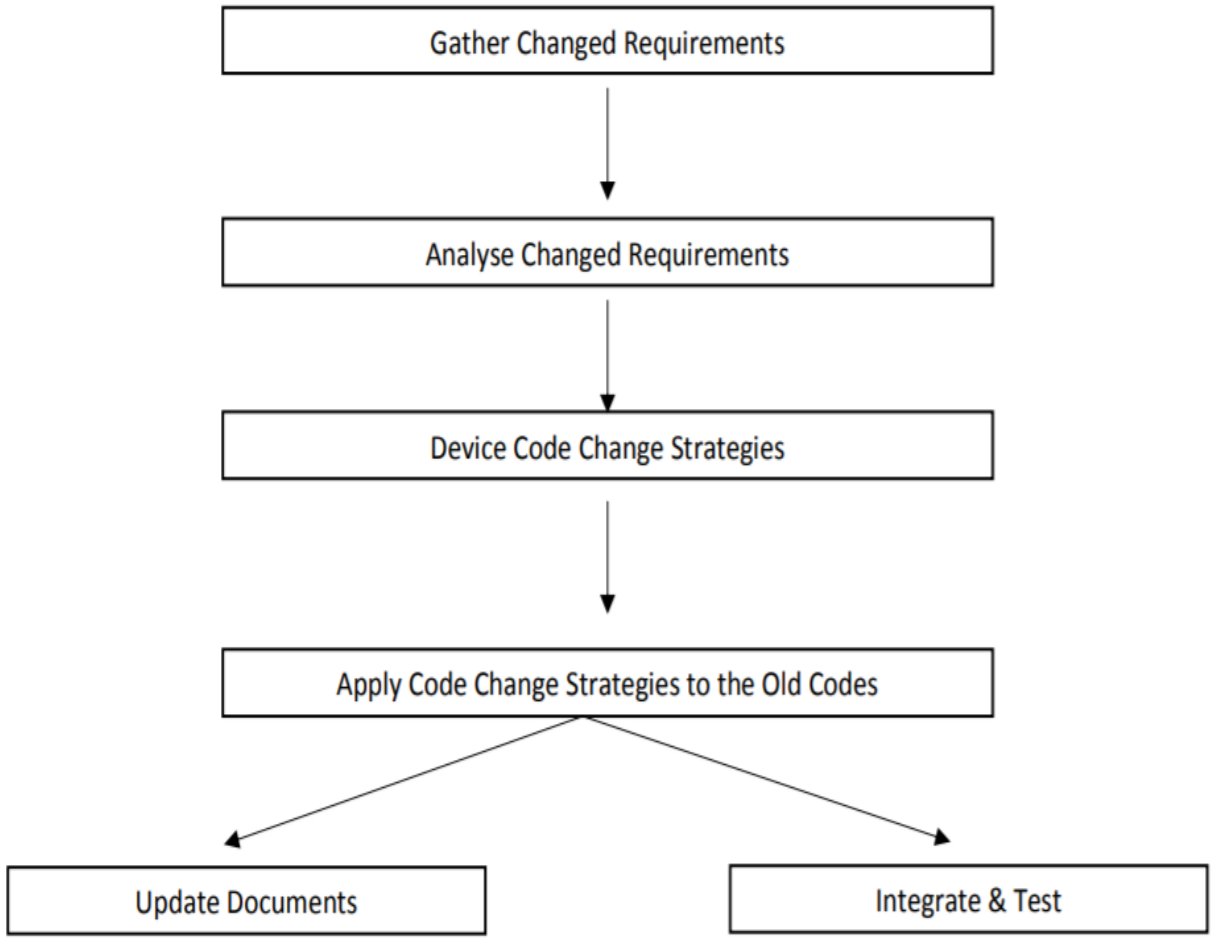
4. Preventive Maintenance: The objective of Preventive Maintenance is to attend problems, which may seem insignificant but can cause serious issues in the future.

Maintenance Process Models

- Two broad categories of process models for software maintenance can be proposed.
- The first model is preferred for projects involving small reworks where the code is changed directly and the changes are reflected in the relevant documents later. This maintenance process is graphically presented **Process Model-1**.
- The second process model for software maintenance is preferred for projects where the amount of rework required is significant. This maintenance process is graphically presented in Process Model-2.

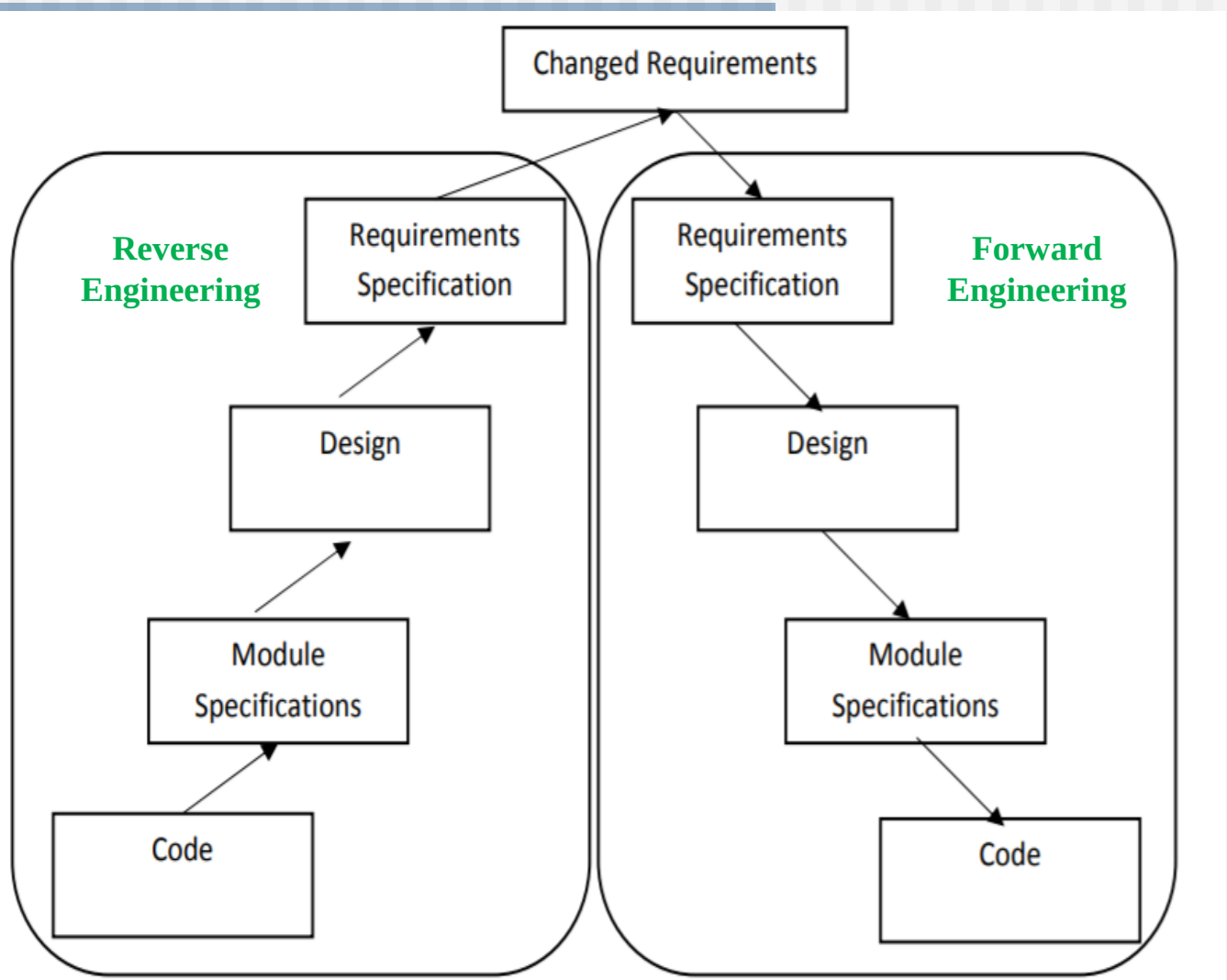
Maintenance Process Model 1

- In this approach, the project starts by gathering the requirements for changes. The requirements are next analyzed to formulate the strategies to be adopted for code change.
- At this stage, the association of at least a few members of the original development team goes a long way in reducing the cycle team, especially for projects involving unstructured and inadequately documented code.
- The availability of a working old system to the maintenance engineers at the maintenance site greatly facilitates the task of the maintenance team as they get a good insight into the working of the old system and also can compare the working of their modified system with the old system.
- Also, debugging o f the reengineered system becomes easier as the program traces of both the systems can be compared to localize the bugs.



Maintenance Process Model 2

- The second process model for software maintenance is preferred for projects where the amount of rework required is significant.
- This approach can be represented by a reverse engineering cycle followed by a forward engineering cycle. Such an approach is also known as software reengineering.
- During the reverse engineering, the old code is analyzed to extract the module specifications. The module specifications are then analyzed to produce the design. The design is analyzed to produce the original requirements specification. The change requests are then applied to this requirements specification to arrive at the new requirements specification. At the design, module specification, and coding a substantial reuse is made from the reverse engineered products



Estimation of approximate Maintenance Cost

- It is well known that maintenance efforts require about 60% of the total life cycle cost for a typical software product.
- However, maintenance costs vary widely from one application domain to another.
- Boehm [1981] proposed a formula for estimating maintenance costs as part of his COCOMO cost estimation model. Boehm's maintenance cost estimation is made in terms of a quantity called the Annual Change Traffic (ACT).

Estimation of Approximate Maintenance Cost

Boehm's maintenance cost estimation is made in terms of a quantity called the **Annual Change Traffic (ACT).** Boehm defined ACT as the fraction of a software product's source instructions which undergo change during a typical year either through addition or deletion.

$$ACT = \frac{KLOC_{added} + KLOC_{deleted}}{KLOC_{total}}$$

where, KLOC added is the total kilo lines of source code added during maintenance.
KLOC deleted is the total kilo lines of source code deleted during maintenance.

Thus, the code that is changed, should be counted in both the code added and the code deleted. The annual change traffic (ACT) is multiplied with the total development cost to arrive at the maintenance cost:

$$\text{maintenance cost} = ACT \times \text{development cost}.$$

Exercise

Find the maintenance cost of a software product where there are a total of 55000 lines of coding. During maintenance 80000 lines of coding was added and 3500 lines of coding was deleted. Development cost of the project was 5 lacs BDT.

References

1. **Software Engineering A practitioner's Approach** by Roger S. Pressman, 7th edition, McGraw Hill, 2010.

2. **Lecture Notes on:** https://docplayer.net/8258886-Lecture-notes-on-software-engineering-course-code-bcs-306.html