CSE 6324

Advanced Topic in Software Engineering

Farnaz Farahanipad

Software Maintenance

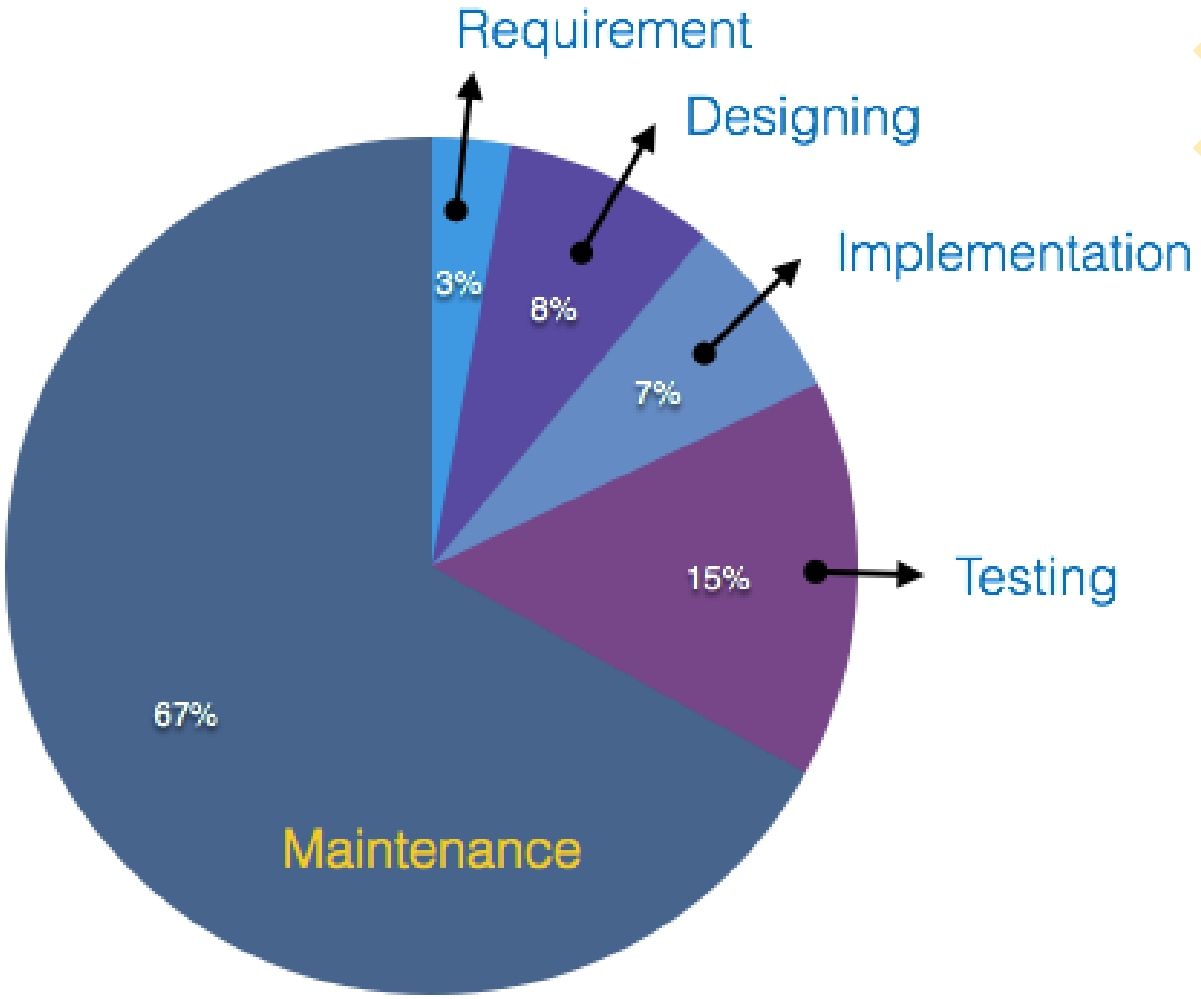
- Software maintenance refers to the process of modifying, updating, and enhancing a software application or system **after** its initial development and deployment. It is an essential component of the software development lifecycle, as it ensures that the software continues to meet the evolving needs of its users and remains relevant and effective over time.

Need for Software Maintenance

- **Market Conditions** - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification.
- **Client Requirements** - Over the time, customer may ask for new features or functions in the software.
- **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.
- **Organization Changes** - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

Maintenance Cost

A study on estimating software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle.



Real-world Factors Affecting Maintenance Cost

- Staff turnover
  - no turnover usually means lower maintenance costs
- Contractual responsibility
  - developers may have no contractual obligation to maintain the delivered system and no incentive to design for future change
- Staff skills
  - maintenance staff are often inexperienced and have limited domain knowledge
- Program age and structure
  - as programs age their structure deteriorates, they become harder to understand and change

Maintenance can be though

- Limited understanding of hardware and software (maintainer).
- Management priorities (maintenance may be low priority).
- Testing difficulties (finding problems).
- Morale problems (maintenance is boring).

Maintenance can be though

- Someone else’s program.
- Developer not available.
- Not design for change.
- Proper documentation does not exist.

Types of Maintenance

- Corrective Maintenance (21%)
  - making changes to repair defects
- Adaptive Maintenance (25%)
  - making changes to adapt software to external environment changes (hardware, business rules, OS, etc.)
- Perfective Maintenance (50%)
  - extending system beyond its original functional requirements
- Preventative Maintenance (4%)
  - modifying work products so that they are more easily corrected, adapted, or enhanced

Key Activities in Software Maintenance

1.Understanding the need for maintenance: The first step in software maintenance is recognizing the need for it. This may be due to changes in user requirements, new features or functionality needed, or the need to fix bugs or security vulnerabilities.

2.Establishing a maintenance plan: A maintenance plan outlines the specific tasks, timelines, and resources needed for maintaining the software application or system. This includes identifying the maintenance team, establishing maintenance processes, and determining the maintenance schedule.

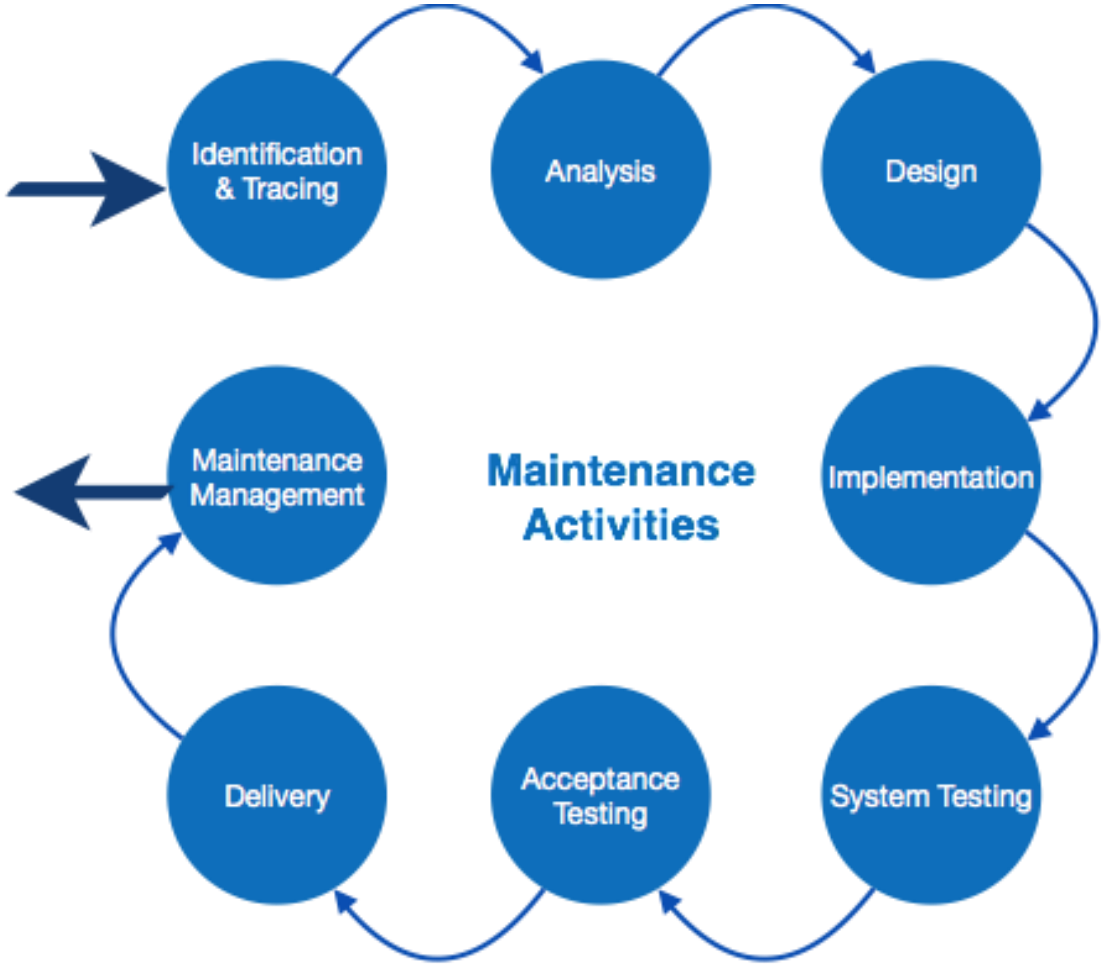
3.Prioritizing maintenance tasks: Not all maintenance tasks are equally important. It is essential to prioritize maintenance tasks based on their impact on the software's performance, functionality, and user experience.

4.Implementing updates and patches: Updating software applications and systems regularly is essential for maintaining their performance, security, and reliability. This includes implementing updates and patches for software components and dependencies.

Key Activities in Software Maintenance

5. **Testing and validating changes:** Before implementing any changes to the software application or system, it is critical to test and validate them thoroughly to ensure that they do not introduce new defects or issues.
6. **Documenting maintenance activities:** Maintaining detailed documentation of maintenance activities is essential for tracking changes, identifying issues, and ensuring that the software remains compliant with regulations and standards.
7. **Engaging users and stakeholders:** Maintaining open communication with users and stakeholders is critical for understanding their needs and expectations, identifying issues, and ensuring that the software continues to meet their evolving needs.
8. Continuous improvement: Continuous improvement is a key aspect of software maintenance. Regularly evaluating the software's performance, functionality, and user experience can help identify areas for improvement and ensure that the software remains relevant and effective over time.

Maintenance Activities



IEEE provides a framework for sequential maintenance process activities.

Maintenance Developers Task

- Understand system.
- Locate information in documentation.
- Keep system documentation up to date.
- Extend existing functions.
- Add new functions.
- Find sources of errors.
- Correct system errors.
- Answer operations questions.
- Restructure design and code.
- Delete obsolete design and code.
- Manage changes.

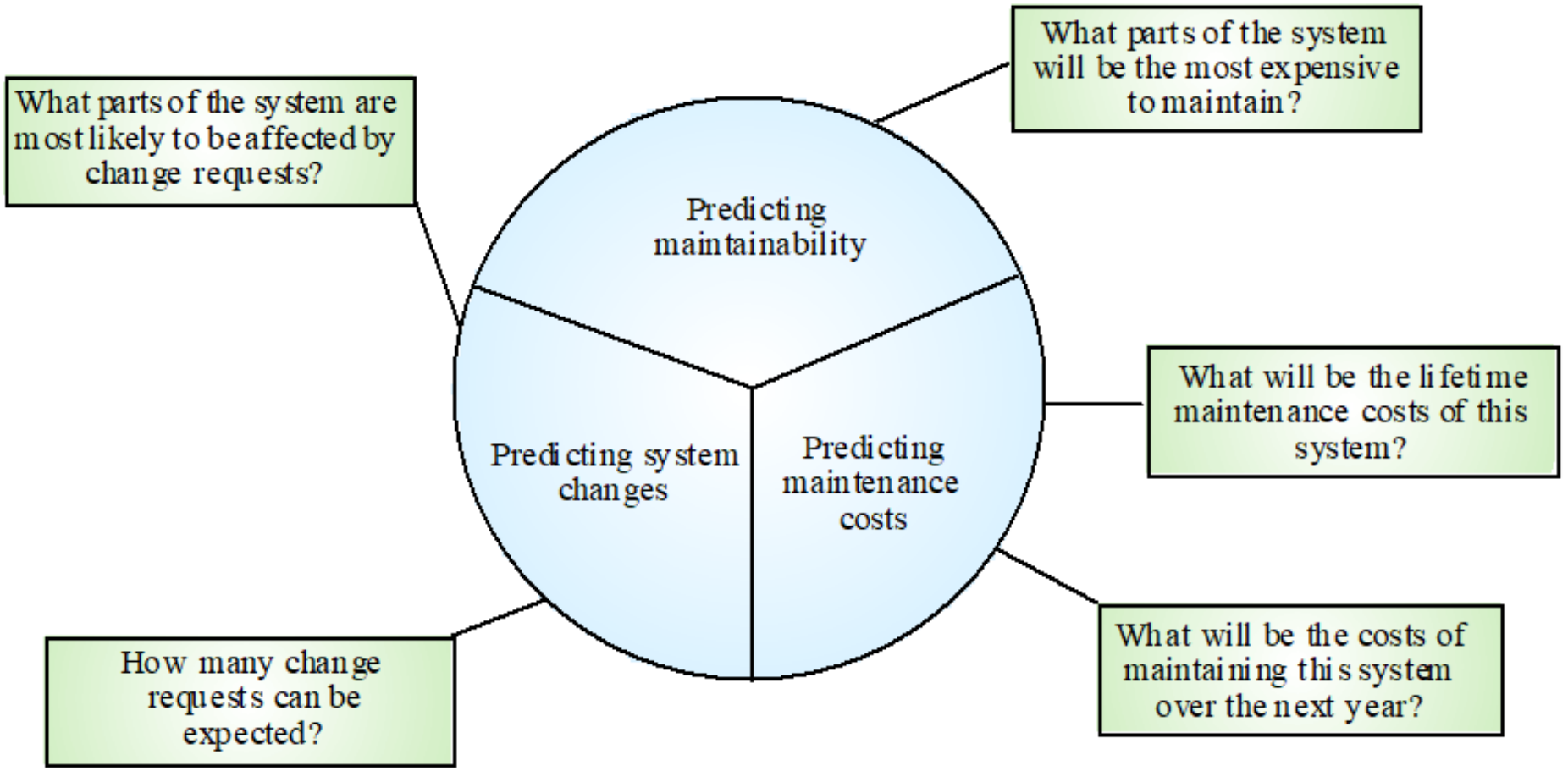
Maintenance Prediction

- Predicting system changes
- Predicting maintainability
- Predicting the cost

Maintenance Prediction

- which parts of the system may cause problems and have high maintenance costs
- Change acceptance depends on the maintainability of the components affected by the change
- Maintenance costs depends on number of changes
- Costs of change depend on maintainability

Maintenance Prediction



Maintenance Complexity Facts

- Predictions of maintainability can be made by assessing component complexities.

Maintenance complexity depends on…

Maintenance Complexity Metrics

Maintenance complexity depends on:

- complexity of control structures
- complexity of data structures
- module size

Maintenance Process Metrics

Maintainability measurements

- **number of requests** for corrective maintenance
- **average time** required for impact **analysis**
- **average time** to **implement** a change request
- number of **outstanding change** requests

Tools and Technologies for Maintenance

- Version control systems (e.g., Git, SVN)
- Issue tracking systems (e.g., Jira, Bugzilla)
- Automated testing frameworks (e.g., JUnit, Selenium)
- Code analysis tools (e.g., SonarQube, Coverity)
- Documentation tools (e.g., Doxygen, Sphinx)

Best practices for effective software maintenance

- Regular code reviews
- Version control and documentation
- Automated testing and continuous integration
- Prioritization of maintenance tasks
- Knowledge management and training

Advantages of Software Maintenance

1.Improved software quality: Regular software maintenance helps to ensure that the software is functioning correctly and efficiently, and that it continues to meet the needs of the users.
2.Enhanced security: Maintenance can include security updates and patches, helping to ensure that the software is protected against potential threats and attacks.
3.Increased user satisfaction: Regular software maintenance helps to keep the software up-to-date and relevant, leading to increased user satisfaction and adoption.
4.Extended software life: Proper software maintenance can extend the life of the software, allowing it to be used for longer periods of time and reducing the need for costly replacements.
5.Cost savings: Regular software maintenance can help to prevent larger, more expensive problems from occurring, reducing the overall cost of software ownership.

Disadvantages of Software Maintenance

1. **Cost**: Software maintenance can be time-consuming and expensive and may require significant resources and expertise.
2. **Schedule disruptions**: Maintenance can cause disruptions to the normal schedule and operations of the software, leading to potential downtime and inconvenience.
3. **Complexity**: Maintaining and updating complex software systems can be challenging, requiring specialized knowledge and expertise.
4. **Risk of introducing new bugs:** The process of fixing bugs or adding new features can introduce new bugs or problems, making it important to thoroughly test the software after maintenance.
5. **User resistance:** Users may resist changes or updates to the software, leading to decreased satisfaction and adoption.

Future Trends in Software Maintenance

- Artificial intelligence (AI) and machine learning (ML): Leveraging AI and ML algorithms for automated code analysis, defect prediction, and intelligent decision-making in software maintenance tasks.
- Automated maintenance and repair: Advancements in automated repair techniques, such as automated bug fixing, code refactoring, and self-healing systems, to reduce manual intervention and improve software quality.
- Autonomous systems and self-adaptive software: Development of autonomous systems capable of self-monitoring, self-diagnosing, and self-healing to adapt to changing conditions and requirements autonomously.

Questions: