SWE-417 SOFTWARE RE-ENGINEERING

Software Engineering Department

Sir Syed University of Engineering & Technology

Week No. 1

CONTENTS

- Evolution versus Maintenance

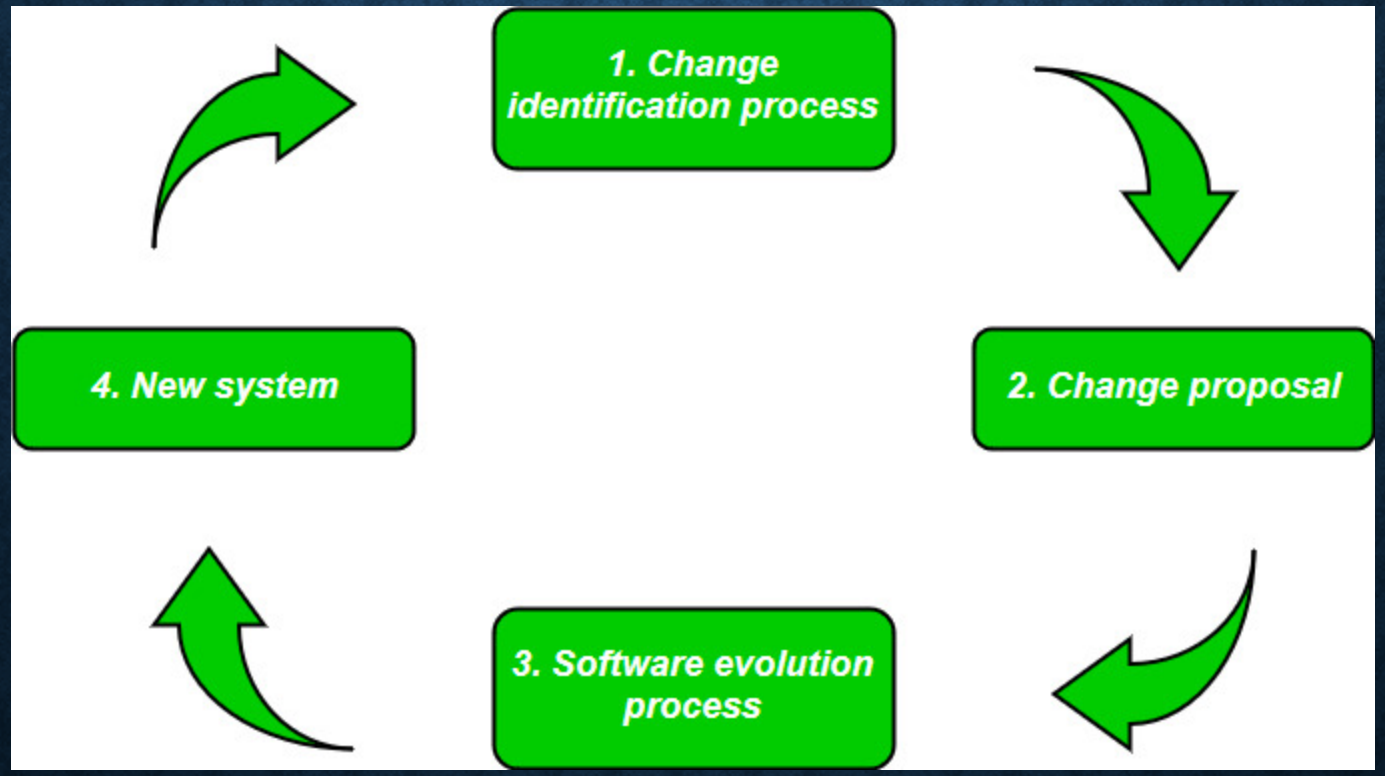
EVOLUTION VERSUS MAINTENANCE

- The terms evolution and maintenance are used interchangeably.
- However there is a semantic difference.
- Lowell Jay Arthur distinguish the two terms as follows:
  - “Software maintenance means to preserve from failure or decline.”
  - “Software evolution means a continuous change from lesser, simpler, or worse state to a higher or better state.”
- Keith H. Bennett and Lie Xu use the term:
  - “Maintenance for all post-delivery support and evolution to those driven by changes in requirements.”

SOFTWARE EVOLUTION

- In 1965, Mark Halpern used the term evolution to define the dynamic growth of software. The term evolution in relation to application systems took gradually in the 1970s.
- Scope: Large, sometimes requiring architectural redesigns, addition of new modules, or technology upgrades.
- Example:
  - Adding new functionality to meet emerging business needs.
  - Migrating a system to a new platform or adopting new technologies (e.g., moving from on-premises infrastructure to the cloud).
  - Refactoring code to improve maintainability and performance.

SOFTWARE EVOLUTION PROCESS



SOFTWARE EVOLUTION

- Lehman formulated a set of observations that he called laws of evolution.
- These laws are the results of studies of the evolution of large-scale proprietary or closed source system (CSS).
- The laws concern what Lehman called E-type systems:
  - “Monolithic systems produced by a team within an organization that solve a real world problem and have human users.”

SOFTWARE EVOLUTION

- In a monolithic software architecture, the entire application is designed and structured as a single, unified codebase, with all the components tightly integrated and interconnected.
- A software system is called "monolithic" if it has a monolithic architecture, in which functionally distinguishable aspects (for example data input and output, data processing, error handling, and the user interface) are all interwoven, rather than containing architecturally separate components.

SOFTWARE EVOLUTION: LAWS OF LEHMAN

- **Continuing change (1st)** – A system will become progressively less satisfying to its user over time, unless it is continually adapted to meet new needs.
- **Increasing complexity (2nd)** – A system will become progressively more complex, unless work is done to explicitly reduce the complexity.

SOFTWARE EVOLUTION: LAWS OF LEHMAN

- **Self-regulation (3rd)** – The process of software evolution is self regulating with respect to the distributions of the products and process artifacts that are produced.
- **Conservation of organizational stability (4th)** – The average effective global activity rate on an evolving system does not change over time, that is the average amount of work that goes into each release is about the same.

SOFTWARE EVOLUTION: LAWS OF LEHMAN

- **Conservation of familiarity (5th)** – The amount of new content in each successive release of a system tends to stay constant or decrease over time.
- **Continuing growth (6th)** – The amount of functionality in a system will increase over time, in order to please its users.

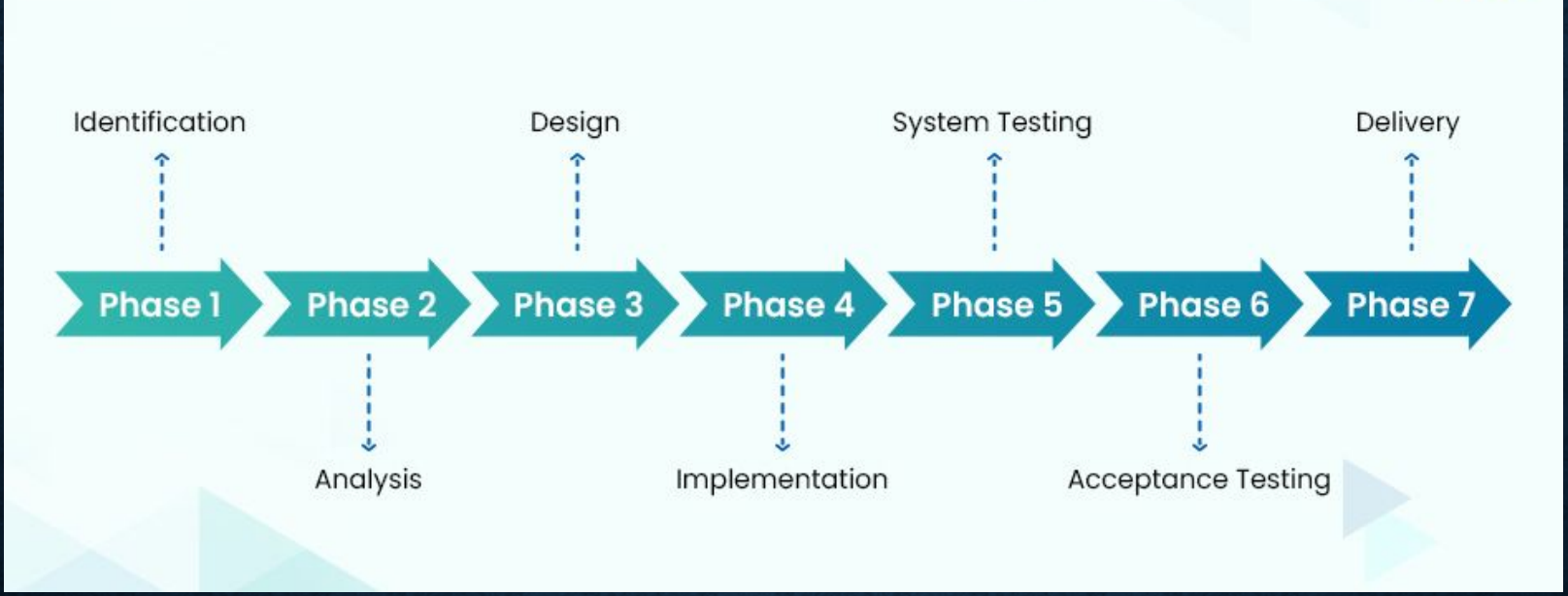
SOFTWARE EVOLUTION: LAWS OF LEHMAN

- **Declining quality (7th)** – A system will be perceived as losing quality over time, unless its design is carefully maintained and adapted to new operational constraints.
- **Feedback system (8th)** – Successfully evolving a software system requires recognition that the development process is a multi-loop, multi-agent, multi-level feedback system.

SOFTWARE MAINTENANCE

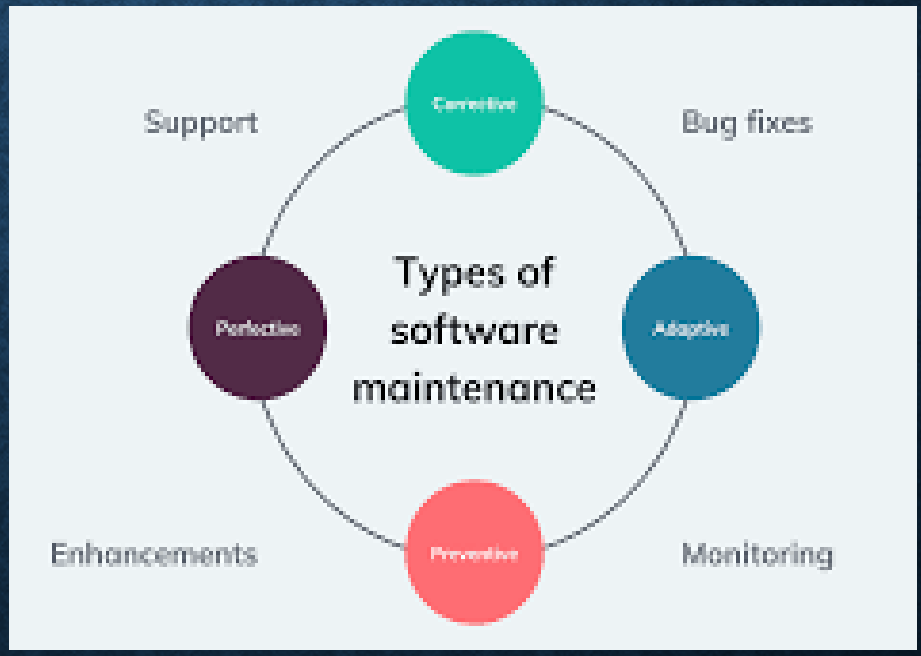
- There will always be defects in the delivered software application because software defect removal and quality control are not perfect. Therefore, software maintenance is needed to repair these defects in the released software.
- Scope: Typically smaller in scale and aimed at keeping the system functional.
- Example:
  - Fixing software defects (bug fixes).
  - Minor adjustments to improve usability or accommodate small business changes.
  - Updating documentation or ensuring compliance with updated standards.

SOFTWARE MAINTENANCE PROCESS



SOFTWARE MAINTENANCE

- SE. Burton wanson defined few types of software maintenance:
  - Corrective,
  - Adaptive,
  - Perfective,
  - Preventive.



SOFTWARE MAINTENANCE

- Swanson's classification of maintenance activities is intention based because the maintenance activities reflect the intents of the developer to carry out specific maintenance tasks on the system.
- In the intention-based classification of maintenance activities, the intention of an activity depends upon the motivations for the change.
  - Activities to make corrections: If there are inconsistencies between the expected behavior of a system and the actual behavior, then some activities are performed to eliminate or reduce the inconsistencies.
  - Activities to make enhancements: A number of activities are performed to implement a change to the system, thereby changing the behavior or implementation of the system.

SOFTWARE MAINTENANCE

- This category of activities is further refined into three subcategories:
  - Enhancements that change existing requirement,
  - Enhancements that add new system requirements, and
  - Enhancements that change the implementation but not the requirements.

COMMERCIAL OFF-THE-SHELF (COTS)

- Commercial Off-The-Shelf (COTS) refers to software or hardware products that are readily available for purchase from third-party vendors or manufacturers.
- These products are designed and developed for the general market and are not custom-built for a specific organization or customer.
- COTS products are typically pre-packaged, come with standardized features, and are intended to meet the needs of a wide range of users and applications.
- **Key characteristics of COTS products:** Readily Available, Standardized Features, Lower Development Costs, Licensing, Updates and Upgrades etc.

EXAMPLES

- Commercial Off-The-Shelf (COTS) based systems can be found in various industries and domains.
- **Office Productivity Software**:
  - Microsoft Office Suite (Word, Excel, PowerPoint, etc.)
  - Google Workspace (formerly G Suite)
- **Operating Systems**:
  - Microsoft Windows, macOS , Linux distributions (Ubuntu, Red Hat, etc.)
- **Content Management Systems (CMS)**:
  - WordPress, Drupal, Joomla

MAINTENANCE OF COTS-BASED SYSTEMS

- Maintenance of COTS (Commercial Off-The-Shelf) based systems involves a set of activities aimed at ensuring the continued functionality, reliability, and security of software applications or systems that rely on third-party COTS components.
- **Key aspects of maintaining COTS-based systems:**

1. Monitoring and Evaluation:

- Regularly monitor the performance and health of the COTS components within the system.
- Keep track of vendor updates, patches, and security bulletins for the COTS software.

2. Security Updates and Patches:

- Apply security patches and updates provided by the COTS vendor promptly to address vulnerabilities.
- Ensure that the system is protected against known security threats.

3. Compatibility Testing:

- Test COTS updates for compatibility with other system components and customizations.

4. **Backup and Recovery Planning:**

- Implement robust backup and disaster recovery plans to protect data.

5. User Support and Training:

- Provide ongoing user support, including training for new features or updates.

SUMMARY

- Software evolution and Software Maintenance are key activities in the software lifecycle, but they serve different purposes and are distinguished by their nature and scope.
- Evolution focuses on growth and adaptability, while maintenance focuses on stability and reliability.
- Both activities ensure that software continues to provide value over time.