Software Maintenance

By

Dr. J.V. Joshua

Introduction

- Software development efforts result in the delivery of a software product that satisfies user requirements. Accordingly, the software product must change or evolve.
- Once in operation, defects are uncovered, operating environments change, and new user requirements surface.
- The maintenance phase of the life cycle begins following a warranty period or post-implementation support delivery

Introduction cont.

- Software maintenance is an integral part of a software life cycle. However, it has not received the same degree of attention that the other phases have.
- Historically, software development has had a much higher profile than software maintenance in most organisations. This is now changing, as organisations strive to squeeze the most out of their software development investment by keeping software operating as long as possible.
- The open source paradigm has brought further attention to the issue of maintaining software artifacts developed by others

Introduction cont.

- Software maintenance is defined as the totality of activities required to provide cost-effective support to software system.
- The objective of software maintenance is to modify existing software while preserving its integrity.
- Software maintenance sustains the software product throughout its life cycle (from development to operations).

Maintenance Models

- Three process reuse oriented models have been proposed:
  1. **Quick fix model**
  2. **Iterative enhancement model**
  3. **Full reuse model**

Maintenance Models

1. **Quick fix model:** In this model, necessary changes are quickly made to the code and then to the accompanying documentation.

Quick fix model steps:

i. source code is modified to fix the problem;
ii. necessary changes are made to the relevant documents; and
iii. the new code is recompiled to produce a new version

Maintenance Models cont.

2. Iterative enhancement model: In this model, first changes are made to the highest level documents. Eventually, changes are propagated down to the code level.

The model works as follows:

- It begins with the existing system's artifacts, namely, **requirements, design, code, test, and analysis** documents.
- It revises the highest-level documents affected by the changes and propagates the changes down through the lower-level documents.
- The model allows maintainers to redesign the system, based on the analysis of the existing system

Maintenance Models cont.

3. Full reuse model: In this model, a new system is built from components of the old system and others available in the repository.

- In the **full reuse model**, reuse is explicit and the following activities are performed:
  - identify the components of the old system that are candidates for reuse
  - understand the identified system components.
  - modify the old system components to support the new requirements.
  - integrate the modified components to form the newly developed system

Change Mini-Cycle Model

Software change is a process that may introduce new requirements to the existing system, or may need to alter the software system if requirements are not correctly implemented. In order to capture this, an evolutionary model known as change mini-cycle.

Change requests:

- defect report and
- enhancement request

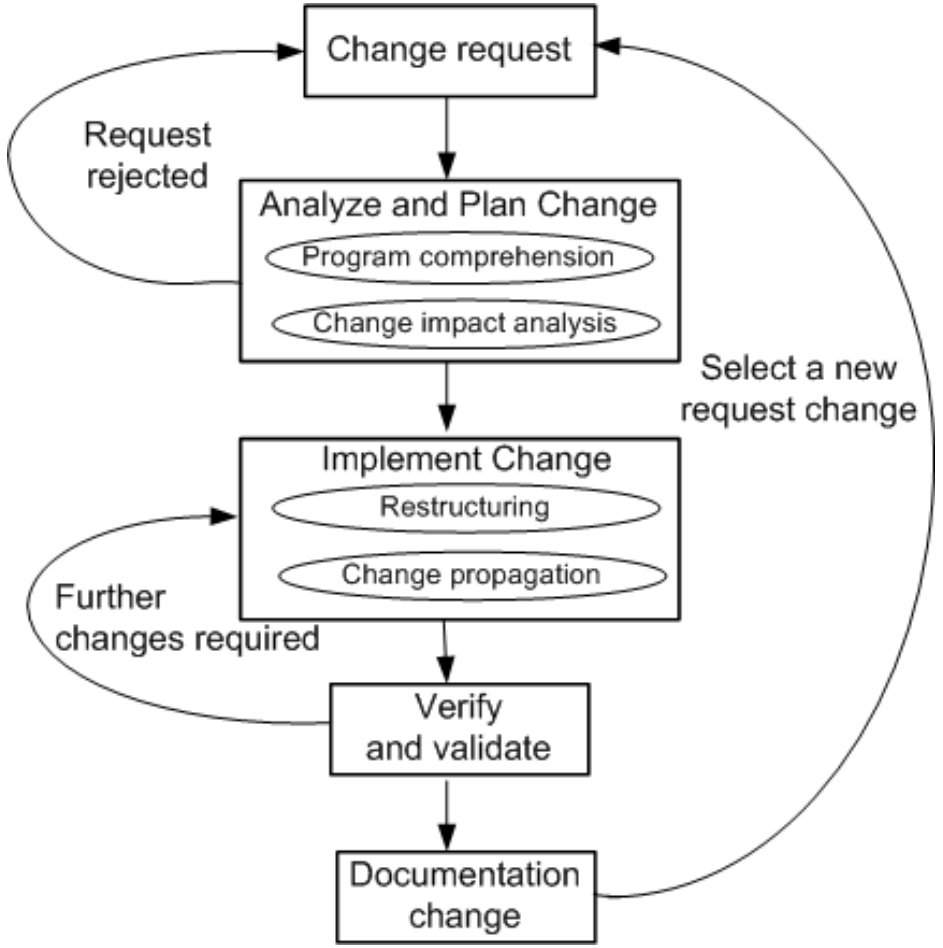


Fig. The change mini-cycle ©Springer, 2008

Maintenance Models cont.

- **Closed Source Software(CSS**) System
  - systems developed under industrial software process
  - proprietary software distributed under a licensing agreement to authorized users with private modification, copying, and republishing restrictions
  - the source code is not shared with the public for anyone to look at or change
  - Examples
    - Microsoft Office, Adobe Acrobat, McAfee anti-virus, windows OS

Maintenance Models cont.

- **Free and open source software(FOSS)**
  - It is both free and open
  - It is liberally licensed to grant users the right to use, copy, study, change, and improve its design through availability of its source code
  - The FOSS movement is attributed to Richard Stallman, who started the GNU project in 1984, and a supporting organization, the Free Software Foundation
  - GNU is a recursive acronym for "**GNU's Not Unix**", chosen because GNU's design is Unix-like, but differs from Unix by being free software and containing no Unix code
  - Examples
    - Mozilla Firefox web browser, PHP, Python, Apache, Open office

- **Commercial off**-the-**shelf(COTS)**
  - software that are ready-made and available for sale to the general public
  - For example, **Microsoft Office** is a COTS product that is a packaged software solution for businesses
  - Outlook
  - Accounting software e.g. Sage. Tally, Quickbook,Navision etc.
  - Games

Maintenance Models cont.

- The Staged Model for CSS
- The Staged Model for FOSS

Maintenance Models cont.

- Rajlich and Bennett defined a **simple *staged* model** to represent the traditional commercial Closed Source Software (CSS) life cycle.
- Their model comprises a sequence of five stages
  - ***initial development***. Develop the first functioning version of the software.
  - ***Evolution***. The developers improve the functionalities and capabilities of the software to meet the needs and expectations of the customer
  - ***Servicing***. The developers only fix minor and emergency defects, and no major functionality is included
  - ***Phaseout***. In this phase, no more servicing is undertaken, while the vendors seek to generate revenue as long as possible
  - ***Closedown***. The software is withdrawn from the market, and customers are directed to migrate to a replacement

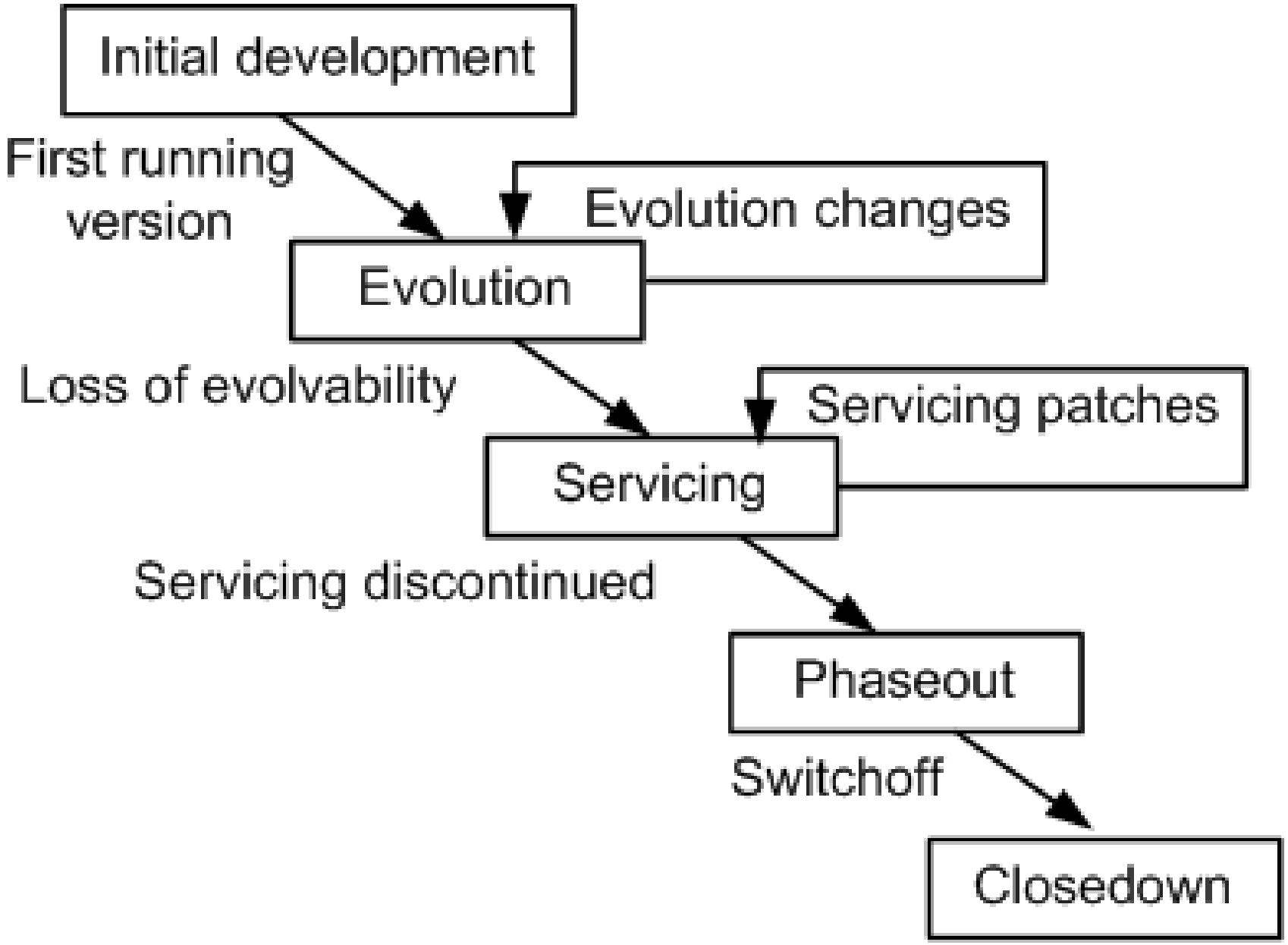


Fig. Stage Model CSS

Maintenance Models cont.

The Staged Model for FOSS

- Three major differences are identified between CSS systems and FOSS systems.
- In Figure below, the rectangle with the label “Initial development” has been visually highlighted because it can be the only initial development stage in the evolution of FOSS systems. In other words, it does not have any evolution track for FOSS system.
- With some systems that were analyzed, after a transition from Evolution to Servicing, a new period of evolution was observed. This possibility is depicted in Figure as a broken arc from the Servicing stage to the Evolution stage.

- In general, the active developers of FLOSS systems get frequently replaced by new developers. Therefore, the dashed line in Figure exhibits this possibility of a transition from Phaseout stage to Evolution stage.

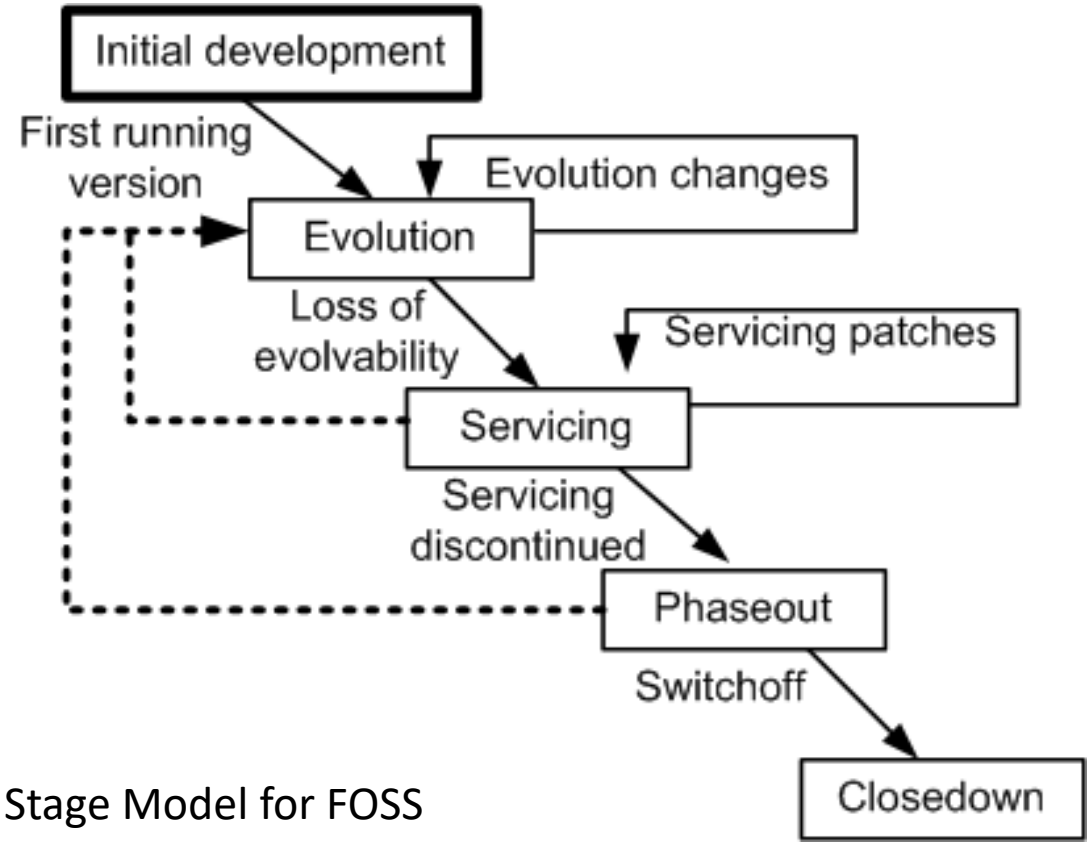


Fig. Stage Model for FOSS

Need for Maintenance

- Maintenance is needed to ensure that the software continues to satisfy user requirements.
- Maintenance is applicable to software that is developed using any software life cycle model (for example, spiral or linear).
- Software products change due to corrective and non-corrective software actions

Need for Maintenance cont.

- Maintenance must be performed in order to:
  - correct faults
  - improve the design
  - implement enhancements
  - interface with other software
  - adapt programs so that different hardware, software, system features, and telecommunications facilities can be used
  - migrate legacy software and
  - retire software

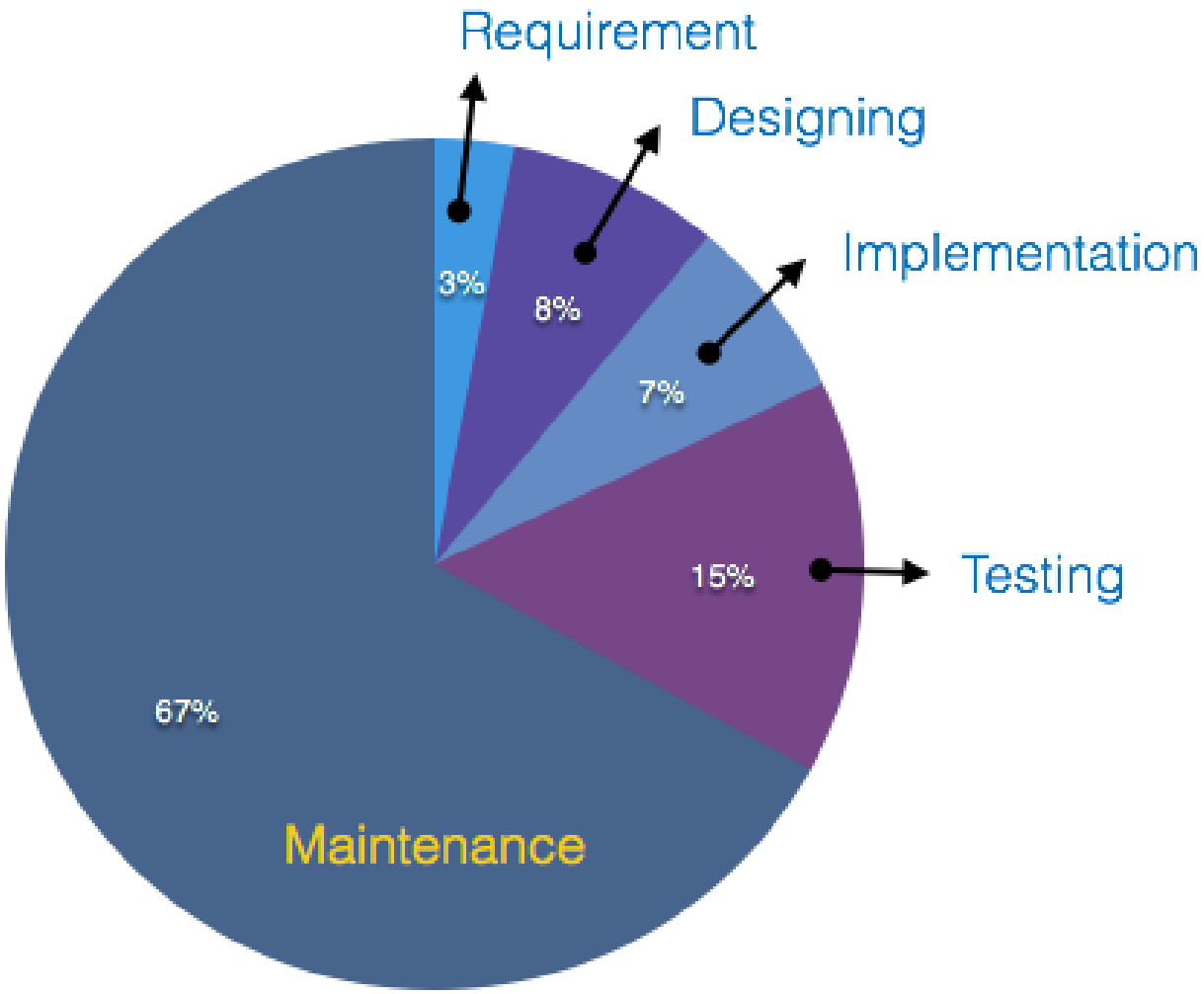
Need for Maintenance cont.

- Five key characteristics comprise the maintainer’s activities
  - maintaining control over the software’s day-to-day functions
  - maintaining control over software modification
  - perfecting existing functions
  - identifying security threats and fixing security vulnerabilities and
  - preventing software performance from degrading to unacceptable levels

Costs of Maintenance

- Maintenance consumes a major share of the financial resources in a software life cycle.
- Reports suggest that the cost of maintenance is high. 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
- Understanding the categories of software maintenance helps to understand the structure of software maintenance costs.
- Also, understanding the factors that influence the maintainability of software can help to contain costs

Cost of Maintenance cont.



Cost of Maintenance cont.

- On an average, the cost of software maintenance is more than 50% of all SDLC phases. There are various factors, which trigger maintenance cost go high, such as:
  - **Real-world factors**
  - **Software-end factors**

- **Real-world factors affecting Maintenance Cost**
  - The standard age of any software is considered up to 10 to 15 years.
  - Older software, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced software on modern hardware.
  - As technology advances, it becomes costly to maintain old software.
  - Most maintenance engineers are newbie and use trial and error method to rectify problem.
  - Often, changes made can easily hurt the original structure of the software, making it hard for any subsequent changes.
  - Changes are often left undocumented which may cause more conflicts in future

Cost of Maintenance cont.

- **Software-end factors affecting Maintenance Cost**
  - Structure of Software Program
  - Programming Language
  - Dependence on external environment(such as policies, competition, process)
  - Staff reliability and availability

Categories of maintenance

Maintenance activities are divided into four groups:

1. Corrective maintenance
2. Adaptive maintenance
3. Perfective maintenance
4. Preventive maintenance

Software Maintenance

- Corrective maintenance: The purpose of corrective maintenance is to correct failures: processing failures and performance failures.
- Examples of corrective maintenance:
  - A program producing a wrong output is an example of processing failure.
  - Similarly, a program not being able to meet real-time requirements is an example of performance failure.
- The process of corrective maintenance includes isolation and correction of defective elements in the software.
- There is a variety of situations that can be described as corrective maintenance such as correcting a program that aborts or produces incorrect results.
- Basically, corrective maintenance is a reactive process, which means that corrective maintenance is performed after detecting defects with the system.

Software Maintenance

- Adaptive maintenance: The purpose of adaptive maintenance is to enable the system to adapt to changes in its data environment or processing environment.
- This process modifies the software to properly interface with a changing or changed environment.
- Adaptive maintenance includes system changes, additions, deletions, modifications, extensions, and enhancements to meet the evolving needs of the environment in which the system must operate.
- Examples of Adaptive maintenance are:
  - changing the system to support new hardware configuration;
  - converting the system from batch to on-line operation; and
  - changing the system to be compatible with other applications.

Software Maintenance

- Perfective maintenance: The purpose of perfective maintenance is to make a variety of improvements, namely, user experience, processing efficiency, and maintainability.
- Examples of perfective maintenance are:
  - the program outputs can be made more readable for better user experience;
  - the program can be modified to make it faster, thereby increasing the processing efficiency;
  - and the program can be restructured to improve its readability, thereby increasing its maintainability.
- Activities for perfective maintenance include restructuring of the code, creating and updating documentations, and tuning the system to improve performance.
- It is also called “reengineering”.

Software Maintenance

- Preventive maintenance: The purpose of preventive maintenance is to prevent problems from occurring by modifying software products.
- Basically, one should look ahead, identify future risks and unknown problems, and take actions so that those problems do not occur.
- Preventive maintenance is very often performed on safety critical and high available software systems.
- The concept of "software rejuvenation" is a preventive maintenance measure to prevent, or at least postpone, the occurrences of failures (crash) due to continuously running the software system.
- It involves occasionally terminating an application or a system, cleaning its internal state, and restarting it.
- Rejuvenation may increase the downtime of the application; however, it prevents the occurrence of more severe failures.

Software Maintenance cont

In summary

- Corrective: *maintaining control over day-to-day functions*
- Adaptive: *maintaining control over system modifications*
- Perfective: *perfecting existing functions*
- Preventive: *preventing system performance from degrading to unacceptable levels*

Software Maintenance Exercises

- One of the credit card companies upgrades its system for handling credit card payments, and this requires a slight change to the type of data that the Gas Station Control System (GSCS) needs to send to it. This situation should

a. lead to a corrective change. b. lead to an adaptive change. c. lead to a perfective change.

d. lead to a preventive change. e. require no maintenance to be performed.

Software Maintenance Exercises

- The gas station owner has stipulated that the GSCS should be able to handle additional gas pumps, if the station decides to invest in them in the future. However, the development team realizes that the way in which it handles concurrency will not scale up if more gas pumps are added at the gas station. This situation should:

a. lead to a corrective change. b. lead to an adaptive change. c. lead to a perfective change.

d. lead to a preventive change. e. require no maintenance to be performed

Software Maintenance Exercises

- An additional service is added for customers at the gas station. (Customers can now rent parking spots.) This situation should

a. lead to a corrective change. b. lead to an adaptive change. c. lead to a perfective change.

d. lead to a preventive change. e. require no maintenance to be performed

Software Maintenance Exercises

- When receipts are printed, if the customer's name exceeds a certain length then the purchase price does not fit on the receipt and is not printed. This situation does not occur very frequently (at most, once a week). This situation should:

a. lead to a corrective change. b. lead to an adaptive change. c. lead to a perfective change.

d. lead to a preventive change. e. require no maintenance to be performed

QUIZ 2 Ten minutes

- **Indicate the types of maintenance in the following Scenario**

1. Isolation and correction of defective elements in the software _____
2. Changing the system to support new hardware configuration _____
3. Converting the system from batch to online operation _____
4. Restructured programs to improve its readability _____
5. Restructured program to achieve good styles to make later program comprehension easier _____