Software Maintenance

Software Maintenance

What is Software Maintenance?

Software Maintenance is a very broad activity that includes error corrections, enhancements of capabilities, deletion of obsolete capabilities, and optimization.

Software Maintenance

Categories of Maintenance

- Corrective maintenance

This refer to modifications initiated by defects in the software.

- Adaptive maintenance

It includes modifying the software to match changes in the ever changing environment.

- Perfective maintenance

It means improving processing efficiency or performance, or restructuring the software to improve changeability. This may include enhancement of existing system functionality, improvement in computational efficiency etc.

Software Maintenance

- Other types of maintenance

There are long term effects of corrective, adaptive and perfective changes. This leads to increase in the complexity of the software, which reflect deteriorating structure. The work is required to be done to maintain it or to reduce it, if possible. This work may be named as **preventive maintenance**.

Software Maintenance

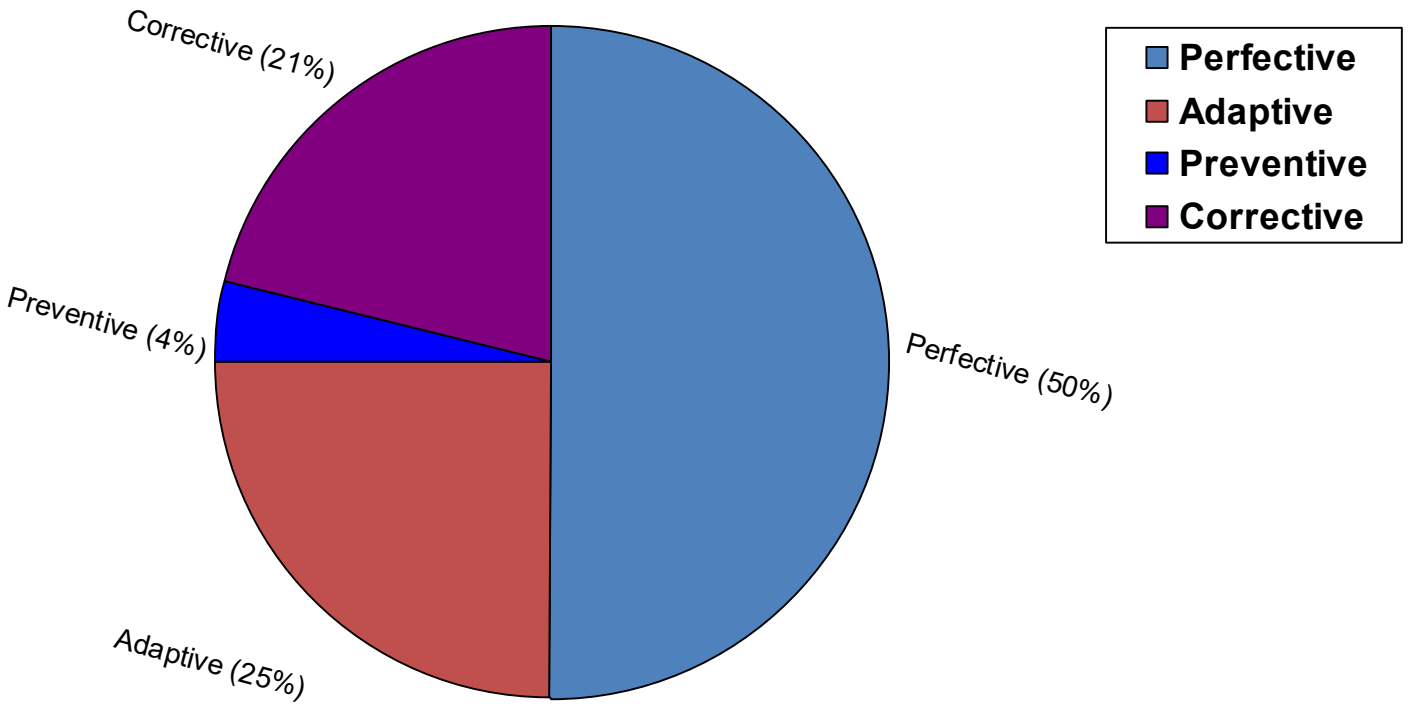

Fig. 1: Distribution of maintenance effort

Software Maintenance

Problems During Maintenance

- Often the program is written by another person or group of persons.
- Often the program is changed by person who did not understand it clearly.
- Program listings are not structured.
- Information gap.
- Systems are not designed for change.

Maintenance is Manageable

A common misconception about maintenance is that it is not manageable.

Report of survey conducted by Lientz & Swanson gives some interesting observations:

1	Emergency debugging	12.4%
2	Routine debugging	9.3%
3	Data environment adaptation	17.3%
4	Changes in hardware and OS	6.2%
5	Enhancements for users	41.8%
6	Documentation Improvement	5.5%
7	Code efficiency improvement	4.0%
8	Others	3.5%

Table 1: Distribution of maintenance effort

Software Maintenance

Potential Solutions to Maintenance Problems

- Budget and effort reallocation
- Complete replacement of the system
- Maintenance of existing system

The Maintenance Process

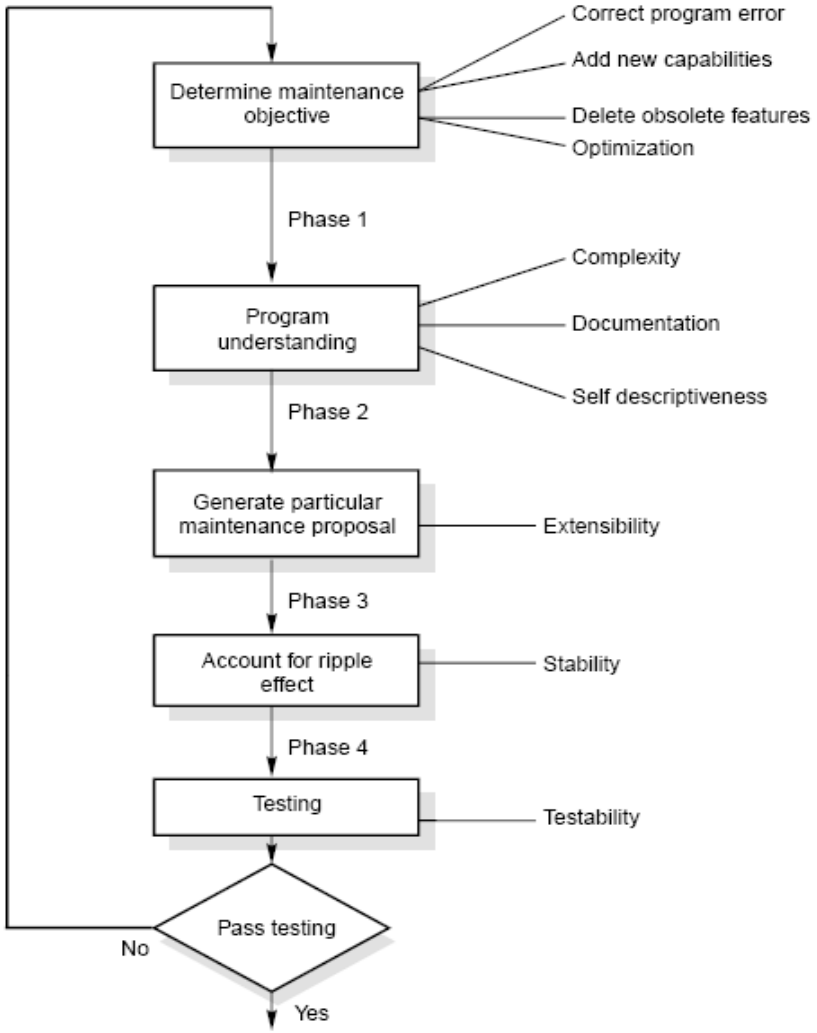

Fig. 2: The software maintenance process

Software Maintenance

- Program Understanding

The first phase consists of analyzing the program in order to understand.

- Generating Particular Maintenance Proposal

The second phase consists of generating a particular maintenance proposal to accomplish the implementation of the maintenance objective.

- Ripple Effect

The third phase consists of accounting for all of the ripple effect as a consequence of program modifications.

Software Maintenance

- Modified Program Testing

The fourth phase consists of testing the modified program to ensure that the modified program has at least the same reliability level as before.

- Maintainability

Each of these four phases and their associated software quality attributes are critical to the maintenance process. All of these factors must be combined to form maintainability.

Estimation of maintenance costs

Phase	Ratio
Analysis	1
Design	10
Implementation	100

Table 3: Defect repair ratio

Software Maintenance

- Belady and Lehman Model

$$M = P + Ke^{(c-d)}$$

where

M : Total effort expended

P : Productive effort that involves analysis, design, coding, testing and evaluation.

K : An empirically determined constant.

c : Complexity measure due to lack of good design and documentation.

d : Degree to which maintenance team is familiar with the software.

Software Maintenance

Example –

The development effort for a software project is 500 person months. The empirically determined constant (K) is 0.3. The complexity of the code is quite high and is equal to 8. Calculate the total effort expended (M) if

(i) maintenance team has good level of understanding of the project (d=0.9)

(ii) maintenance team has poor understanding of the project (d=0.1)

Software Maintenance

Solution

Development effort (P) = 500 PM

K = 0.3

C = 8

(i) maintenance team has good level of understanding of the project (d=0.9)

$$M = P + Ke^{(c-d)}$$

$$= 500 + 0.3e^{(8-0.9)}$$

$$= 500 + 363.59 = 863.59 \text{ PM}$$

(ii) maintenance team has poor understanding of the project (d=0.1)

$$M = P + Ke^{(c-d)}$$

$$= 500 + 0.3e^{(8-0.1)}$$

$$= 500 + 809.18 = 1309.18 \text{ PM}$$

Software Maintenance

- Boehm Model

Boehm used a quantity called Annual Change Traffic (ACT).

"The fraction of a software product's source instructions which undergo change during a year either through addition, deletion or modification".

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

(Annual Maintenance Effort) AME = ACT x SDE

Where, **SDE :** Software development effort in person months

ACT : Annual change Traffic

EAF : Effort Adjustment Factor

AME = ACT * SDE * EAF

Software Maintenance

Example – 2

Annual Change Traffic (ACT) for a software system is 15% per year. The development effort is 600 PMs. Compute estimate for Annual Maintenance Effort (AME). If life time of the project is 10 years, what is the total effort of the project ?

Software Maintenance

Solution

The development effort = 600 PM

Annual Change Traffic (ACT) = 15%

Total duration for which effort is to be calculated = 10 years

The maintenance effort is a fraction of development effort and is assumed to be constant.

AME = ACT x SDE

= 0.15 x 600 = 90 PM

Maintenance effort for 10 years = 10 x 90 = 900 PM

Total effort = 600 + 900 = 1500 PM

Software Maintenance

Example – 3

A software project has development effort of 500 PM. It is assumed that 10% code will be modified per year. Some of the cost multipliers are given as:

1. Required software Reliability (RELY) : high
2. Date base size (DATA) : high
3. Analyst capability (ACAP) : high
4. Application experience (AEXP) : Very high
5. Programming language experience (LEXP) : high

Other multipliers are nominal. Calculate the Annual Maintenance Effort (AME).

Solution

Annual change traffic (ACT) = 10%

Software development effort (SDE) = 500 Pm

Using Table from COCOMO model, effort adjustment factor can be calculated given below :

RELY = 1.15

ACAP = 0.86

AEXP = 0.82

LEXP = 0.95

DATA = 1.08

Software Maintenance

Other values are nominal values. Hence,

EAF = 1.15 x 0.86 x 0.82 x 0.95 x 1.08 = 0.832

AME = ACT * SDE * EAF

= 0.1 * 500 * 0.832 = 41.6 PM

AME = 41.6 PM

Reverse Engineering

Reverse engineering is the process followed in order to find difficult, unknown and hidden information about a software system.

Software Maintenance

- Scope and Tasks

The areas where reverse engineering is applicable include (but not limited to):

1. Program comprehension
2. Redocumentation and/ or document generation
3. Recovery of design approach and design details at any level of abstraction
4. Identifying reusable components
5. Identifying components that need restructuring
6. Recovering business rules, and
7. Understanding high level system description

Software Maintenance

- Levels of Reverse Engineering

Reverse Engineers detect low level implementation constructs and replace them with their high level counterparts.

The process eventually results in an incremental formation of an overall architecture of the program.

Software Maintenance

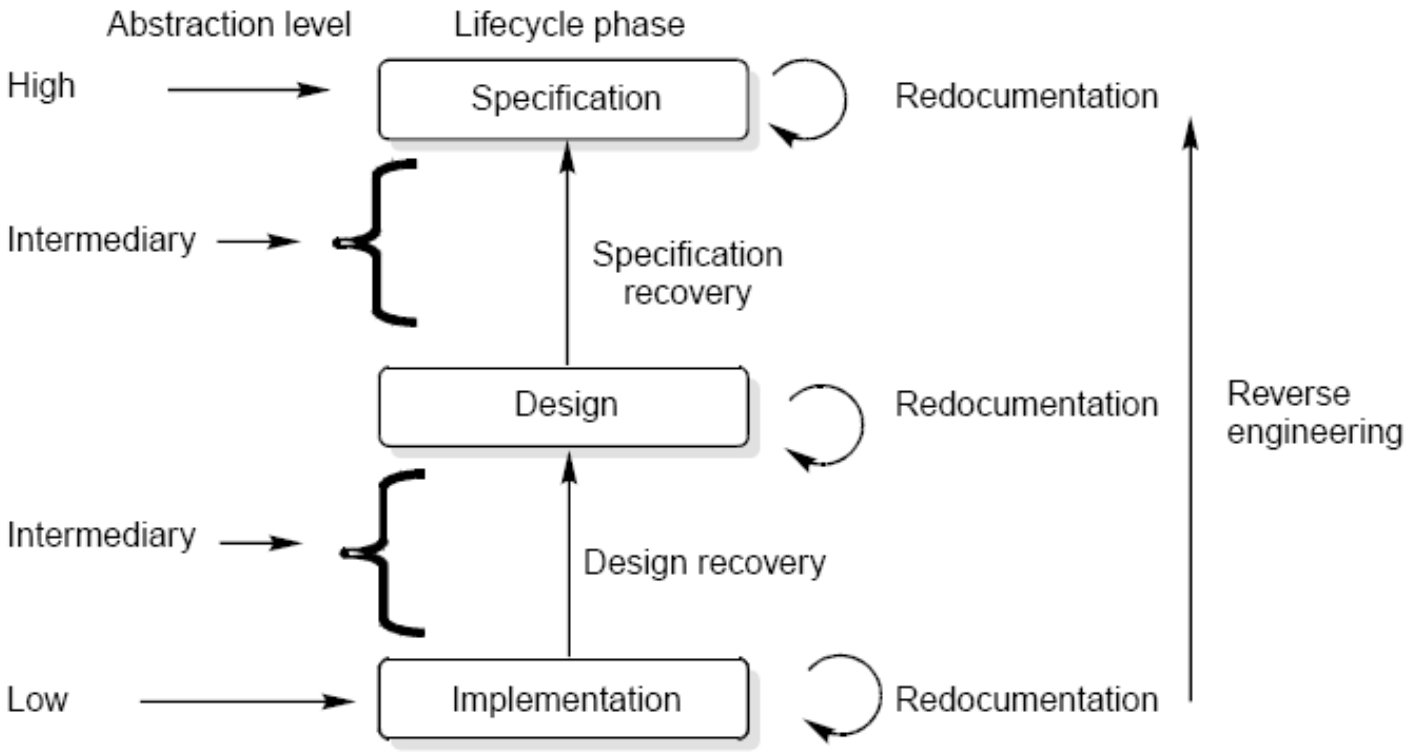

Fig. 11: Levels of abstraction

Software Maintenance

Redocumentation

Redocumentation is the recreation of a semantically equivalent representation within the same relative abstraction level.

Design recovery

Design recovery entails identifying and extracting meaningful higher level abstractions beyond those obtained directly from examination of the source code. This may be achieved from a combination of code, existing design documentation, personal experience, and knowledge of the problem and application domains.

Software RE-Engineering

Software re-engineering is concerned with taking existing legacy systems and re-implementing them to make them more maintainable.

The critical distinction between re-engineering and new software development is the starting point for the development as shown in Fig.12.

Software Maintenance

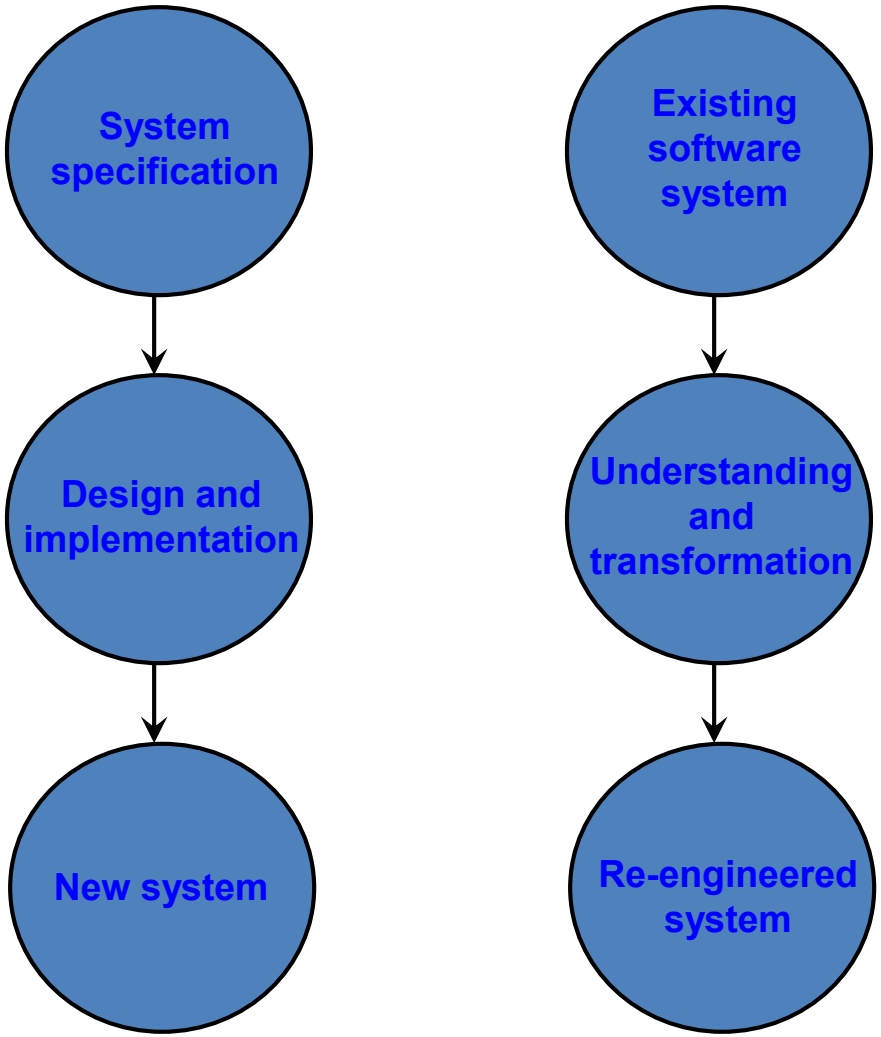

Fig. 12: Comparison of new software development with re-engineering

Software Maintenance

The following suggestions may be useful for the modification of the legacy code:

- ✓ Study code well before attempting changes
- ✓ Concentrate on overall control flow and not coding
- ✓ Heavily comment internal code
- ✓ Create Cross References
- ✓ Build Symbol tables
- ✓ Use own variables, constants and declarations to localize the effect
- ✓ Keep detailed maintenance document
- ✓ Use modern design techniques

Software Maintenance

- Source Code Translation

1. **Hardware platform update:** The organization may wish to change its standard hardware platform. Compilers for the original language may not be available on the new platform.

2. **Staff Skill Shortages:** There may be lack of trained maintenance staff for the original language. This is a particular problem where programs were written in some non standard language that has now gone out of general use.

3. **Organizational policy changes:** An organization may decide to standardize on a particular language to minimize its support software costs. Maintaining many versions of old compilers can be very expensive.

Configuration Management

The software development and maintenance is controlled by a process called as configuration management. The configuration management is different in development and maintenance phases of life cycle due to different environments.

- Configuration Management Activities

The activities are divided into four broad categories.

1. The identification of the components and changes
2. The control of the way by which the changes are made
3. Auditing the changes
4. Status accounting recording and documenting all the activities that have taken place

Software Maintenance

The following documents are required for these activities

- ✓ Project plan
- ✓ Software requirements specification document
- ✓ Software design description document
- ✓ Source code listing
- ✓ Test plans / procedures / test cases
- ✓ User manuals

Software Maintenance

- Software Versions

Two types of versions namely revisions (replace) and variations (variety).

Version Control :

A version control tool is the first stage towards being able to manage multiple versions. Once it is in place, a detailed record of every version of the software must be kept. This comprises the

- ✓ Name of each source code component, including the variations and revisions
- ✓ The versions of the various compilers and linkers used
- ✓ The name of the software staff who constructed the component
- ✓ The date and the time at which it was constructed

Software Maintenance

- Change Control Process

Change control process comes into effect when the software and associated documentation are delivered to configuration management change request form (as shown in fig. 14), which should record the recommendations regarding the change.

Fig. 14: Change request form

Change Request			
Project:			**Date:**
Change Requestor:			**Change No:**
Change Category (Check all that apply): ☐ Schedule ☐ Cost ☐ Scope ☐ Requirements/Deliverables ☐ Testing/Quality ☐ Resources			
Does this Change Affect (Check all that apply): ☐ Corrective Action ☐ Preventative Action ☐ Defect Repair ☐ Updates ☐ Other			
Describe the Change Being Requested:			
Describe the Reason for the Change:			
Describe all Alternatives Considered:			
Describe any Technical Changes Required to Implement this Change:			
Describe Risks to be Considered for this Change:			
Estimate Resources and Costs Needed to Implement this Change:			
Describe the Implications to Quality:			
Disposition: ☐ Approve ☐ Reject ☐ Defer			
Justification of Approval, Rejection, or Deferral:			

Change Board Approval:		
Name	**Signature**	**Date**

Software Maintenance

Documentation

Software documentation is the written record of the facts about a software system recorded with the intent to convey purpose, content and clarity.

Software Maintenance

- User Documentation

S.No.	Document	Function
1.	System Overview	Provides general description of system's functions.
2.	Installation Guide	Describes how to set up the system, customize it to local hardware needs and configure it to particular hardware and other software systems.
3.	Beginner's Guide	Provides simple explanations of how to start using the system.
4.	Reference Guide	Provides in depth description of each system facility and how it can be used.
5.	Enhancement	Booklet Contains a summary of new features.
6.	Quick reference card	Serves as a factual lookup.
7.	System administration	Provides information on services such as net-working, security and upgrading.

Table 5: User Documentation

Software Maintenance

- System Documentation

It refers to those documentation containing all facets of system, including analysis, specification, design, implementation, testing, security, error diagnosis and recovery.

Software Maintenance

- System Documentation

S.No.	Document	Function
1.	System Rationale	Describes the objectives of the entire system.
2.	SRS	Provides information on exact requirements of system as agreed between user and developers.
3.	Specification/ Design	Provides description of: (i) How system requirements are implemented. (ii) How the system is decomposed into a set of interacting program units. (iii) The function of each program unit.
4.	Implementation	Provides description of: (i) How the detailed system design is expressed in some formal programming language. (ii) Program actions in the form of intra program comments.

Software Maintenance

S.No.	Document	Function
5.	System Test Plan	Provides description of how program units are tested individually and how the whole system is tested after integration.
6.	Acceptance Test Plan	Describes the tests that the system must pass before users accept it.
7.	Data Dictionaries	Contains description of all terms that relate to the software system in question.

Table 6: System Documentation