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

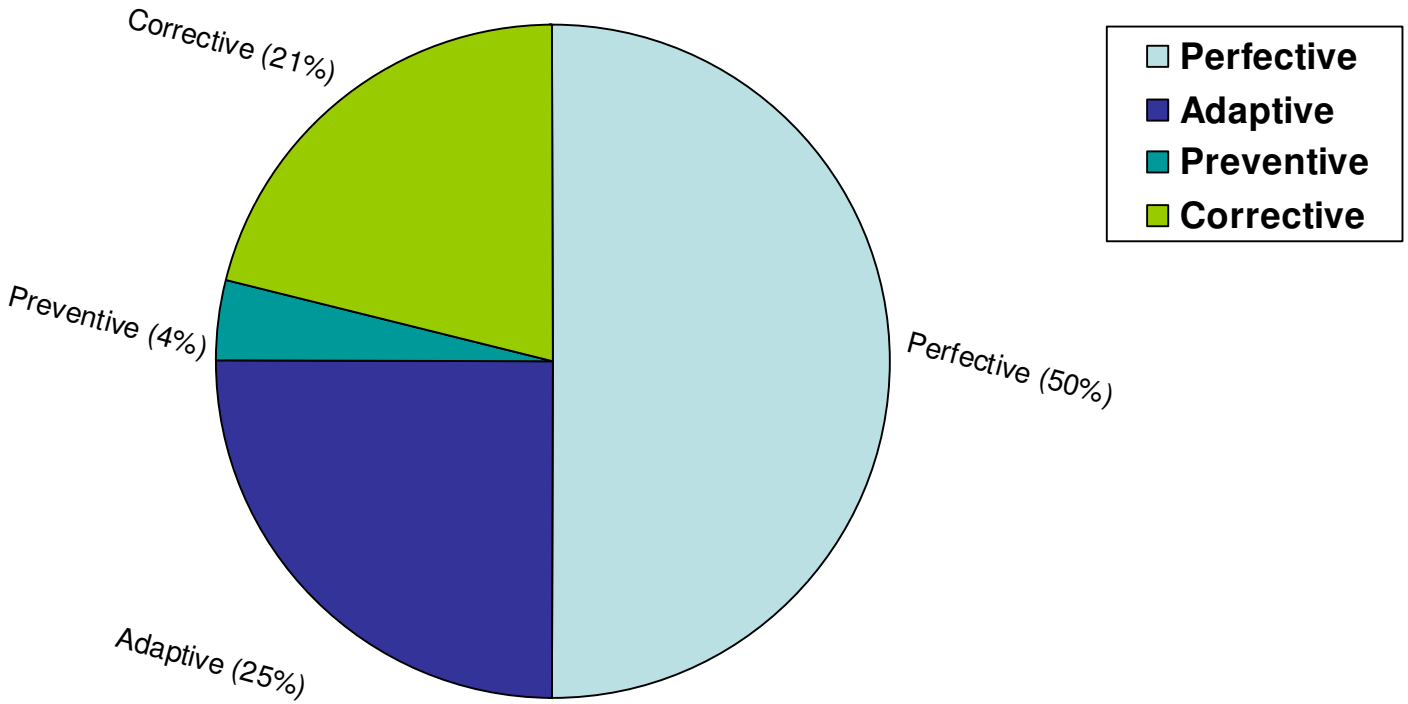

Fig. 1: Distribution of maintenance effort

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.
- High staff turnover.
- 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

Kinds of maintenance requests

1	New reports	40.8%
2	Add data in existing reports	27.1%
3	Reformed reports	10%
4	Condense reports	5.6%
5	Consolidate reports	6.4%
6	Others	10.1%

Table 2: Kinds of maintenance requests

Software Maintenance

Potential Solutions to Maintenance Problems

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

Software Maintenance

The Maintenance Process

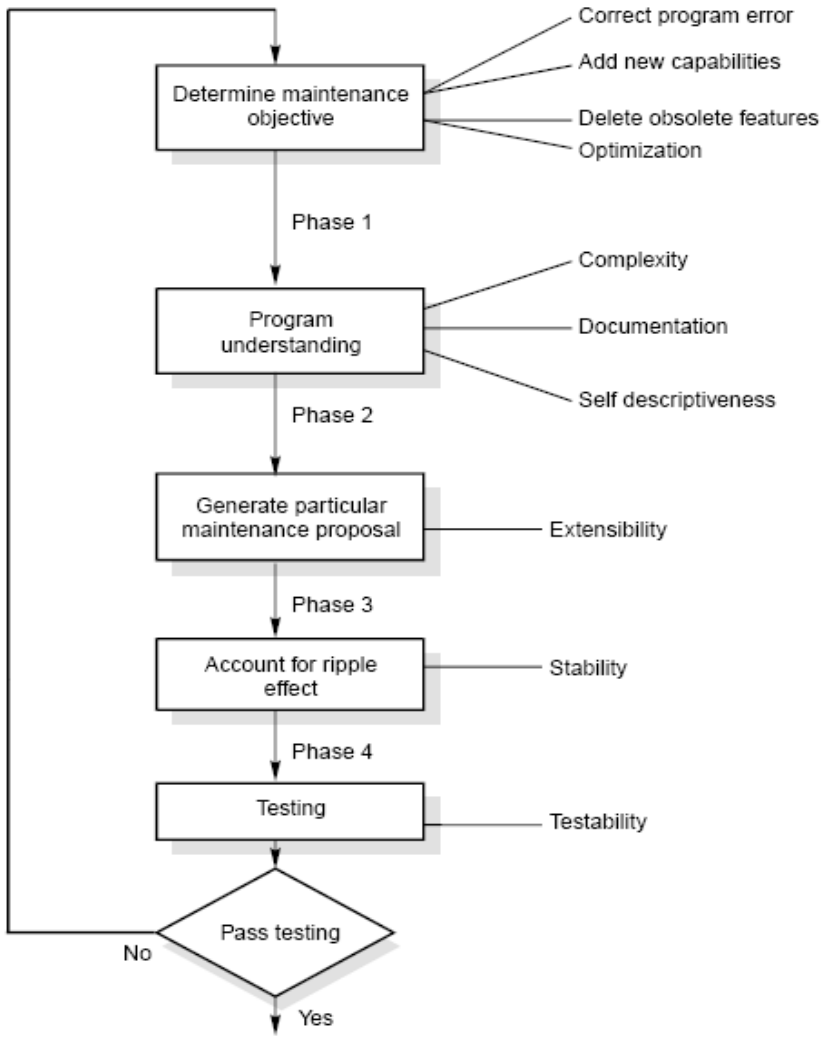

Fig. 2: The software maintenance process

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.

Software Maintenance

Maintenance Models

- Quick-fix Model

This is basically an adhoc approach to maintaining software. It is a fire fighting approach, waiting for the problem to occur and then trying to fix it as quickly as possible.

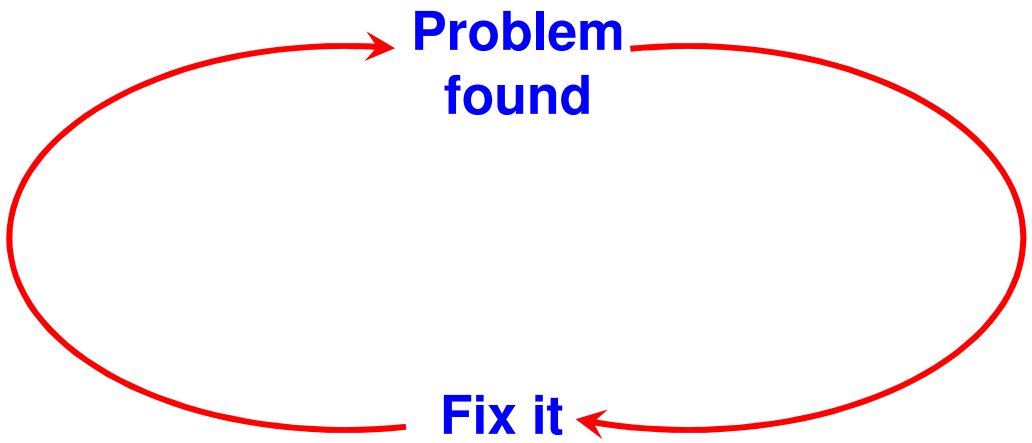

Fig. 3: The quick-fix model

Software Maintenance

- Iterative Enhancement Model
  - Analysis
  - Characterization of proposed modifications
  - Redesign and implementation

Software Maintenance

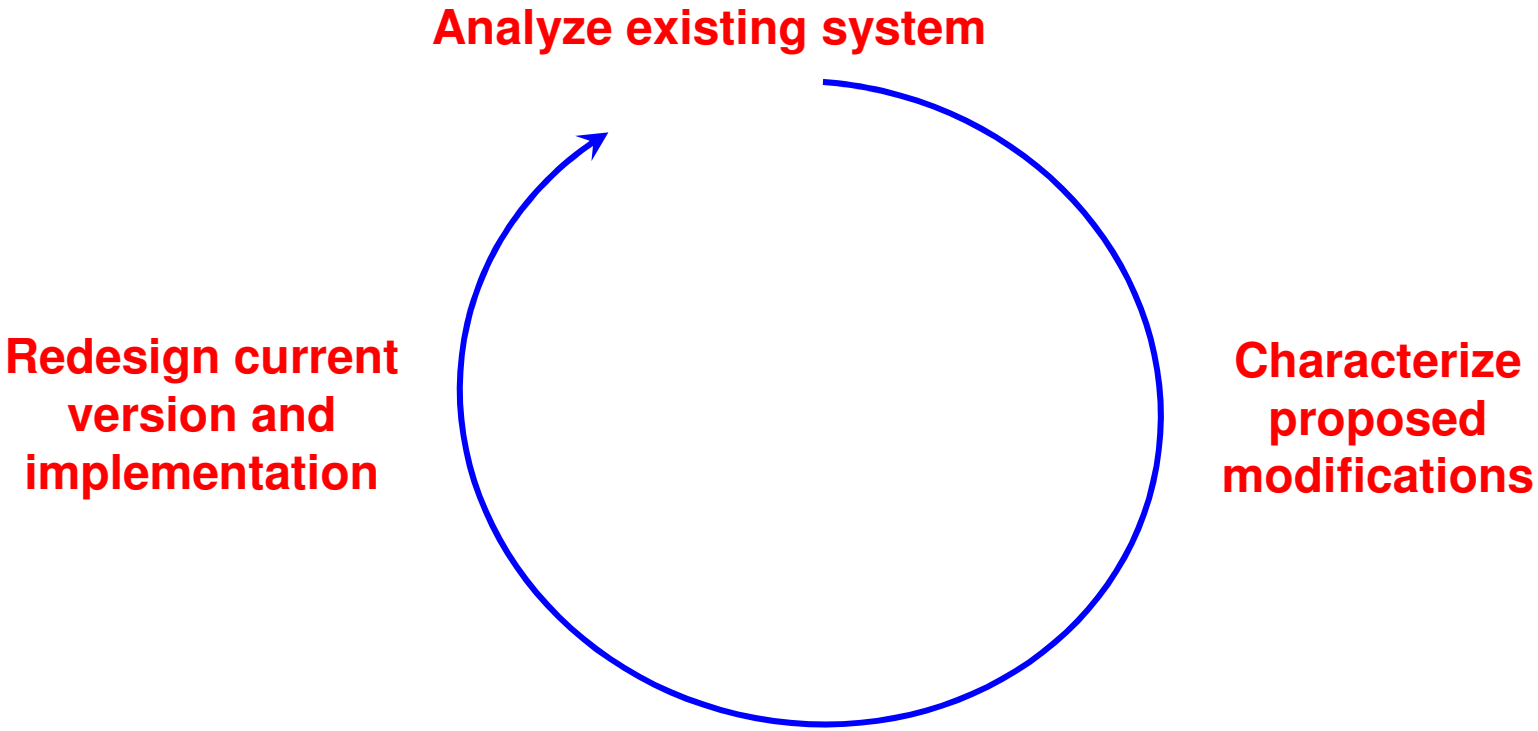

Fig. 4: The three stage cycle of iterative enhancement

Software Maintenance

- Reuse Oriented Model

The reuse model has four main steps:

1. Identification of the parts of the old system that are candidates for reuse.
2. Understanding these system parts.
3. Modification of the old system parts appropriate to the new requirements.
4. Integration of the modified parts into the new system.

Software Maintenance

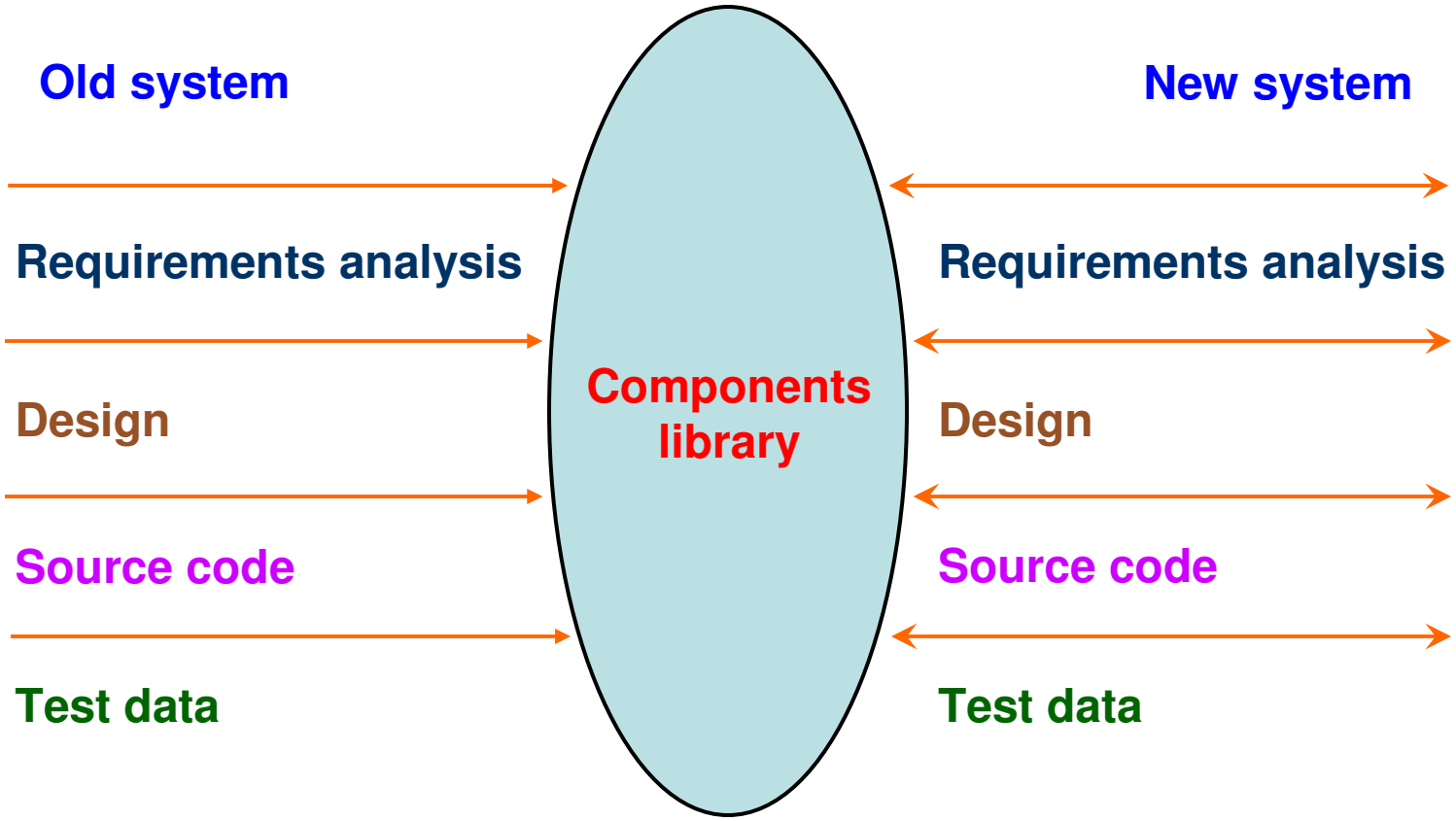

Fig. 5: The reuse model

Software Maintenance

- Boehm's Model

Boehm proposed a model for the maintenance process based upon the economic models and principles.

Boehm represent the maintenance process as a closed loop cycle.

Software Maintenance

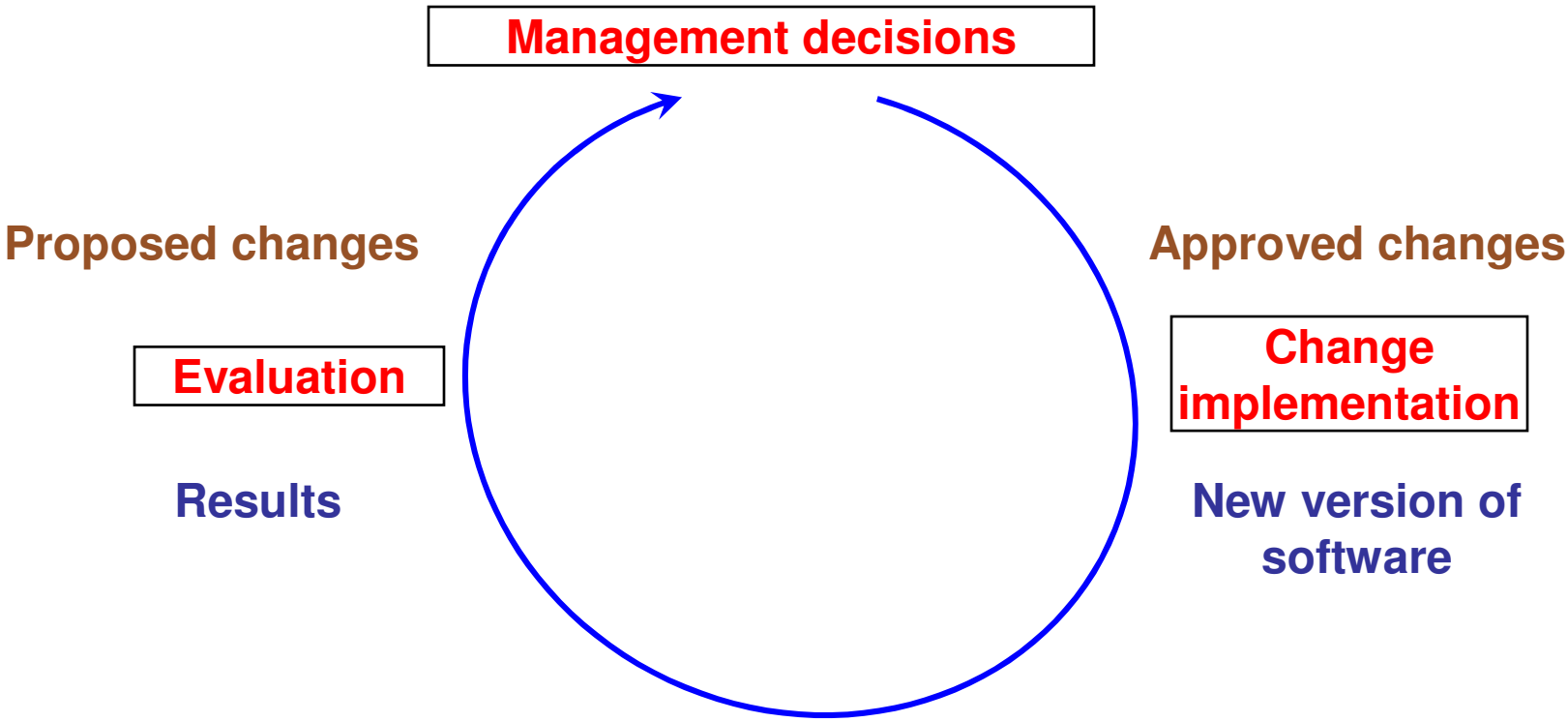

Fig. 6: Boehm's model

Software Maintenance

- ## Taute Maintenance Model

It is a typical maintenance model and has eight phases in cycle fashion. The phases are shown in Fig. 7

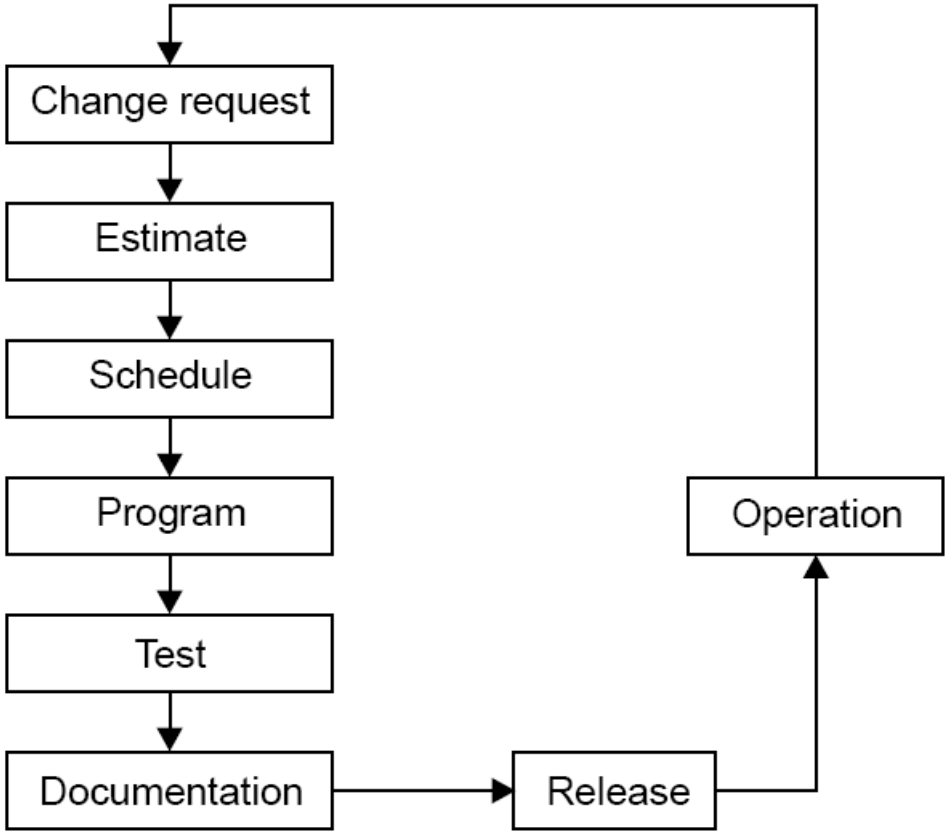

Fig. 7: Taute maintenance model

Software Maintenance

Phases :

1. Change request phase
2. Estimate phase
3. Schedule phase
4. Programming phase
5. Test phase
6. Documentation phase
7. Release phase
8. Operation phase

Software Maintenance

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 – 9.1

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".

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

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 – 9.2

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.

$$AME = ACT \times SDE$$

$$= 0.15 \times 600 = 90 \text{ PM}$$

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

Total effort = 600 + 900 = 1500 PM

Software Maintenance

Example – 9.3

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).

Software Maintenance

Solution

Annual change traffic (ACT) = 10%

Software development effort (SDE) = 500 Pm

Using Table 5 of 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

Regression Testing

Regression testing is the process of retesting the modified parts of the software and ensuring that no new errors have been introduced into previously test code.

"Regression testing tests both the modified code and other parts of the program that may be affected by the program change. It serves many purposes :

- increase confidence in the correctness of the modified program
- locate errors in the modified program
- preserve the quality and reliability of software
- ensure the software's continued operation

Software Maintenance

- Development Testing Versus Regression Testing

Sr. No.	Development testing	Regression testing
1.	We create test suites and test plans	We can make use of existing test suite and test plans
2.	We test all software components	We retest affected components that have been modified by modifications.
3.	Budget gives time for testing	Budget often does not give time for regression testing.
4.	We perform testing just once on a software product	We perform regression testing many times over the life of the software product.
5.	Performed under the pressure of release date of the software	Performed in crisis situations, under greater time constraints.

Software Maintenance

- Regression Test Selection

Regression testing is very expensive activity and consumes significant amount of effort / cost. Many techniques are available to reduce this effort/ cost.

1. Reuse the whole test suite

2. Reuse the existing test suite, but to apply a regression test selection technique to select an appropriate subset of the test suite to be run.

Software Maintenance

Fragment A		Fragment B (modified form of A)	
S_1	y = (x - 1) * (x + 1)	S_1'	y = (x -1) * (x + 1)
S_2	if (y = 0)	S_2'	if (y = 0)
S_3	return (error)	S_3'	return (error)
S_4	else	S_4'	else
S_5	return $\left(\frac{1}{y}\right)$	S_5'	return $\left(\frac{1}{y-3}\right)$

Fig. 8: code fragment A and B

Software Maintenance

Test cases		
Test number	Input	Execution History
t_1	x = 1	S_1, S_2, S_3
t_2	x = -1	S_1, S_2, S_3
t_3	x = 2	S_1, S_2, S_5
t_4	x = 0	S_1, S_2, S_5

Fig. 9: Test cases for code fragment A of Fig. 8

Software Maintenance

If we execute all test cases, we will detect this divide by zero fault. But we have to minimize the test suite. From the fig. 9, it is clear that test cases t_3 and t_4 have the same execution history i.e. S_1, S_2, S_5. If few test cases have the same execution history; minimization methods select only one test case. Hence, either t_3 or t_4 will be selected. If we select t_4 then fine otherwise fault not found.

Minimization methods can omit some test cases that might expose fault in the modified software and so, they are not safe.

A safe regression test selection technique is one that, under certain assumptions, selects every test case from the original test suite that can expose faults in the modified program.

Software Maintenance

Selective Retest Techniques

Selective retest techniques may be more economical than the “retest-all” technique.

Selective retest techniques are broadly classified in three categories :

1. **Coverage techniques :** They are based on test coverage criteria. They locate coverable program components that have been modified, and select test cases that exercise these components.

2. **Minimization techniques:** They work like coverage techniques, except that they select minimal sets of test cases.

3. **Safe techniques:** They do not focus on coverage criteria; instead they select every test case that cause a modified program to produce different output than its original version.

Software Maintenance

Rothermal identified categories in which regression test selection techniques can be compared and evaluated. These categories are:

Inclusiveness measures the extent to which a technique chooses test cases that will cause the modified program to produce different output than the original program, and thereby expose faults caused by modifications.

Precision measures the ability of a technique to avoid choosing test cases that will not cause the modified program to produce different output than the original program.

Efficiency measures the computational cost, and thus, practically, of a technique.

Generality measures the ability of a technique to handle realistic and diverse language constructs, arbitrarily complex modifications, and realistic testing applications.

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 there 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

Reverse Engineering encompasses a wide array of tasks related to understanding and modifying software system. This array of tasks can be broken into a number of classes.

- Mapping between application and program domains

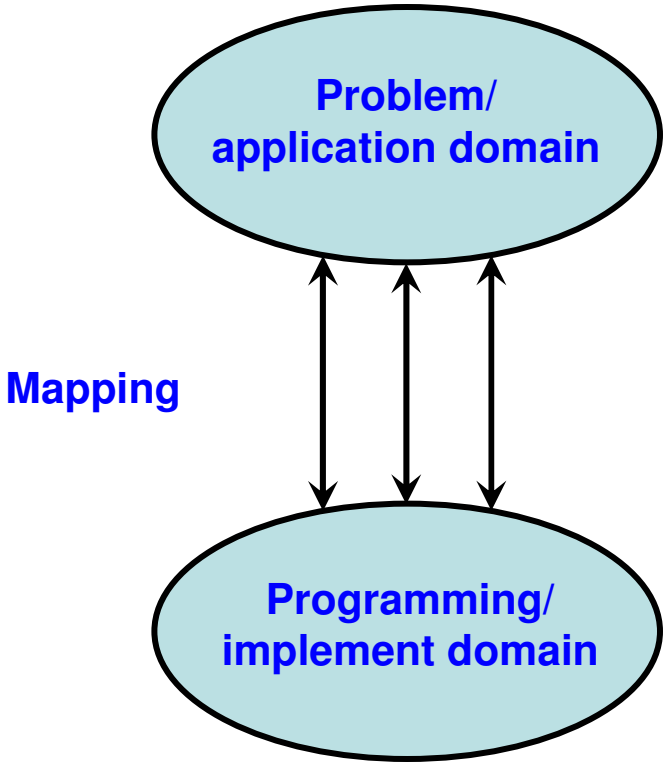

Fig. 10: Mapping between application and domains program

Software Maintenance

- Mapping between concrete and abstract levels
- Rediscovering high level structures
- Finding missing links between program syntax and semantics
- To extract reusable component

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

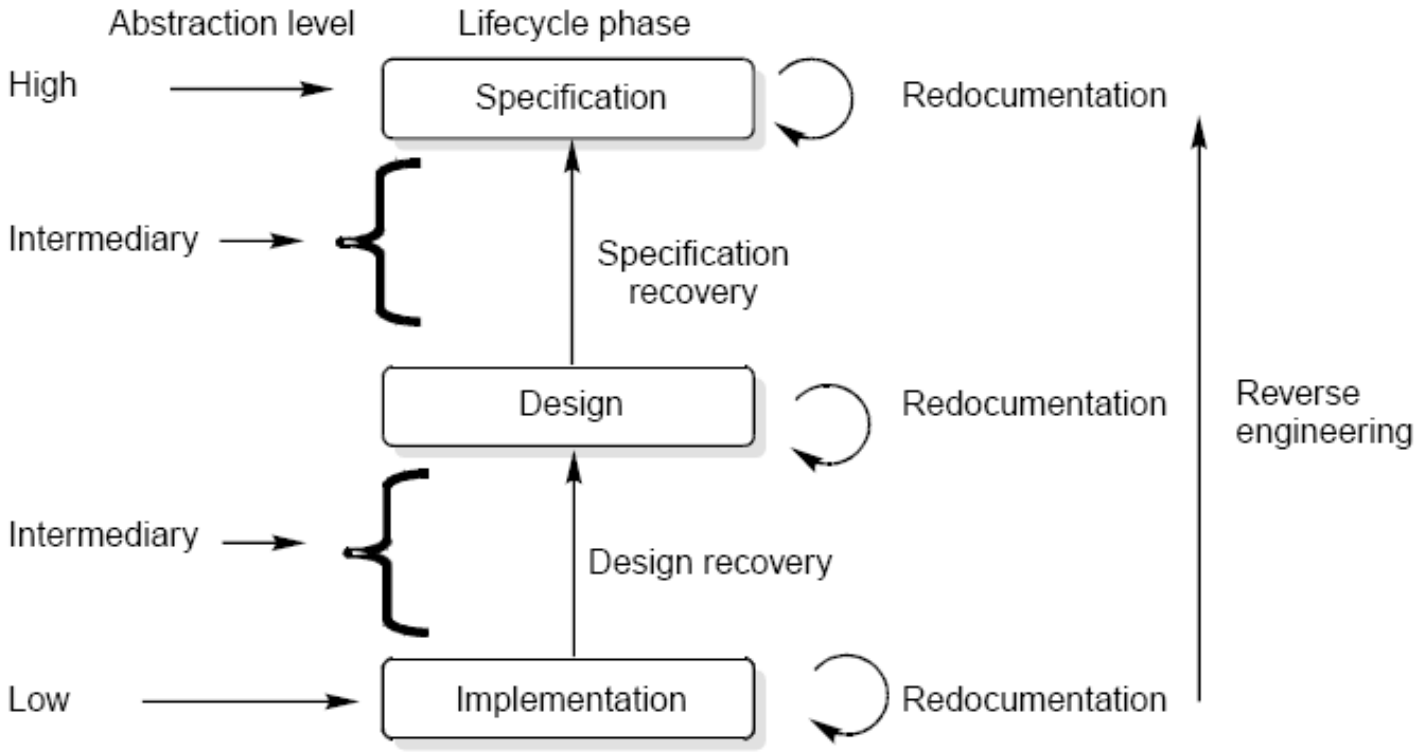

Fig. 11: Levels of abstraction

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 Maintenance

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

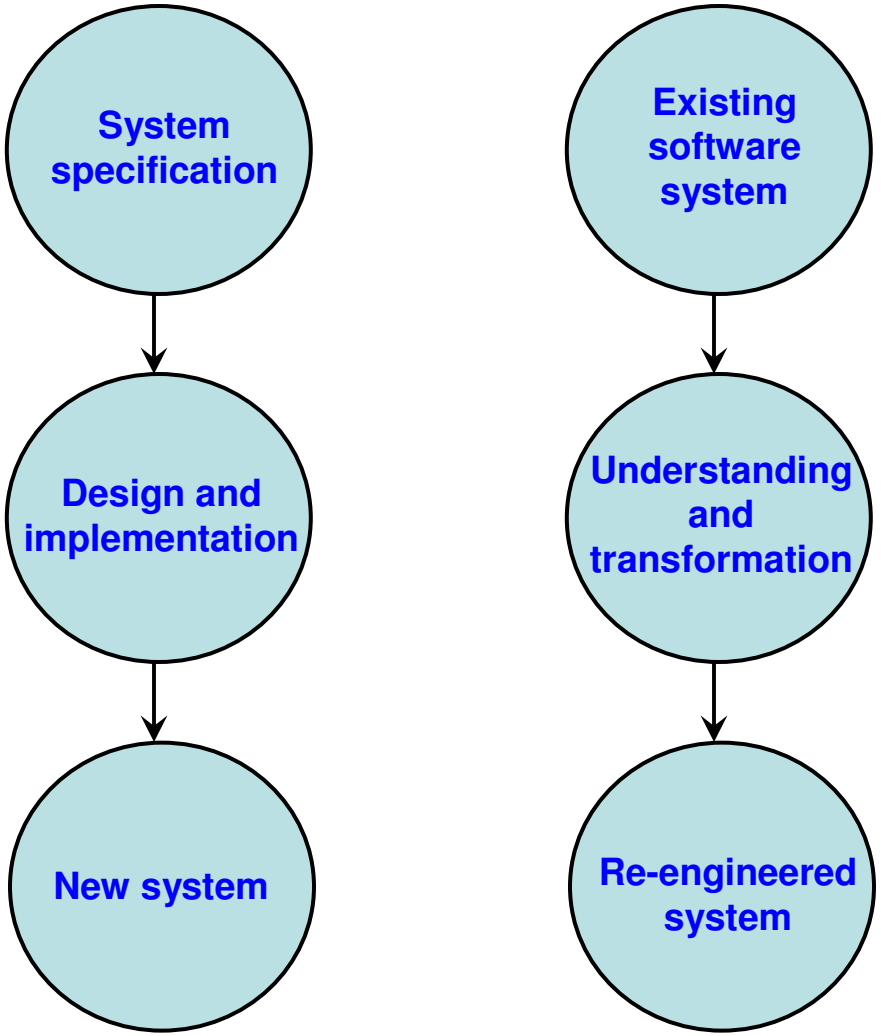

Fig. 12: Comparison of new software development with re-engineering

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.

Software Maintenance

- Program Restructuring

1. **Control flow driven restructuring:** This involves the imposition of a clear control structure within the source code and can be either inter modular or intra modular in nature.

2. **Efficiency driven restructuring:** This involves restructuring a function or algorithm to make it more efficient. A simple example is the replacement of an IF-THEN-ELSE-IF-ELSE construct with a CASE construct.

Software Maintenance

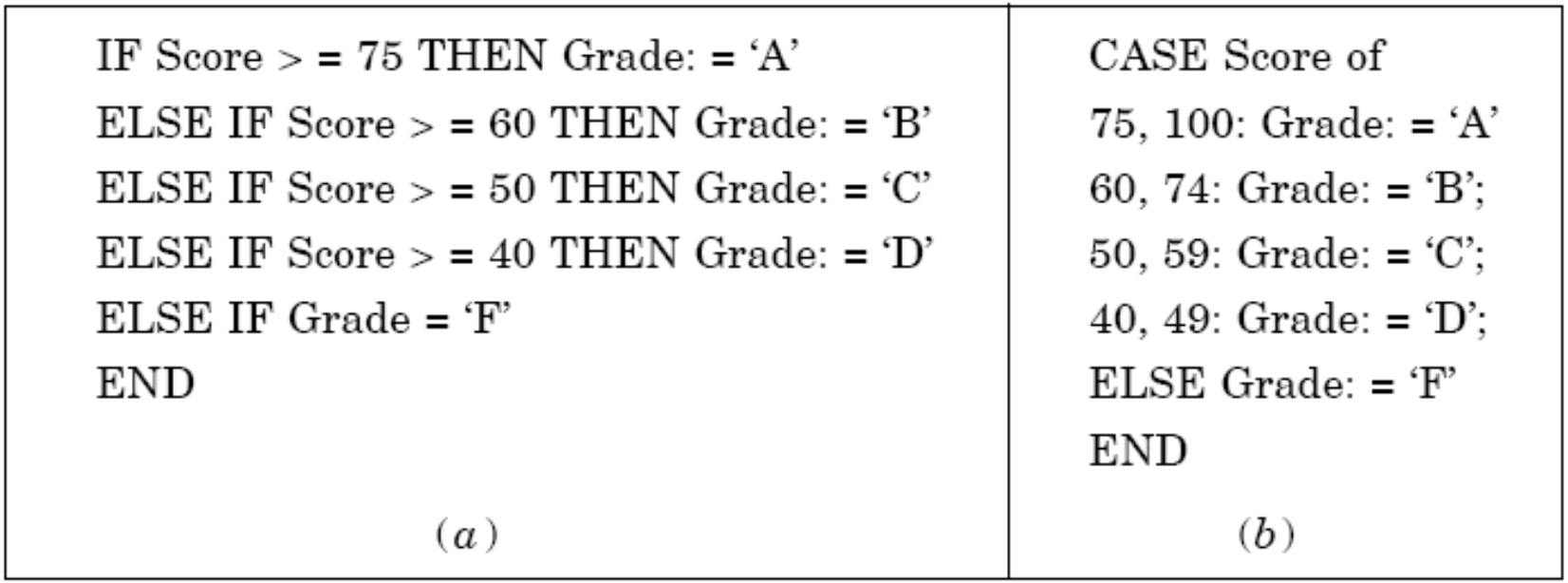

Fig. 13: Restructuring a program

Software Maintenance

3. **Adaption driven restructuring:** This involves changing the coding style in order to adapt the program to a new programming language or new operating environment, for instance changing an imperative program in PASCAL into a functional program in LISP.

Configuration Management

The process of software development and maintenance is controlled is called 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 take 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.

CHANGE REQUEST FORM

Project ID:

Change Requester with date:

Requested change with date:

Change analyzer:

Components affected:

Associated components:

Estimated change costs:

Change priority:

Change assessment:

Change implementation:

Date submitted to CCA:

Date of CCA decision:

CCA decision:

Change implementer:

Date submitted to QA:

Date of implementation:

Date submitted to CM:

QA decision:

Fig. 14: Change request form

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

Multiple Choice Questions

Note: Choose most appropriate answer of the following questions:

9.1 Process of generating analysis and design documents is called

(a) Inverse Engineering
(b) Software Engineering
(c) Reverse Engineering
(d) Re-engineering

9.2 Regression testing is primarily related to

(a) Functional testing
(b) Data flow testing
(c) Development testing
(d) Maintenance testing

9.3 Which one is not a category of maintenance ?

(a) Corrective maintenance
(b) Effective maintenance
(c) Adaptive maintenance
(d) Perfective maintenance

9.4 The maintenance initiated by defects in the software is called

(a) Corrective maintenance
(b) Adaptive maintenance
(c) Perfective maintenance
(d) Preventive maintenance

9.5 Patch is known as

(a) Emergency fixes
(b) Routine fixes
(c) Critical fixes
(d) None of the above

Multiple Choice Questions

9.6 Adaptive maintenance is related to

(a) Modification in software due to failure

(b) Modification in software due to demand of new functionalities

(c) Modification in software due to increase in complexity

(d) Modification in software to match changes in the ever-changing environment.

9.7 Perfective maintenance refers to enhancements

(a) Making the product better

(b) Making the product faster and smaller

(c) Making the product with new functionalities

(d) All of the above

9.8 As per distribution of maintenance effort, which type of maintenance has consumed maximum share?

(a) Adaptive (b) Corrective

(c) Perfective (d) Preventive

9.9 As per distribution of maintenance effort, which type of maintenance has consumed minimum share?

(a) Adaptive (b) Corrective

(c) Perfective (d) Preventive

Multiple Choice Questions

9.10 Which one is not a maintenance model ?

(a) CMM
(b) Iterative Enhancement model
(c) Quick-fix model
(d) Reuse-Oriented model

9.11 In which model, fixes are done without detailed analysis of the long-term effects?

(a) Reuse oriented model
(b) Quick-fix model
(c) Taute maintenance model
(d) None of the above

9.12 Iterative enhancement model is a

(a) three stage model
(b) two stage model
(c) four stage model
(d) seven stage model

9.13 Taute maintenance model has

(a) Two phases
(b) six phases
(c) eight phases
(d) ten phases

9.14 In Boehm model, ACT stands for

(a) Actual change time
(b) Actual change traffic
(c) Annual change traffic
(d) Annual change time

Multiple Choice Questions

9.15 Regression testing is known as

(a) the process of retesting the modified parts of the software

(b) the process of testing the design documents

(c) the process of reviewing the SRS

(d) None of the above

9.16 The purpose of regression testing is to

(a) increase confidence in the correctness of the modified program

(b) locate errors in the modified program

(c) preserve the quantity and reliability of software

(d) All of the above

9.17 Regression testing is related to

(a) maintenance of software

(b) development of software

(c) both (a) and (b)

(d) none of the above.

9.18 Which one is not a selective retest technique

(a) coverage technique

(b) minimization technique

(c) safe technique

(d) maximization technique

Multiple Choice Questions

9.19 Purpose of reverse engineering is to

(a) recover information from the existing code or any other intermediate document

(b) redocumentation and/or document generation

(c) understand the source code and associated documents

(d) All of the above

9.20 Legacy systems are

(a) old systems

(b) new systems

(c) undeveloped systems

(d) None of the above

9.21 User documentation consists of

(a) System overview

(b) Installation guide

(c) Reference guide

(d) All of the above

9.22 Which one is not a user documentations ?

(a) Beginner's Guide

(b) Installation guide

(c) SRS

(d) System administration

Multiple Choice Questions

9.23 System documentation may not have

(a) SRS
(b) Design document
(c) Acceptance Test Plan
(d) System administration

9.24 The process by which existing processes and methods are replaced by new techniques is:

(a) Reverse engineering
(b) Business process re-engineering
(c) Software configuration management
(d) Technical feasibility

9.25 The process of transforming a model into source code is

(a) Reverse Engineering
(b) Forward engineering
(c) Re-engineering
(d) Restructuring

Exercises

9.1 What is software maintenance? Describe various categories of maintenance. Which category consumes maximum effort and why?

9.2 What are the implication of maintenance for a one person software production organisation?

9.3 Some people feel that "maintenance is manageable". What is your opinion about this issue?

9.4 Discuss various problems during maintenance. Describe some solutions to these problems.

9.5 Why do you think that the mistake is frequently made of considering software maintenance inferior to software development?

9.6 Explain the importance of maintenance. Which category consumes maximum effort and why?

9.7 Explain the steps of software maintenance with help of a diagram.

9.8 What is self descriptiveness of a program? Explain the effect of this parameter on maintenance activities.

Exercises

9.9 What is ripple effect? Discuss the various aspects of ripple effect and how does it affect the stability of a program?

9.10 What is maintainability? What is its role during maintenance?

9.11 Describe Quick-fix model. What are the advantage and disadvantage of this model?

9.12 How iterative enhancement model is helpful during maintenance? Explain the various stage cycles of this model.

9.13 Explain the Boehm's maintenance model with the help of a diagram.

9.14 State the various steps of reuse oriented model. Is it a recommended model in object oriented design?

9.15 Describe the Taute maintenance model. What are various phases of this model?

9.16 Write a short note on Boledy and Lehman model for the calculation of maintenance effort.

Exercises

9.17 Describe various maintenance cost estimation model.s

9.18 The development effort for a project is 600 PMs. The empirically determined constant (K) of Belady and Lehman model is 0.5. The complexity of code is quite high and is equal to 7. Calculate the total effort expended (M) if maintenance team has reasonable level of understanding of the project (d=0.7).

9.19 Annual change traffic (ACT) in a software system is 25% per year. The initial development cost was Rs. 20 lacs. Total life time for software is 10 years. What is the total cost of the software system?

9.20 What is regression testing? Differentiate between regression and development testing?

9.21 What is the importance of regression test selection? Discuss with help of examples.

9.22 What are selective retest techniques? How are they different from "retest-all" techniques?

Exercises

9.23 Explain the various categories of retest techniques. Which one is not useful and why?

9.24 What are the categories to evaluate regression test selection techniques? Why do we use such categorisation?

9.25 What is reverse engineering? Discuss levels of reverse engineering.

9.26 What are the appropriate reverse engineering tools? Discuss any two tools in detail.

9.27 Discuss reverse engineering and re-engineering.

9.28 What is re-engineering? Differentiate between re-engineering and new development.

9.29 Discuss the suggestions that may be useful for the modification of the legacy code.

9.30 Explain various types of restructuring techniques. How does restructuring help in maintaining a program?

Exercises

9.31 Explain why single entry, single exit modules make testing easier during maintenance.

9.32 What are configuration management activities? Draw the performa of change request form.

9.33 Explain why the success of a system depends heavily on the quantity of the documentation generated during system development.

9.34 What is an appropriate set of tools and documents required to maintain large software product/

9.35 Explain why a high degree of coupling among modules can make maintenance very difficult.

9.36 Is it feasible to specify maintainability in the SRS? If yes, how would we specify it?

9.37 What tools and techniques are available for software maintenance? Discuss any two of them.

Exercises

9.38 Why is maintenance programming becoming more challenging than new development? What are desirable characteristics of a maintenance programmer?

9.39 Why little attention is paid to maintainability during design phase?

9.40 List out system documentation and also explain their purpose.