Software Maintenance

Course Contents

1. What is Software Maintenance?
2. Why is Software Maintenance Necessary
3. Types of Maintenance
4. The Importance of Categorizing Software Changes
5. A Comparison between Development and Maintenance
6. The Cost of Maintenance
7. Software Maintenance Framework
8. Potential Solution to Maintenance Problem

Introduction

Software maintenance is often considered to be (if it is considered at all) an unpleasant, time consuming, expensive and unrewarding occupation - something that is carried out at the end of development only when absolutely necessary (and hopefully not very often).

Part of the confusion about software maintenance

- relates to its definition;
- when it begins,
- when it ends and
- how it relates to development.

Therefore it is necessary to first consider what is meant by the term software maintenance.

What is Software Maintenance?

- In order to define software maintenance we need to define exactly what is meant by software. It is a common misreading to believe that software is programs and that maintenance activities are carried out exclusively on programs. This is because many software maintainers are more familiar with, or rather are more exposed to programs than other components of a software system – usually it is the program code that attracts most attention

Definition of Software

- A more comprehensive view of software is given by McDermid (1991) who states that it "consists of the programs, documentation and operating procedures by which computers can be made useful to man".

Table shows the components of a software system according to this view and includes some examples of each.

Software Components	**Examples**	
Program	Source code	
	Object code	
Documentation	Analysis/specification:	(a) Formal specification
		(b) Context diagram
		(c) Data flow diagrams
	Design:	(a) Flowcharts
		(b) Entity-relationship charts
	Implementation:	(a) Source code listings
		(b) Cross-reference listing
	Testing:	(a) Test data
		(b) Test results
Operating Procedures	1. Instructions to set up and use the software system	
	2. Instructions on how to react to system failures	

Definition of Maintenance

- To apply the traditional definition of maintenance in the context of software means that software maintenance is only concerned with correcting errors. However, correcting errors accounts for only part of the maintenance effort.

Definitions of Software Maintenance

Some definitions focus on the particular activities carried out during maintenance, for example, According to Martin and McClure (1983), software maintenance must be performed in order to

- Correct errors
- Correct design imperfections
- Interface with other systems
- Make enhancements
- Make necessary changes to the system
- Make changes in files or databases
- Improve the design
- Convert programs so that different hardware, software, system features, and telecommunications facilities can be used

- Others insist on a general view which considers software maintenance as “any work that is undertaken after delivery of a software system”.
- Changes that have to be made to computer programs after they have been delivered to the customer or user." (Martin and McClure 1983).
- "Maintenance covers the life of a software system from the time it is installed until it is phased out.“ (von Mayrhauser 1990).

- The common theme of the above definitions is that maintenance is an "after-the-fact" activity. Based on these definitions, no maintenance activities occur during the software development effort (the pre-delivery stage of the life cycle of a software system).
- Maintenance occurs after the product is in operation (during the post-delivery stage).

- During development little consideration is made to the maintenance phase which is to follow the developers considering their work done once the system is handed over to users.

Other perspectives include:

- The 'bug-fixing' view which considers software maintenance as an activity involving the detection and correction of errors found in programs,
- The 'need-to-adapt' view which sees maintenance as a activity which entails changing the software when its operational environment or original requirement changes.
- The 'support' to users view where maintenance of software is seen as providing a service to users of the system.

IEEE S/W Maintenance Standard

- The IEEE software maintenance standard, IEEE STD 1219-1993, which draws on these different views, defines software maintenance as:
- ***Modification of a software product after delivery, to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment***

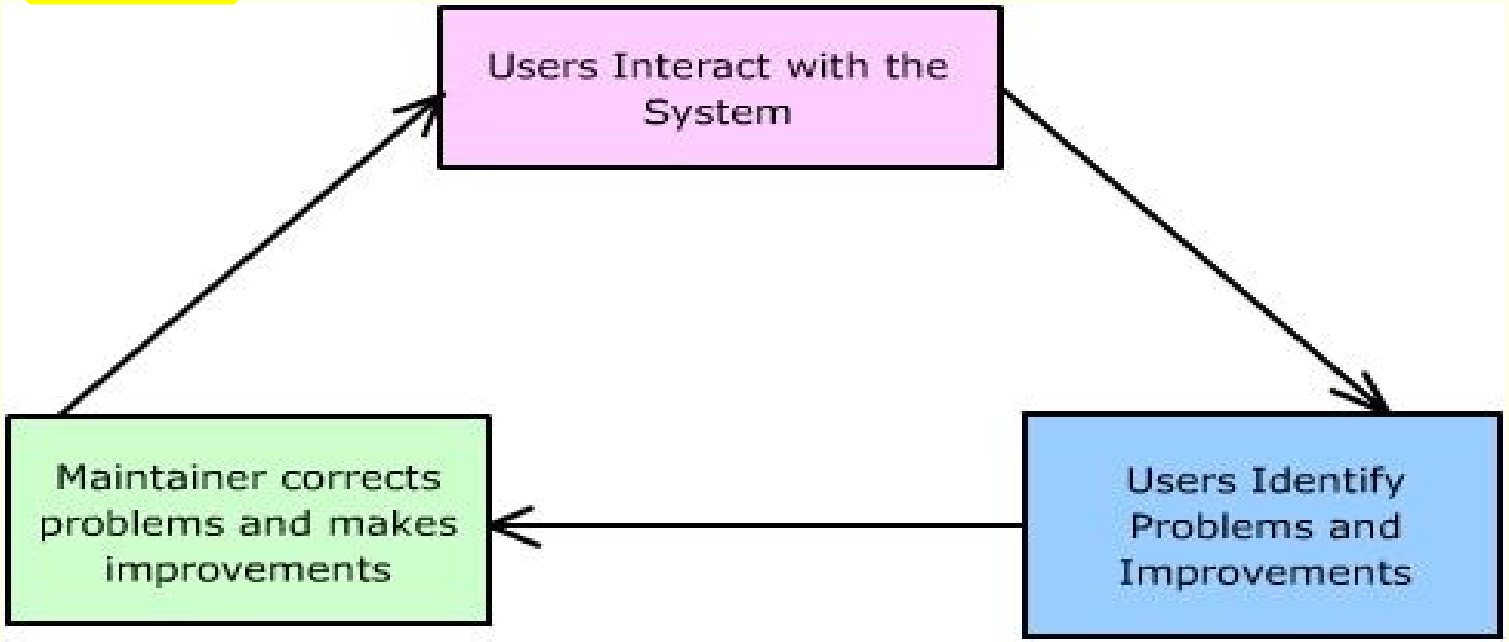
- *Maintenance is the **activity** associated with keeping **operational** computer systems continuously in tune with **requirements of users** & **data processing operation**.*
- ***Software Maintenance is any work done to a computer program after its first installation and implementation in an operational environment***

The Maintenance of s/w system is motivated by a number of factors:

- To provide continuity of service: This entails fixing bugs, recovering from failures, and accommodating changes in the operating system and hardware,
- To support compulsory upgrades: These are usually caused by changes in government regulations, and also by attempts to maintain a competitive edge over rival products.
- To support user requests for improvements: Examples include enhancement of functionality, better performance and customization to local working patterns.
- To facilitate future maintenance work: This usually involves code and database restructuring, and updating documentation.

Why is Software Maintenance necessary?

- The lifecycle of maintenance on a software product and why (theoretically) it may be never ending.



- Lehman's (1980) first two laws of software evolution help explain why the Operations and Maintenance phase can be the longest of the life-cycle processes.
- His first law is the Law of Continuing Change, which states that a system needs to change in order to be useful.
- The second law is the Law of Increasing Complexity, which states that the structure of a program proves as it evolves. Over time, the structure of the code degrades until it becomes more cost-effective to rewrite the program.

Types of Software Maintenance

- Corrective
- Adaptive
- Perfective
- Preventive

Corrective Maintenance

- Changes required by actual errors (defects or residual "bugs") in a system are termed corrective maintenance. These defects visible themselves when the system does not operate as it was designed or advertised to do.
- Corrective maintenance has been estimated to account for 20% of all maintenance activities.

- A defect or "bug" can result from design errors, logic errors and coding errors.
- Design errors occur when for example changes made to the software are incorrect, incomplete, wrongly communicated or the change request misunderstood.
- Logic errors result from invalid tests and conclusions, incorrect implementation of design specification, faulty logic flow or incomplete test data.

- Coding errors are caused by incorrect implementation of detailed logic design and incorrect use of the source code logic.
- Defects are also caused by data processing errors and system performance errors. All these errors, sometimes called “bugs” prevent the software from conforming to its agreed specification.

Adaptive Maintenance

- Any effort that is initiated as a result of changes in the environment in which a software system must operate is termed adaptive change.
- Adaptive change is a change driven by the need to accommodate modifications in the environment of the software system, without which the system would become increasingly less useful until it became obsolete.

- This type of maintenance the user does not see a direct change in the operation of the system, but the software maintainer must expend resources to effect the change.
- This task is estimated to consume about 25% of the total maintenance activity.

Perfective Maintenance

- The third widely accepted task is that of perfective maintenance.
- This is actually the most common type of maintenance surrounding enhancements both to the function and the efficiency of the code and includes all changes, insertions, deletions, modifications, extensions, and enhancements made to a system to meet the evolving and/or expanding needs of the user.
- A successful piece of software tends to be subjected to a succession of changes resulting in an increase in its requirements.

- Perfective maintenance is by far the largest consumer of maintenance resources, estimates of around 50% are not uncommon.

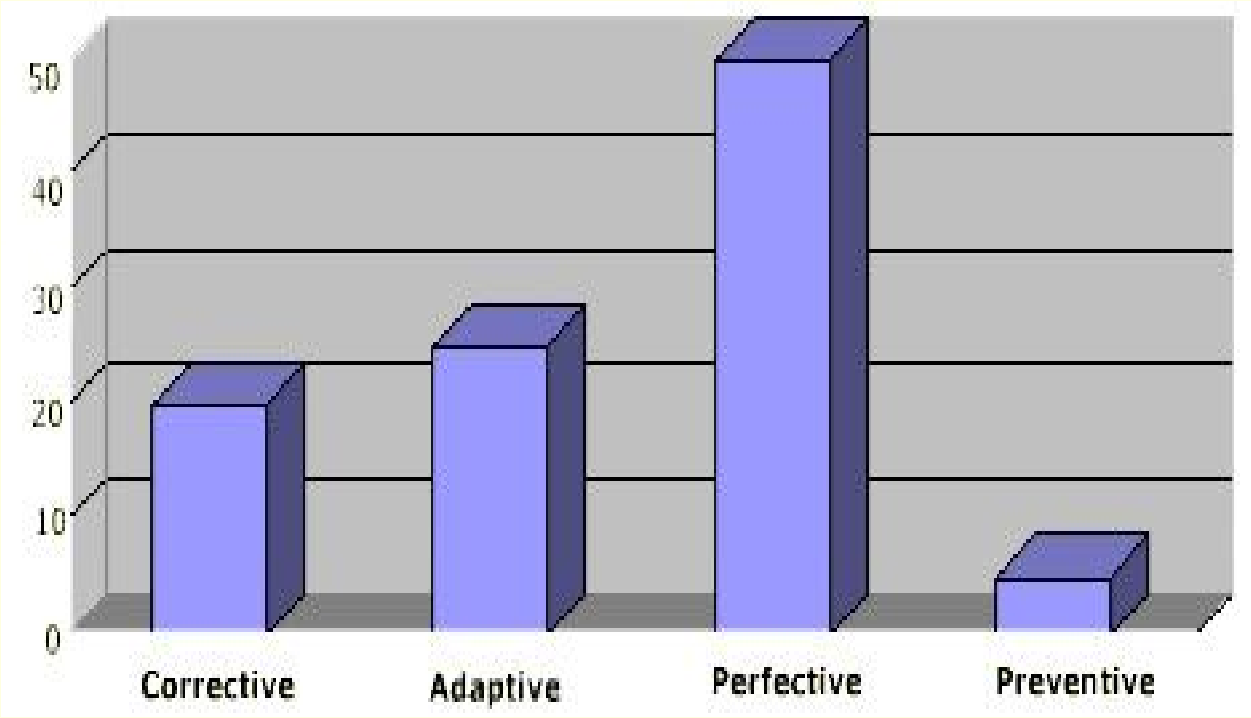
Perfective Maintenance

- The long-term effect of corrective, adaptive and perfective change is expressed in Lehman's law of increasing entropy:
- *As a large program is continuously changed, its complexity, which reflects weakening structure, increases unless work is done to maintain or reduce it.* (Lehman 1985).

Preventive Maintenance

- The IEEE defined preventative maintenance as "maintenance performed for the purpose of preventing problems before they occur" (IEEE 1219 1993).
- This is the process of changing software to improve its future maintainability or to provide a better basis for future enhancements.

Distribution of maintenance by categories



Maintenance as Ongoing Support

- This category of maintenance work refers to the service provided to satisfy non-programming related work requests. Ongoing support, although not a change in itself, is essential for successful communication of desired changes.
- The objectives of ongoing support include effective communication between maintenance and end user personnel, training of end-users and providing business information to users and their organizations to aid decision making.

- Effective communication is essential as maintenance
- Good customer relations are important for several reasons and can lead to a reduction in the misinterpretation of users change requests, a better understanding of users' business needs and increased user involvement in the maintenance process.
- Failure to achieve the required level of communication between the maintenance organization and those affected by the software changes may eventually lead to software failure.

- Training of end users - typical services provided by the maintenance organization include manuals, telephone support , help desk, on-site visits, informal short courses, and user groups.
- Business information - users need various types of timely and accurate business information (for example, time, cost, resource estimates) to enable them take strategic business decisions. Questions such as "should we enhance the existing system or replace it completely" may need to be considered.

The Importance of Categorizing Software Changes

- In principle, software maintenance activities can be classified individually.
- In practice, however, they are often intertwined.
- For example, in the course of modifying a program due to the introduction of a new operating system (adaptive change),
- obscure 'bugs' may be introduced. The bugs have to be traced and dealt with (corrective maintenance).
- Similarly, the introduction of a more efficient sorting algorithm into a data processing package (perfective maintenance) may require that the existing program code be restructured (preventive maintenance).

- Despite the overlapping nature of these changes, there are several reasons why a good understanding of the distinction between them is important.

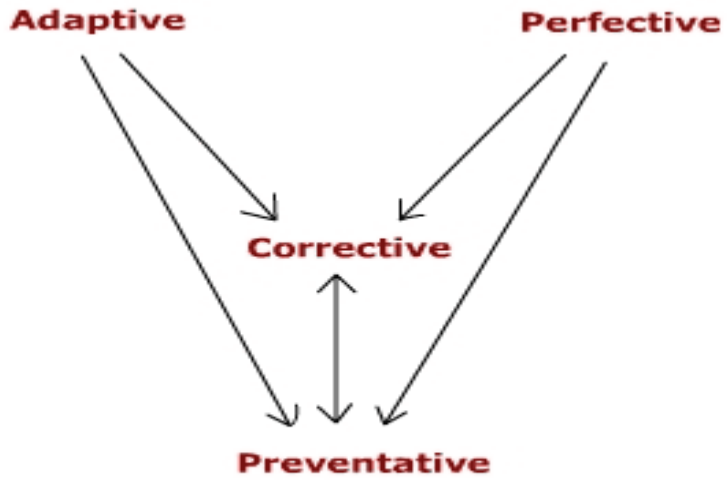


Fig 1.3 The Relationship between the different types of software change

- There are several reasons why a good understanding of the distinction between them is important.
- Firstly, it allows management to set priorities for change requests.
- Some changes require a faster response than others. Secondly, there are limitations to software change.
- Ideally changes are implemented as the need for them arises.

In practice, however this is not always possible for several reasons:

- Resource Limitations: Some of the major hindrances to the quality and productivity of maintenance activities are the lack of skilled and trained maintenance programmers and the suitable tools and environment to support their work. Cost may also be an issue.
- Quality of the existing system: In some 'old' systems, this can be so poor that any change can lead to unpredictable ripple effects and a potential collapse of the system.
- Organisational strategy: The desire to be on a par with other organizations, especially rivals, can be a great determinant of the size of a maintenance budget.
- Inertia: The resistance to change by users may prevent modification to a software product, however important or potentially profitable such change may be.

- By studying the types of maintenance activities above it is clear that regardless of which tools and development model is used, maintenance is needed.

Comparison between Development and Maintenance

In the course of designing an enhancement, the designer needs to investigate the current system to abstract the architectural and the low-level designs.

This information is then used to:

- ascertain how the change can be accommodated
- predict the potential ripple effect of the change
- determine the skills and knowledge required to do the job

Maintenance Costs as a Percentage of Total Software Life-cycle Costs

Survey	Year	Maintenance (%)
Canning	1972	60
Boehm	1973	40-80
deRose/Nyman	1976	60-70
Mills	1976	75
Zeikowitz	1979	67
Cashman and Holt	1979	60-80

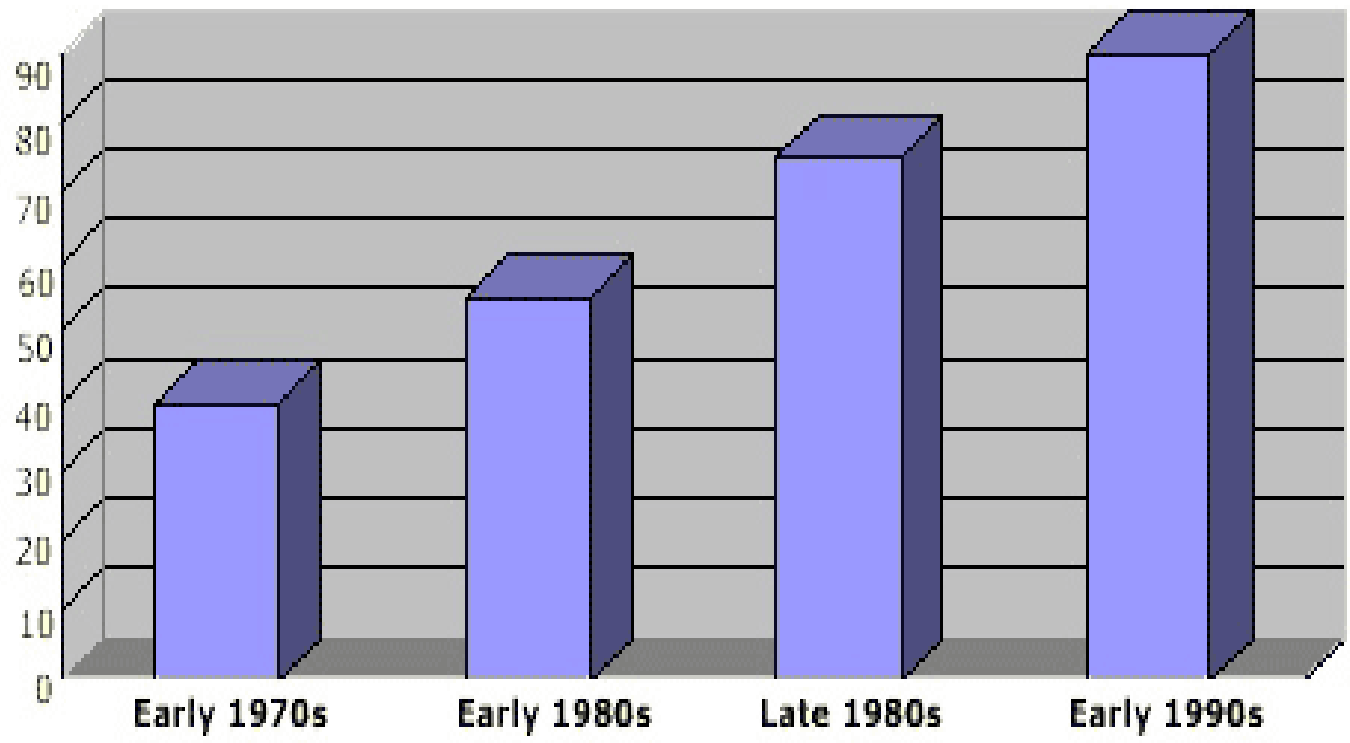


Fig 1.4 The Percentage of Software life-cycle costs devoted to maintenance.

Components of the Software Maintenance Framework

Component	Feature
1. Users & requirements	• Requests for additional functionality, error correction and improve maintainability • Request for non-programming related support
2. Organisational environment	• Change in policies • Competition in the market place
3. Operational environment	• Hardware innovations • Software innovations
4. Maintenance process	• Capturing requirements • Variation in programming and working practices • Paradigm shift • Error detection and correction

5. Software product	• Maturity and difficulty of application domain • Quality of documentation • Malleability of programs • Complexity of programs • Program structure • Inherent quality
6. Maintenance personnel	• Staff turnover • Domain expertise

Users and their Requirements

Users often have little understanding of software maintenance and so can be unsupportive of the maintenance process.

They may take the view that:

- Software maintenance is like hardware maintenance
- Changing software is easy
- Changes cost too much and take too long

Users may be unaware that their request:

- may involve major structural changes to the software which may take time to implement
- must be feasible, desirable, prioritized, scheduled and resourced
- may conflict against one another or against company policy such that it is never implemented

Organisational and Operational Environment

An environmental factor is a factor which acts upon the product from outside and influences its form or operation.

The two Categories of environment are:

- The organizational environment - e.g. business rules, government regulations, taxation policies, work patterns, competition in the market place
- The operational environment - software systems e.g. operating systems, database systems, compilers and hardware systems e.g. processor, memory, peripherals

Maintenance Process

The facets of a maintenance process which affect the evolution of the software or contribute to maintenance costs include:

- The difficulty of capturing change (and changing) requirements
- Variation in programming practice
- Paradigm shift
- Error detection and correction

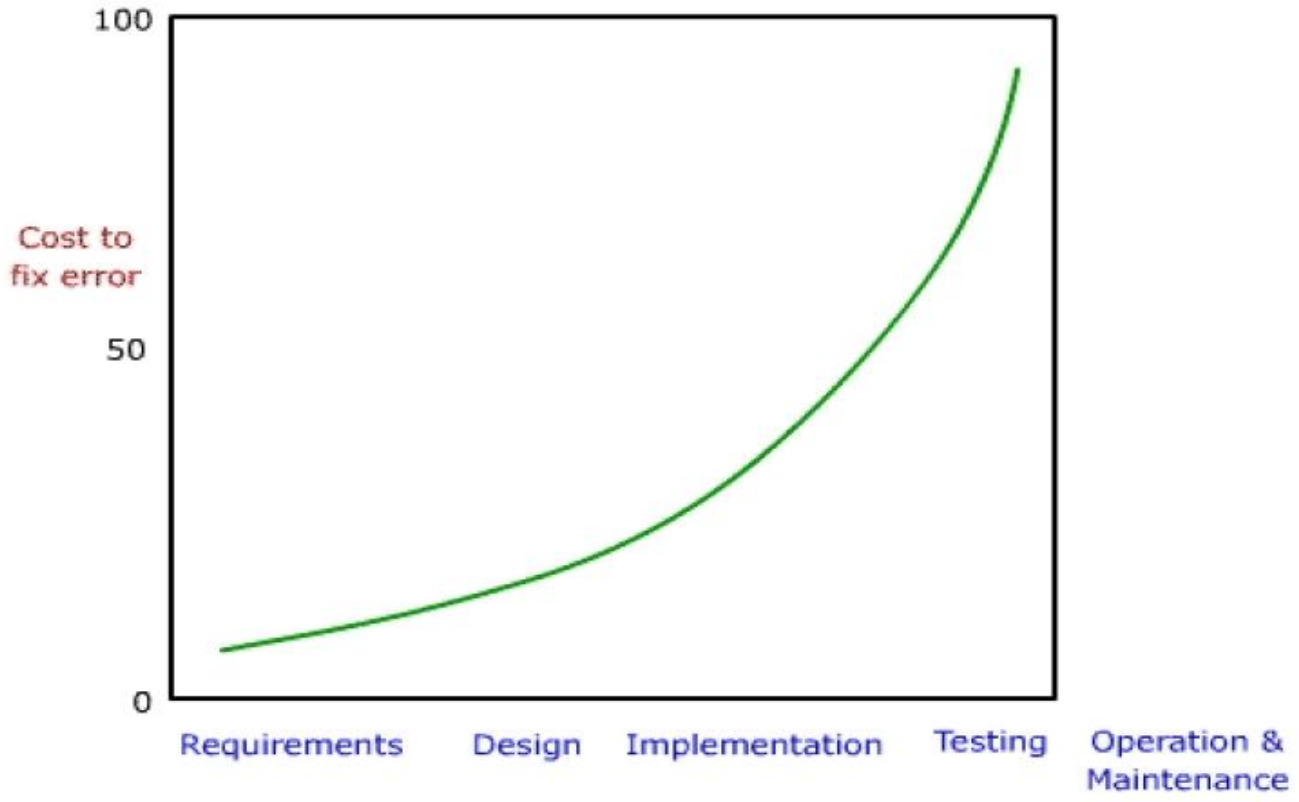


Fig 1.5 Cost of fixing errors increases in later phases of the life cycle

Aspects of a software product that contribute to the maintenance challenge include:

- Maturity and difficulty of the application domain
- Inherent difficulty of the original problem
- Quality of the documentation
- Malleability of the programs
- Inherent quality

Maintenance Personnel

- Staff turnover: Due to high staff turnover many systems end up being maintained by individuals who are not the original authors therefore a substantial proportion of the maintenance effort is spent on just understanding the code. Staff who leave take irreplaceable knowledge with them.
- Domain expertise: Staff may end up working on a system for which they have neither the system domain knowledge.
- This problem may be worsened by the absence of documentation or out of date or inadequate documentation or the application domain knowledge.

Obviously the factors of product, environment, user and maintenance personnel do not exist in isolation but interact with one another

- Relationship between product and environment
- Relationship between product and user
- Interaction between personnel and product

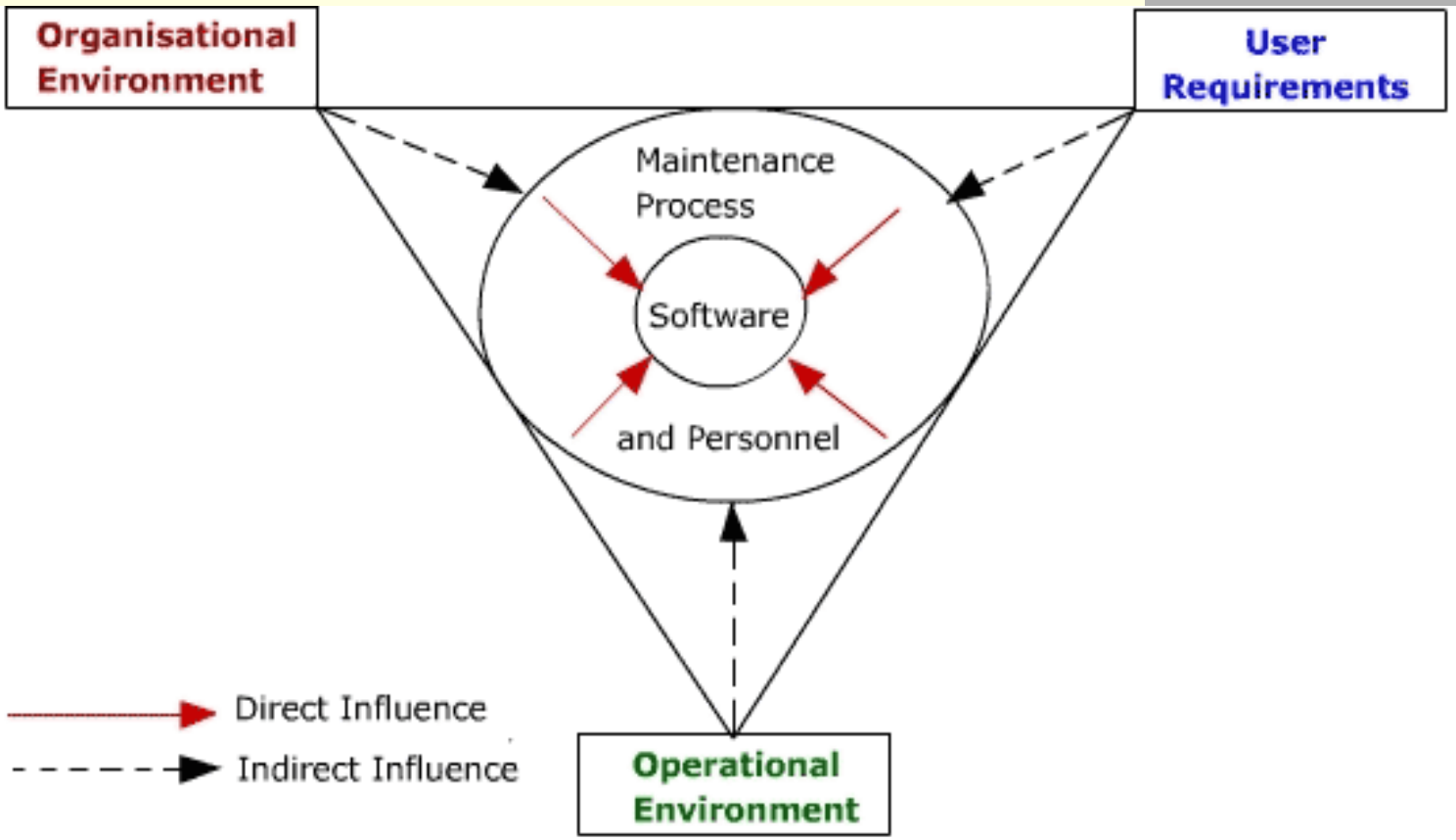


Fig 1.6 Inter-relationship between maintenance factors

Potential Solutions to the Maintenance Problem

- Budget and Effort Reallocation
- Complete Replacement of the System
- Maintenance of the Existing System