Chapter 2

The Maintenance Framework

Overview

- Definitions: Framework, Safety Critical
- Software Maintenance Framework
  - User requirements (Progressive, Regressive)
  - Organizational Environment
    - Change in Policy and Competition in Marketplace
  - Operational environment
    - Hardware and Software Innovations
  - Maintenance process
    - Capturing change requirements, Variation in programming practices, paradigm shift, dead paradigm for living systems, Error detection and correction

Overview

- Software Maintenance Framework
  - Software product
    - Application domain, quality of documentation, inherent quality, malleability of programs
  - Maintenance personnel
    - Staff turnover, Domain Expertise and working practices
- Components of Software Maintenance Framework
- Relationship between Software Maintenance Components
  - Product and Environment, Product and User, Maintenance personnel and product

Related Definitions

Framework - a set of ideas, conditions, or assumptions that determine how something will be approached, perceived, or understood.

Safety-critical - a system where failure could result in death, injury or illness, major economic loss, environmental or property damage.

Software Maintenance Framework

Def: The context and environment in which software maintenance activities are carried out is known as Software maintenance framework.

There are 6 elements of this framework.

- User requirements
- Organizational Environment
- Operational environment
- Maintenance process
- Software product
- Maintenance personnel

Components of Software Maintenance Framework

1. **User requirements:** Individuals who use the system, regardless of their involvement in its development or maintenance. After a software is After a software is operational, user may request modification for:
    1. Progressive work: To refine existing functions or to introduce new features
    2. Anti-regressive work: To make programs well structured, better documented, more understandable and capable of further development.

Components of Software Maintenance Framework

2. **Environment:** The totality of conditions and influences which act from outside upon an entity

The environments affecting software systems are

- Operating environment and
- Organizational environment

a) Operating Environment:

i) Hardware innovations: Hardware platform on which a software system runs may be subject to change. For example, when a processor is upgraded, compilers that previously produced machine code for that processor may need to be modified.

Components of Software Maintenance Framework

ii)Software innovations: Host software may warrant a corresponding modification in the software product. Example : Operating systems, Operating systems, database management systems and compilers are examples of host software systems whose modification may affect other may affect other software products.

Real world Case: **Solaris 1.x was upgraded to Solaris 2.x .** Applications that ran on Solaris 1.x was modified and users had to retrain and learn the use of new commands. More efficient and cost-effective system but the cost went beyond the retail price

Components of Software Maintenance Framework

b) Organizational Environment: Entities of the organizational environment are policies and imposed factors of business and taxation, and also competition in the marketplace.

i) **Change in policies**: Any change in business rules and taxation policies which are incorporated into programs leads to theirs corresponding modification. Example : Value Added Tax (VAT)

ii) **Competition in the marketplace:** To have a competitive edge over their rivals, substantial modifications are done. These are done to satisfy are done to satisfy customers or increase the existing 'client base'.

Components of Software Maintenance Framework

3. **Maintenance Process**: Important factors of software maintenance process are the capturing of change requirements, programming practice, programming practice, 'dead' paradigms and error detection.

- **Capturing change requirements**: This is the process of finding out exactly what changes are required. Requirements and user problems only really become clear when a system is in use. Many users know what they want but lack the ability to express it due to information gap.

- **Variation in programming practice**: This refers to differences in approach used for writing and maintaining programs. Traditional guidelines would include: avoiding the use of 'GOTOs', the use of meaningful identifier names, logical program layout, and use of program commentary to document design and implementation rationale.

Components of Software Maintenance Framework

- **Paradigm shift:** This refers to an alteration in the way we develop and maintain software. Many programs in operation and in need of maintenance that were developed without the means to take advantage of the more advanced and more recently developed technique

- **Dead' paradigms for 'living' systems:** Many 'living systems' are developed using 'dead paradigms'. The resulting system is satisfactory only at the point at which it is delivered to the user. Thereafter, it becomes difficult - with few exceptions - to accommodate the changing needs of the users and their organisations.
- **Error detection and correction:** 'Error-free' software is non-existent. Software products have 'residual' errors which are difficult to detect even with the most powerful testing techniques and tools. The later these errors are discovered during the life-cycle of a software product, the more expensive they are to correct.

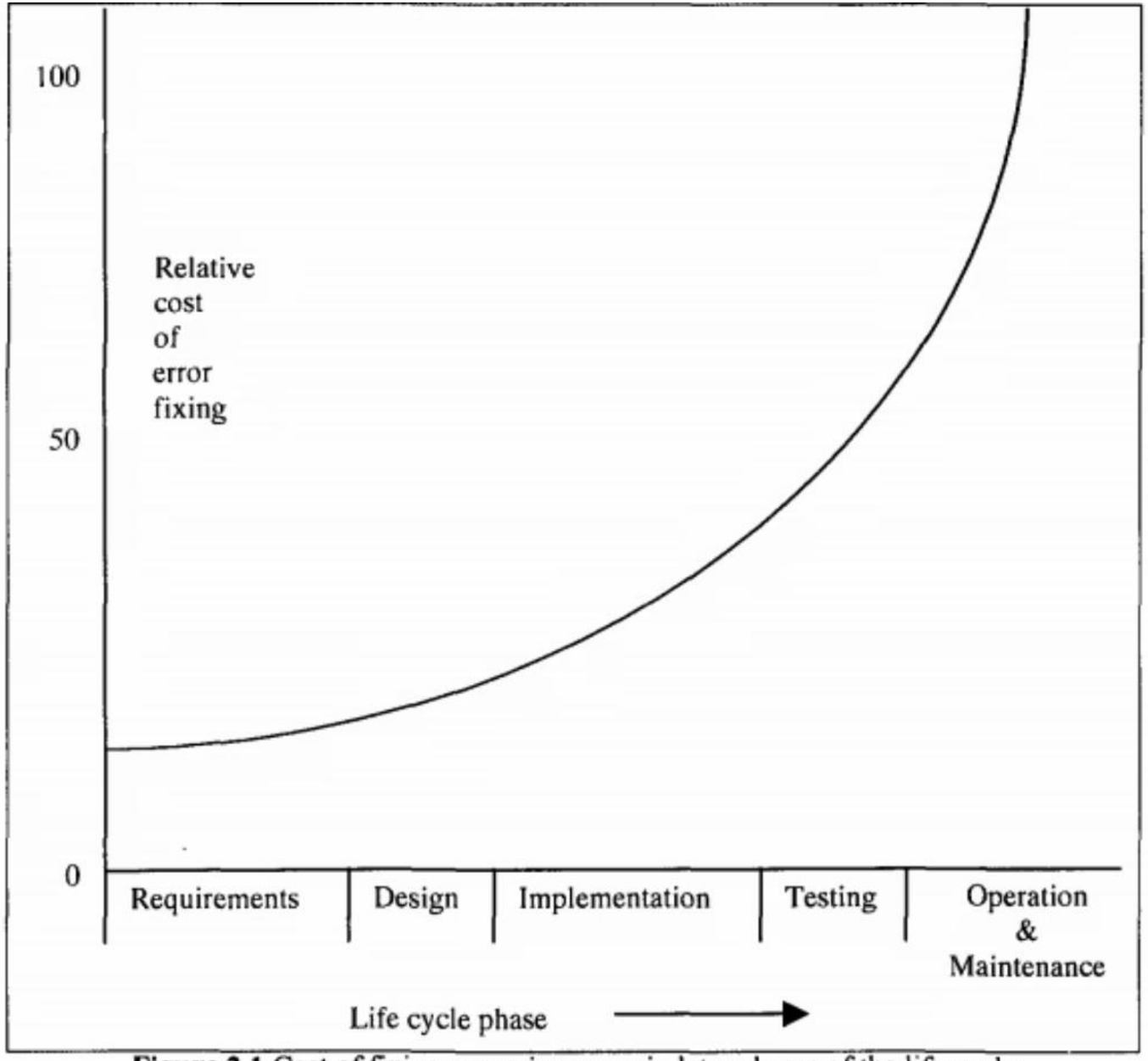


Figure 2.1 Cost of fixing errors increases in later phases of the life cycle

Components of Software Maintenance Framework

4. **Software Product**: There is a tendency to assume that computer programs are static artefacts that do not change once they correctly implement the agreed system specification.

it is not just the programs themselves but also the accompanying documentation and documentation and operating procedures that are subject to such changes.

- **Maturity and difficulty of the application domain**: Applications that have been widely used and well understood are less likely to undergo substantial modification

For example, accounts and payroll packages are likely to be subject to fewer requests for changes in requirements than a medical information system.

Components of Software Maintenance Framework

- **Quality of the documentation**: The lack of up-to-date systems' documentation is one of the major problems that software maintainers face. Programs are often modified without a corresponding update of the documents affected.
- **Changeability of the programs:** The malleable or 'soft' nature of software products makes them more vulnerable to undesirable modification than hardware items. Ad hoc software changes may have unknown and even fatal repercussions. This is particularly true of safety-related or safety critical systems.
- **Inherent quality:** The nature of the evolution of a software product is very closely tied to the nature of its associated programs. This programs. This tendency for the system to decay as more changes are undertaken implies that preventive maintenance needs to be needs to be undertaken

Components of Software Maintenance Framework

5. **Maintenance Personnel:** People are involved at all stages of the maintenance process. Maintenance personnel include maintenance maintenance managers, analysts, designers, programmers and testers.

- **Staff turnover:** Due to the high staff turnover within the Information Technology industry, especially with regard to software maintenance, most systems end up being maintained by people who are not the original authors. So, they spend a substantial proportion of the maintenance effort just understanding the code.
- **Domain expertise**: The migration of staff to other projects or departments can mean that they end up working on a system for which they system for which they have neither the system domain knowledge nor the application domain knowledge. Programmers can introduce can introduce changes to programs without being aware of their effects on other parts of the system.

Components of Software Maintenance Framework

- **Working practices**: Software systems do not change unless they are changed by people. Factors that can make the job more difficult include such things as

- a maintainer's desire to be creative (or 'clever')- the use of undocumented assumption sets- assumption sets- undocumented design and implementation decisions. It should always be should always be kept in mind that after time has elapsed, programmers find it difficult to difficult to understand their own code.

Relations Between the Maintenance Components

Relations Between Product and Environment: A software product does not exist in a vacuum, rather it can be seen as an entity which is hosted entity which is hosted by its organisational and operational environments.

Relation between product and user: One of the objectives of a software product is to serve the needs of its users. The needs of the users change all the time and thus the product needs to adapt.

Interaction between personnel and product: The maintenance personnel who implement changes are themselves the conduit through which change is implemented. The type of maintenance process used and the nature of the maintenance personnel themselves, will affect the quality of the change.

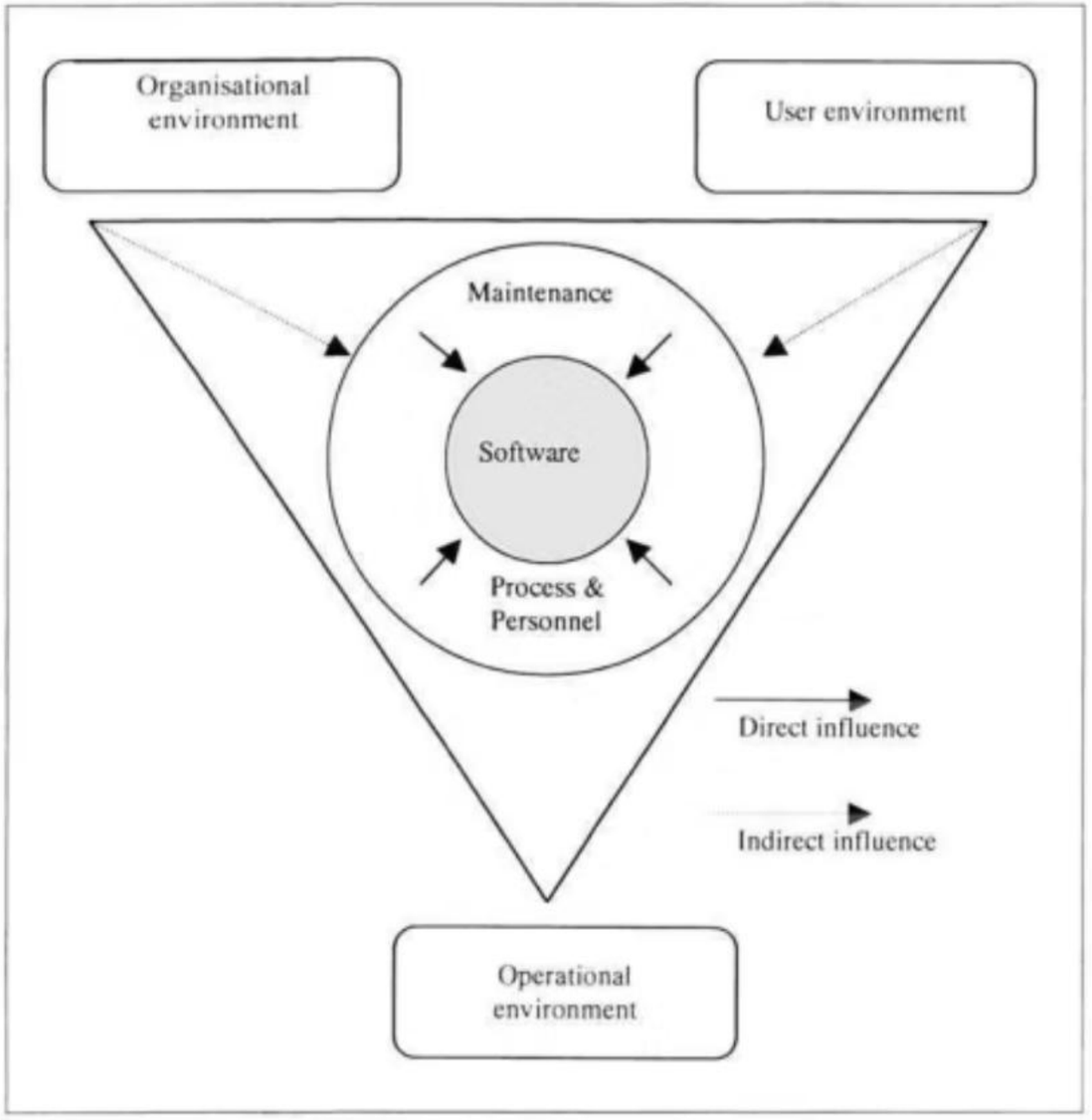


Figure 2.2 Inter-relationship between maintenance factors

CH5: The Maintenance Process

Software Life Cycle vs Software Process

The **Software Life Cycle** refers to the stages a software product goes through from its initial conception to its final retirement. These stages typically include requirements analysis, design, implementation, testing, deployment, and maintenance.

The **Software Process** is the set of activities, methods, and practices used to develop and maintain software. It defines how the software life the software life cycle stages are executed and managed to ensure quality and efficiency.

While the software life cycle describes *what* phases a software product experiences, the software process specifies *how* those phases are phases are carried out.

Software Maintenance Models

Def: In software terms the model is the abstract representation of the software production process, the series of changes through which a software product evolves from initial idea to the system in use.

For software maintenance, it is the representation of those parts of the process specifically pertaining to the evolution of the software.

Software Maintenance Process

Software Maintenance is an important phase of Software Development Life Cycle (SDLC), and it is implemented in the system through a proper software maintenance process, known as **Software** known as **Software Maintenance Life Cycle (SMLC)**. This life cycle consists of seven different phases, each of which can be used in iterative manner and can be extended so that customized items and customized items and processes can be included. These seven phases of Software Maintenance process are:

1. **Identification Phase:** In this phase, the requests for modifications in the software are identified and analysed. Each of the requested modification is then assessed to determine and classify the type of maintenance activity it requires. This is either generated by the system itself, via logs or error messages, or by the user.
2. **Analysis Phase:** The feasibility and scope of each validated modification request are determined and a plan is prepared to incorporate the changes in the software. The input attribute comprises validated modification request, initial estimate of resources, project documentation, and repository information. The cost of modification and maintenance is also estimated.
3. **Design Phase:** The new modules that need to be replaced or modified are designed as per the requirements specified in the earlier stages. Test cases are developed for the new design including the safety and security issues. These test cases are created for the validation and verifcation of the system.
4. **Implementation Phase:** In the implementation phase, the actual modification in the software code are made, new features that support the specifications of the present software are added, and the software are added, and the modified software is installed. The new modules are coded with the assistance of structured design created in the design phase.
5. **System Testing Phase:** Regression testing is performed on the modified system to ensure that no defect, error or bug is left undetected. Furthermore, it validates that no new faults are introduced in the software as a result of maintenance activity. Integration testing is also carried out between new modules and the system.
6. **Acceptance Testing Phase:** AAcceptance testing is performed on the fully integrated system by the user or by the third party specified by the end user. The main objective of this testing is to verify that all the features of the software are according to the requirements stated in the modification request.
7. **Delivery Phase:** Once the acceptance testing is successfully accomplished, the modified system is delivered to the users. In addition to this, the user is provided proper consisting of manuals and consisting of manuals and help files that describe the operation of the software along with its hardware specifications. The final testing of the system is done by the client after the system is after the system is delivered.

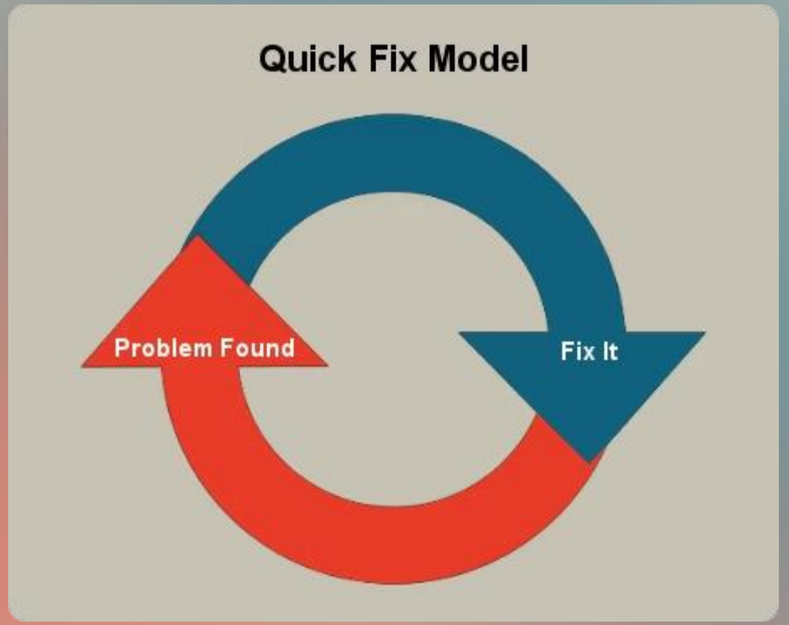
Software Maintenance Models

Software maintenance involves the modification of a software product after delivery to correct faults, improve performance, or adapt it to a changed environment. Various models guide the maintenance process to efficiently manage updates, fixes, and enhancements throughout the software's lifecycle. Understanding these models helps organizations plan and execute maintenance activities effectively.

Quick Fix model

This is an ad hoc approach used for maintaining the software system. The objective of this model is to identify the problem and then fix it as quickly as possible.

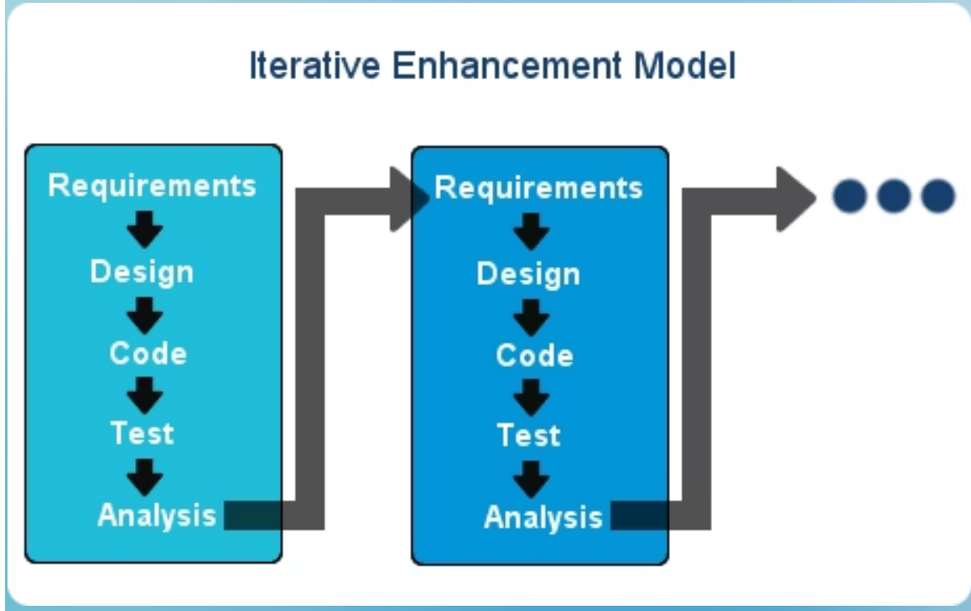
The advantage is that it performs its work quickly and at a low cost. This model is an approach to modify the software code with little consideration for its impact on the overall structure of the software system.



Iterative Enhancement Model Model

Iterative enhancement model considers the changes made to the system are iterative in nature. This model incorporates changes in the software based on the analysis of the existing system.

It assumes complete documentation of the software is available in the beginning. Moreover, it attempts to control complexity and tries to maintain good design.



Iterative Enhancement Model is divided into three stages:

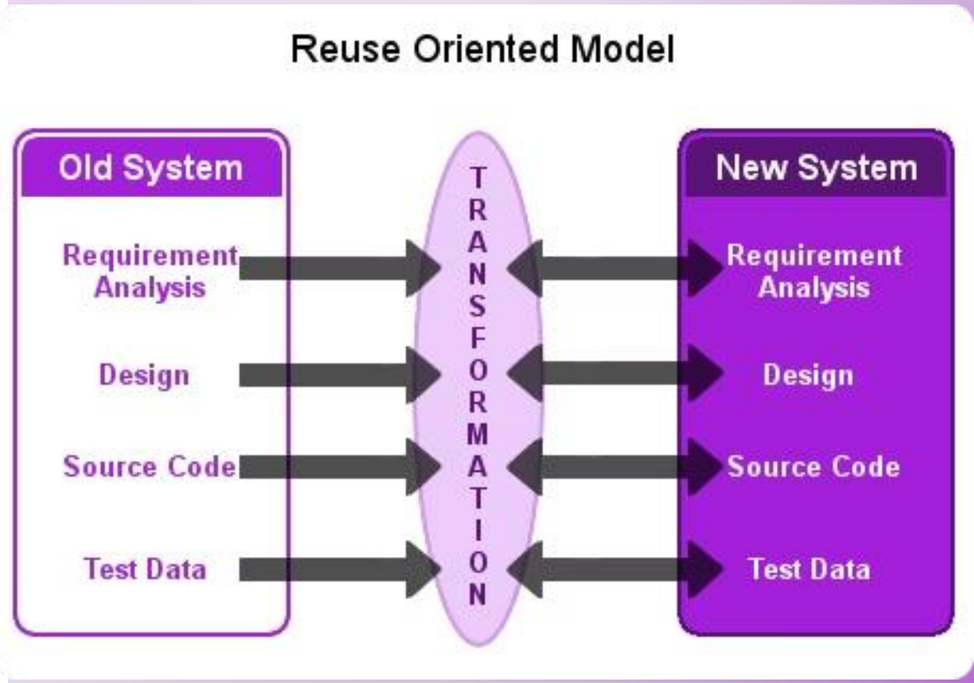
1. Analysis of software system.
2. Classification of requested modifications.
3. Implementation of requested modifications.

Reuse Oriented Model

The parts of the old/existing system that are appropriate for reuse are identified and understood, in Reuse Oriented Model.

These parts are then go through modification and enhancement, which are done on the basis of the specified new requirements.

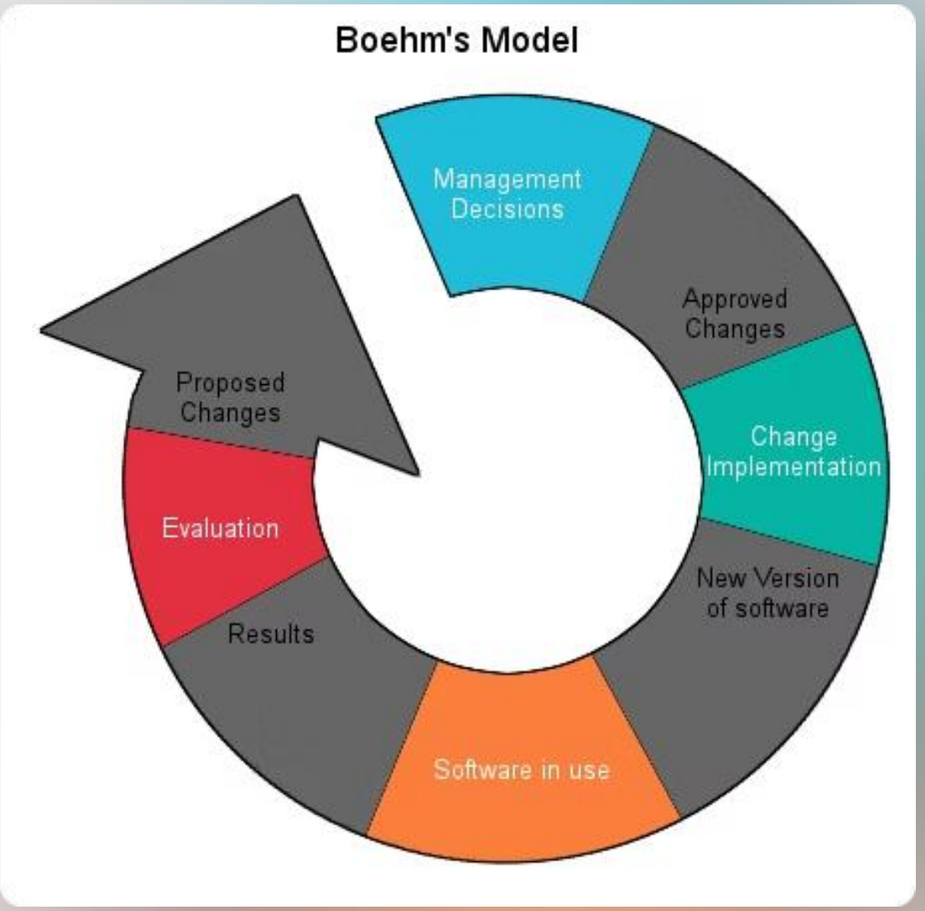
The final step of this model is the integration of modified parts into the new system.



Boehm's Model

Boehm’s Model Boehm's model performs maintenance process based on the economic models and principles.

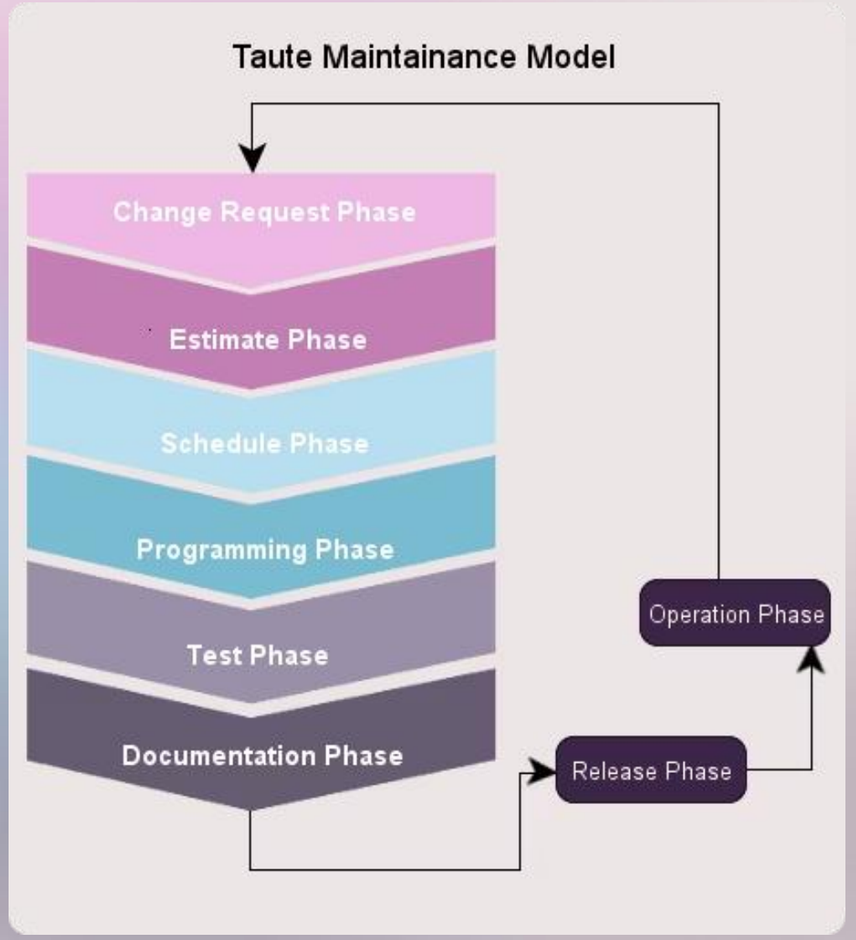
It represents the maintenance process in a closed loop cycle, wherein changes are suggested and approved first and then executed.



Taute Maintenance Model

Named after the person who proposed the model, Taute's model is a typical maintenance model that consists of eight phases in cycle fashion.

The process of maintenance begins by requesting the change and ends with its operation.



Process Maturity

Some companies carry on successful operations for a long time reliant on a few very good programmers. If they do not put resources into building the maturity of the processes themselves, there will come a point where the operation cannot continue.

Organizations need a means by which to assess the maturity and effectiveness of their processes.

- Software Capability Maturity model
- Capability Maturity Model Integration