Fundamental of Software Engineering
CSE 3205

Chapter-Seven

Software Maintenance

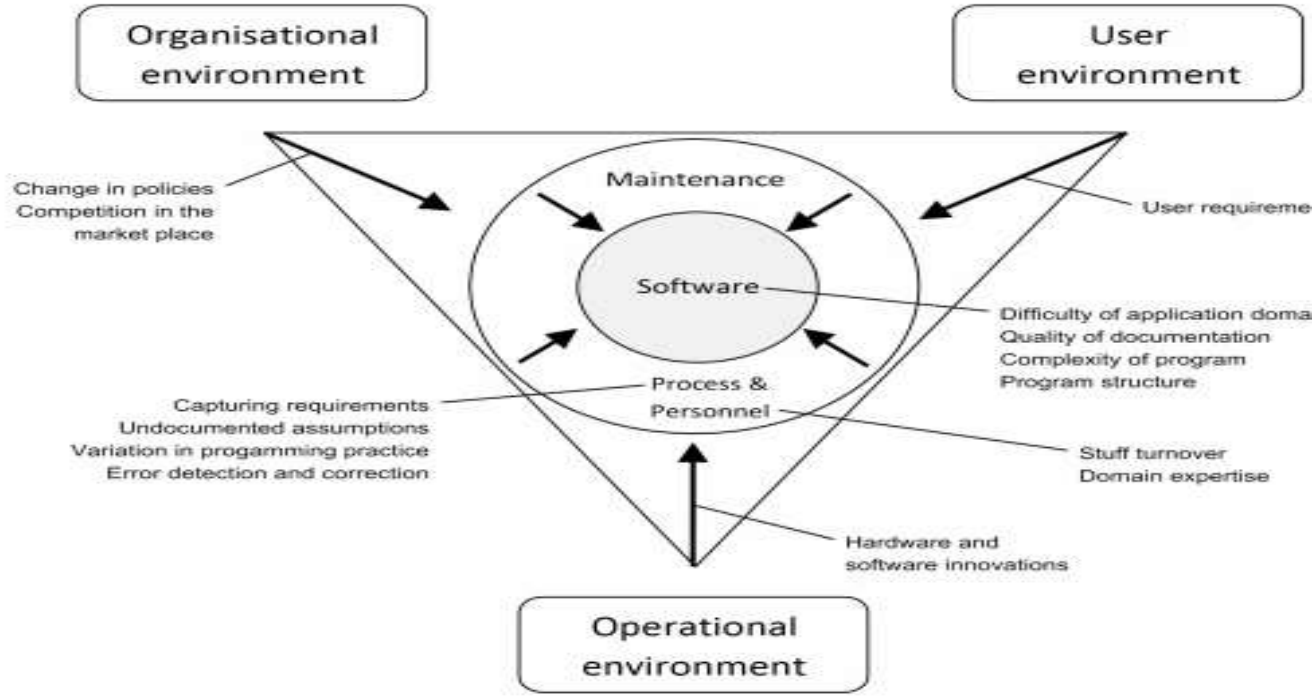
School of Electrical Engineering and Computing
Department of Computer Science and Engineering
ASTU

9-Feb-24

Software maintenance

- Software maintenance is the process of changing a system after it has been delivered.
- Modifying a program after it has been put into use.
- Maintenance does not normally involve major changes to the system's architecture.
- Changes are implemented by modifying existing components and adding new components to the system.

Maintenance Framework



External influence factors

- **User environment**
  The user requirements involve requests for additional functionality and error correction as well as requests for non-programming-related support.
- **Operational environment**
  The changes here can be the exchange of the hardware platform or of the software system, e.g. a new operating system or a different database management system.
- **Organizational environment**
  The changes in the organizational environment comprise changes in policies, e.g. changes in business rules or taxation policies and changes which enable to stay competitive.

Internal Influence Factors

Maintenance process

The maintenance process itself is the most important part in the software maintenance framework. It covers the following tasks:

* Capturing change requirements - process of finding out what changes are required.
* Variation in programming practice - the differences in the approaches used for writing and maintaining programs. Basic guidelines (e.g. meaningful identifiers, no 'GOTO's) avoid variations in the code.
* Paradigm shift - alteration of the way a software is developed and maintained. There are still many systems in use developed in low-level programming languages.

Internal Influence factors (2)

- **Software to maintain**
  The main aspects of the software itself are the difficulty of the application domain, the quality of the documentation (obsolete or no documentation at all) and the structure of the program.
- **Maintenance personnel**
  The high staff turnover is a big problem in the field of software maintenance. Most software systems are maintained by people who are not the original authors of the program. The maintenance personnel often lacks expertise in the domain. Programmers can introduce errors in other parts of the software by changing one part of the software (ripple effect).
  Good employees for software maintenance have the following abilities:
  * Knowledge of the programming language
  * Abstraction, compression of information, and analytical abilities
  * Impact analysis abilities

  Often the maintenance personnel is inexperienced. Beath and Swanson [30] reported that 25 % of the maintenance personnel are students and 60 % are newly hired people.

Need for software maintenance

- Maintenance to repair software faults – Changing a system to correct deficiencies in the way meets its requirements.
- Maintenance to adapt software to a different operating environment
  - Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation.
- Maintenance to add to or modify the system's functionality – Modifying the system to satisfy new requirements.

Sources of change

- New business or market conditions which cause changes in product requirements or business rules.
- New customer needs that demand modification of data, functionality or services delivered by the system.
- Reorganization and/or business downsizing that changes priorities team structures
- Budgetary or scheduling constraints that cause a redefinition of the system
- Most changes are justified

The Cost of Maintenance

- One study found
  - Requirements Definition 3%
  - Preliminary Design 3%
  - Detailed Design 5%
  - Implementation 7%
  - Testing 15%
  - Maintenance 67%
- Another study found at least 50% of effort spent on maintenance
- Another study found between 65% and 75% on maintenance
- In embedded real-time systems, maintenance costs may be up to 4 times development costs

Why is Maintenance so Costly

- Most software is between 10 and 15 years old
- Much of that software is showing its age as it was created when program size and storage space were far more important factors
  - This has lead to inflexible designs, coding and documentation
- Maintenance is usually done by inexperienced staff unfamiliar with the application
- Developers don't like maintenance
- Changes often cause new faults in the system
- Changes tend to degrade the structure of a program
- Changes are often not documented

Factors Affecting Maintenance Costs(1)

- Module Independence
  - the ability to modify one part of the system
  - potential advantage of OO
- Programming Language
  - the higher the level of the language, the cheaper the maintenance
- Programming Style
  - the way in which a program is written
- Program Validation and Testing
  - the more time and effort spent on design validation and program testing, the fewer errors and the less the need for corrective maintenance

Factors Affecting Maintenance Costs (2)

- Quality of Program Documentation
  - the better the documentation, the easier it is to maintain
- Configuration Management Techniques
  - keeping track of all the system documents and ensuring they are consistent is a major cost of maintenance, therefore good CM tools and practices reduce this cost
- Application Domain
  - the less well-understood the domain, the greater the likelihood of the need for adaptive maintenance as users and developers begin to understand the domain
- Staff Stability
  - Maintenance costs are reduced if developers have to maintain their own systems, highly unusual though over the life of a system

Factors Affecting Maintenance Costs (3)

- Age of the System
  - the older the system, the more likely that its structure has degraded and the harder the maintenance will be
  - Attracting staff who know the old languages/databases/operating systems becomes harder and more expensive
- Dependence of the System on the External Environment
  - the higher the dependence, the far more likely that adaptive maintenance will be needed
- Hardware Stability
  - If the hardware platform will not change over the life of the system, maintenance for this reason will not be needed

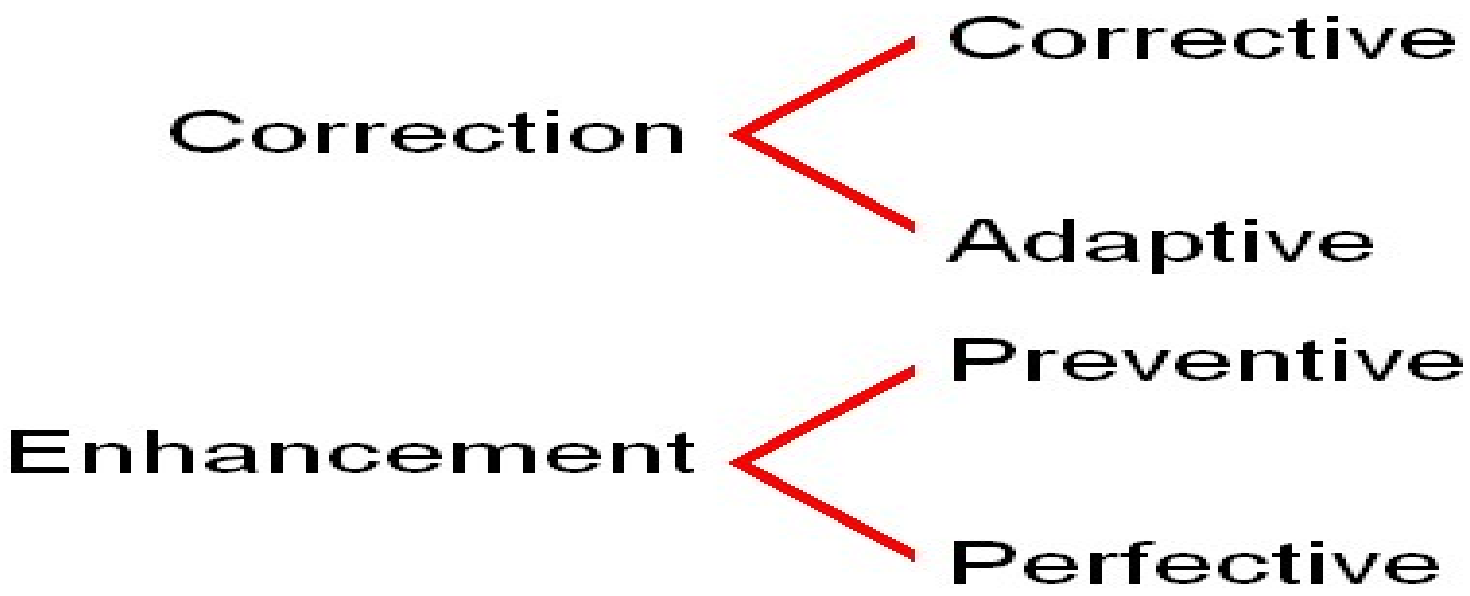
Differences between Maintenance and New Development

- Constraints of an Existing System
  - Changes must conform or be compatible with an existing architecture, design and code constraints
- Shorter Time Frames
  - Development spans 6 months upwards
  - Maintenance spans hours or days up to 6 months
- Available Test Data
  - Development creates all test data from scratch
  - Maintenance uses existing test data with regression testing, creating new data for the changes

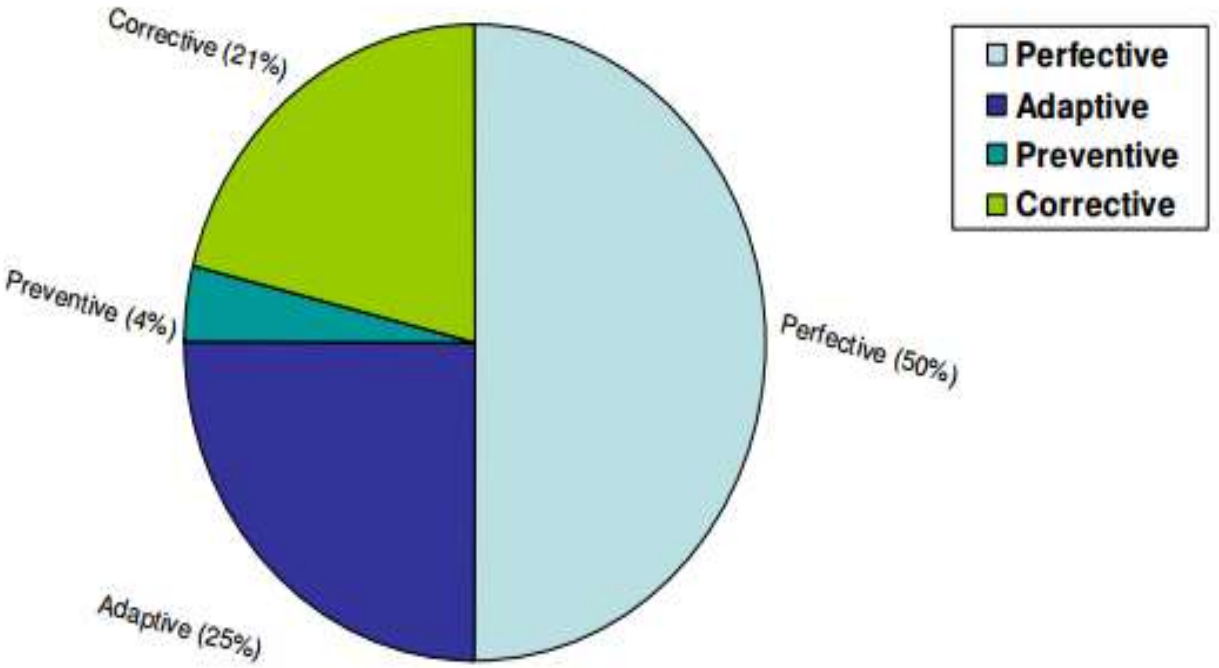
Types of Maintenance

- So how and why do we spend so much time and effort on maintenance?
- Maintenance is much more than fixing bugs
- Commonly divided into four main categories
  - Corrective Maintenance
  - Adaptive Maintenance
  - Perfective Maintenance
  - Preventive maintenance

Types of Maintenance



Distribution of Maintenance effort



Corrective Maintenance(1)

- Focused on fixing failures
- Is a reactive process
  - failures and their associated faults generally need to be corrected either immediately or in the near future
- Failures vary in their cost to correct
  - Coding - usually relatively cheap
  - Design - more expensive as they may require changes to several program components
  - Requirements - most expensive - may require extensive system redesign
- Design and Requirements are the source of approximately 80% of failures

Corrective Maintenance (2)

- Fixing a fault has a 20 to 50% chance of introducing another fault
- Reasons for new faults include
  - the ripple effect, where a change in one area may cause changes in seemingly unrelated areas
  - Person who makes the repair is generally not the person who wrote the code or designed the system
- Two types of corrective maintenance
  - Emergency Repairs - short time frame, often a single program, failure needs to be repaired as soon as possible
  - Scheduled Repairs - failure doesn't need immediate attention, re-examination of all emergency repairs

Adaptive Maintenance

- The evolution of the system to meet the needs of the user and the business
- Caused by
  - internal needs
  - external competition
  - external requirements e.g. changes in law
- Essentially we are introducing new requirements to the system
- Therefore we should treat like a new development in our approach and methods

Perfective Maintenance

- Old proverb says "If it isn't broken, don't fix it"
- Perfective maintenance ignores this ancient piece of wisdom
- Is about improving the quality of a program that already works
- Aim to achieve
  - reduced costs in using the system
  - increase maintainability of the system
  - more closely meet the users' needs

Perfective Maintenance (2)

- Includes all efforts to polish or refine the quality of the software or the documentation
- Important that the potential benefits of the perfective maintenance outweigh
  - the costs of the maintenance
  - and the opportunity costs of improvements elsewhere or using the resources on new developments
- Therefore before performing perfective maintenance, one should go through an analysis process
- Nevertheless, a little perfective maintenance can have dramatic effects

Preventative Maintenance

- Can be seen as radical perfective maintenance or as an alternative to maintenance
- More commonly known as Software Re- engineering
- Taking a legacy system and converting its structure or converting to a new language
- Old system starts as a specification for the new system
- Common method now is known as wrappers where an entire system is placed in an OO wrapper and treated as one large object

Alternatives to Maintenance

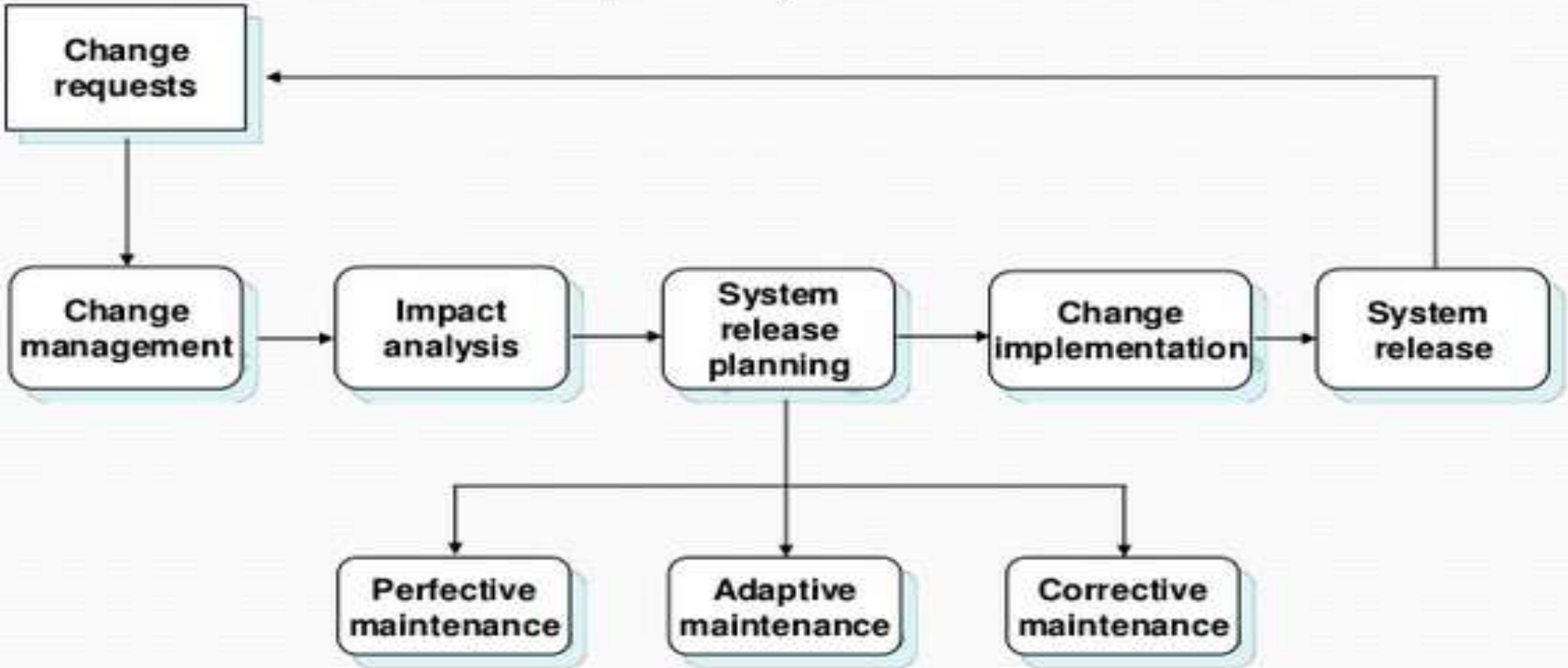
- Sometimes, maintenance is not enough on its own
- Partial restructuring integrated with adaptive maintenance
  - use for orderly improvement with each system release
- Complete restructuring or overhaul of the existing code
  - use on highly maintenance-prone system

Alternatives to Maintenance (2)

- Complete redesign and rewrite
  - Use when more than 20% of program must be changed
  - Use when program is being upgraded to a new technology
  - Don’t use when the design and function of a system are not known
- Retirement of the system and complete redevelopment
  - Use when moving to a new technology
  - use when the costs of maintaining the software and the hardware exceed the cost of re-development

The Maintenance Process

Below is an ideal process, that is often not achieved



The Maintenance Process (2)

- Change Management
  - Uniquely identify, describe and track the status of all change requests
- Impact Analysis
  - Identifies all systems and system products affected by a change request
  - makes an estimate of the resources needed to effect the change

  analyses the benefits of the change
- System Release Planning
  - to establish the schedule and contents of a system release
  - don't want each change request released as they are processed

The Maintenance Process (3)

- Change Implementation
  - Design Changes
  - Coding
  - Testing - must perform regression testing
- System Release includes
  - documentation
  - software
  - training
  - hardware changes
  - data conversion

Maintenance Problems

- Someone else's program.
- Developer not available.
- Proper documentation doesn't exist.
- Not designed for change.
- High staff turnover
- Maintenance activity not highly regarded.

Potential Solutions to Maintenance Problems

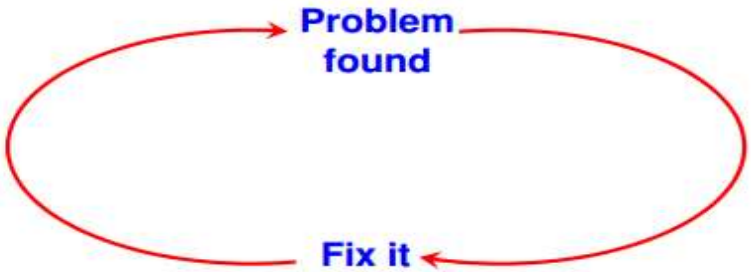
- Budget and effort reallocation
- Complete replacement of the system
- Maintenance of existing system

Maintenance Models(1)

1. Quick-fix Model

This is basically an adhoc approach to maintaining software.

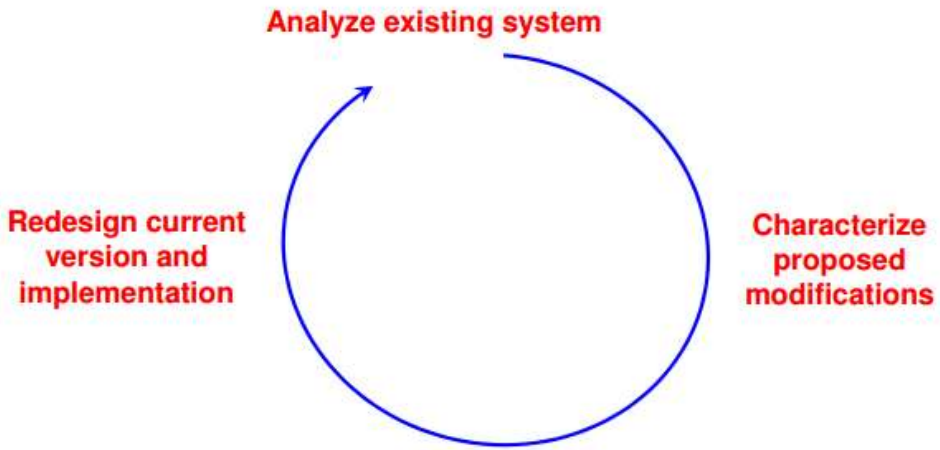
It is a fire fighting approach, waiting for the problem to occur and then trying to fix it as quickly as possible.



Maintenance Models(2)

2. Iterative Enhancement Model

- Analysis
- Characterization of proposed modifications
- Redesign and implementation

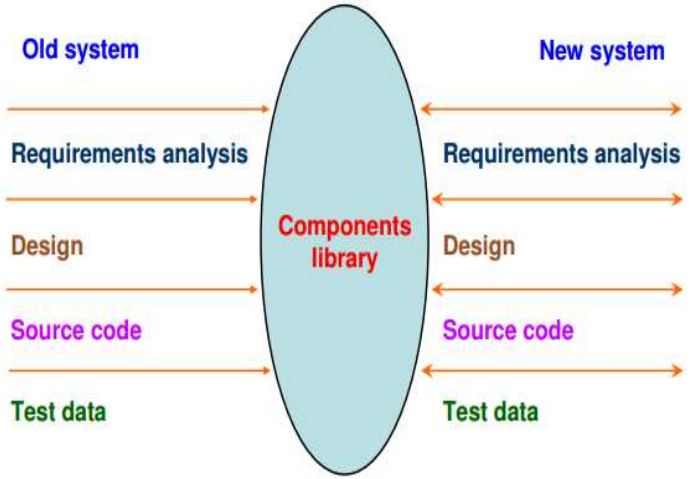


Maintenance Models(3)

3. Reuse Oriented Model

The reuse model has four main steps

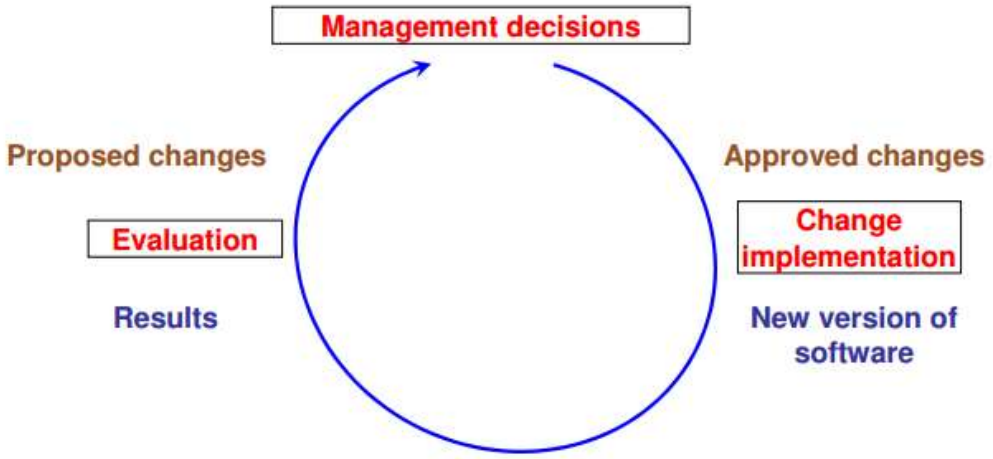
1. Identification of the parts of the old system that are candidates for reuse.
2. Understanding these system parts.
3. Modification of the old system parts appropriate to the new requirements.
4. Integration of the modified parts into the new system.



Maintenance Models(4)

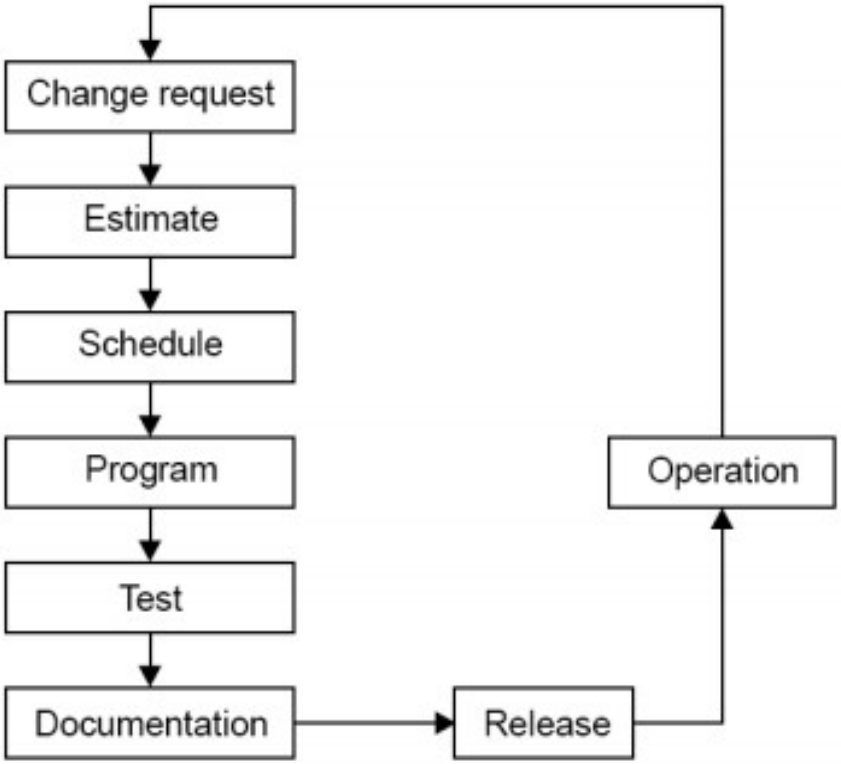
4. Boehm's Model

- 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



Maintenance Models(5)

5. Taute Maintenance Model



Maintenance Models(6)

6. 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

Lehman's Laws of Software Change

- Five laws based upon the growth and evolution of a number of large software systems

1. Law of Continuing Change
    - A program used in a real-world environment necessarily must change or become less useful in that environment

2. Law of Increasing Complexity
    - As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure

Lehman's Laws of Software Change (2)

3. Law of Large Program Evolution
   - Program evolution is a self-regulating process. System attributes such as size, time between releases and the number of reported errors are approximately invariant for each system release
4. Law of Organizational Stability
   - Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development
5. Law of Conservation of Familiarity
   - Over the lifetime of a system, the incremental change in each release is approximately constant

Motivating Maintenance Staff

- Often considered a dead-end in organizations as well as being boring!
- Critical to the success of the organization
- Possible strategies
  - Couple software objectives to organizational goals
  - Couple software maintenance rewards to organizational performance
  - Integrate software maintenance personnel into operational teams
  - Create a discretionary, perfective/preventative maintenance budget which allows the maintenance team to decide when to re-engineer the system
  - Involve maintenance staff early in the software process during standards preparation, reviews and test preparation

Summary

- Maintenance
  - Important, difficult and costly
  - Can, and should, be manage
  - Has a bad reputation but can and should be challenging and rewarding

- legacy systems a significant increasing problem
  - Number of approaches to dealing with legacy systems
  - Many involve transformation to OO and/or component based paradigms (e.g., Abstraction / high cohesion and Encapsulation / low coupling)

  - The business value of a legacy system and the quality of the application
    - software and its environment should be assessed to determine whether the system should be replaced, transformed or maintained

Summary

- Maintenance is a major cost for software and must be planned for during the entire life cycle.
- Design workflow —use information-hiding techniques
- Implementation workflow — good coding style
- Documentation must be complete, correct, and current.
- During maintenance, maintainability must not be compromised.
- Maintenance is so critical and challenging that the best people should be put on the task and rewarded accordingly.