Intro

Software Maintenance

- Changes to a software
- Software is evolution
- Lasts longer than development
- more jobs

Challenges

- Constraints
- team instability
- limited domain knowledge
- lack of documentation
- limited access to original developers.

Why ?

- to fix bugs
- to implement enhancements
- to increase non-functional qualities
- to decrease complexity
- to make it work in new environment.
- delete unused functionalities

Notes:

Definition: A process of modifying a software system after delivery to correct faults, improve it other attributes.

diff between

S/W product	S/W maintenance
It is the result of software development	It is the result in a service being delivered to the customer.

The types of Maintenance Activities:

① **Corrective:** Taking existing code & correcting a fault that deviates from its document requirements.

② Adaptive: Adding new features to the existing code.

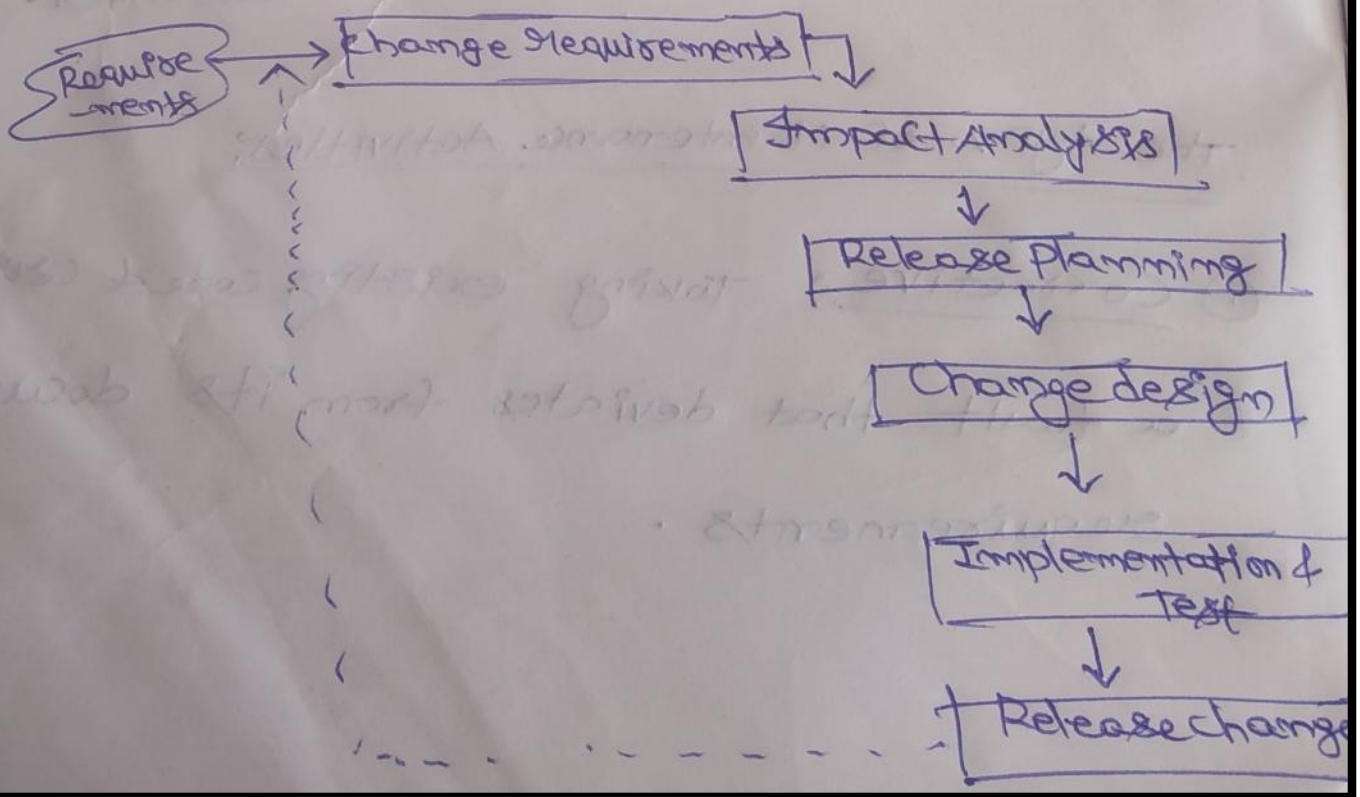
③ Perfective: Typically made to improve the maintainability of code.

④ Inspection: These are usually made as a result of code inspection to reduce the likelihood of a failure.

Maintenance process

Maintenance process vary considerably depending on the type of software being maintained.

→ the most expensive part of software life cycle is software maintenance process.

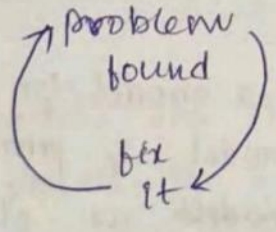


(6) Explain in detail about software Maintenance process Model.

Ans: A process model is the representation of progress of the process. The process model organize maintenance into a sequence of related activities or phases & define the order in which these phases are to be executed.

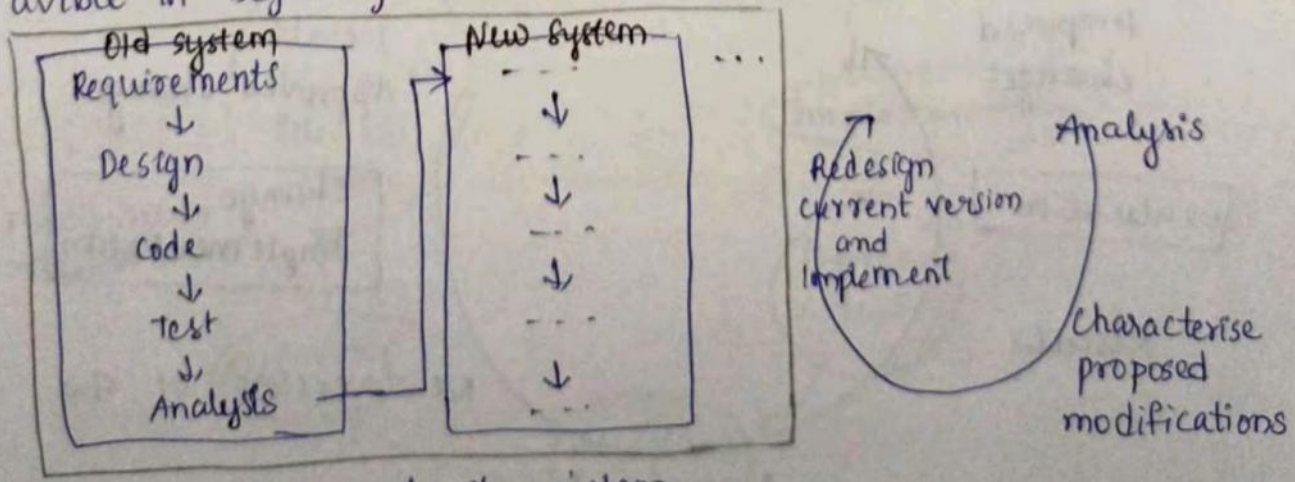
To overcome internal & external software problems, maintenance process models are proposed. These follow various approaches & techniques to make it cost effective. Few of them are:

Quick-Fix Model: Basically an ad-hoc approach for maintaining software objective is to identify problem & fix it as quickly as possible. Fire-fighting approach. Modifies software code with little consideration for its impact on overall structure of software system.



It works quickly
It is low cost.

Iterative Enhancement Model: changes made to system are iterative in nature. Incorporates changes based on analysis of existing system. It assumes complete documentation of s/w is avlble in beginning. Tries to maintain good design & control complexity.

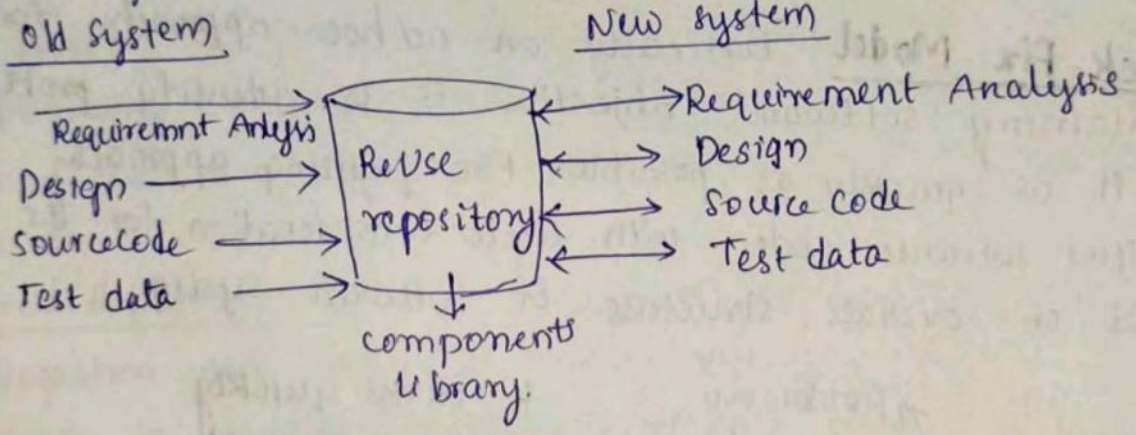


3 stages:
1. Analysis of s/w system
2. Classification of requested modifications
3. Implementation of requested modifications

The Re-Use Model: It begins with requirement analysis & design of a new system & reuses the appropriate requirements, design, code & tests from earlier versions of existing s/w

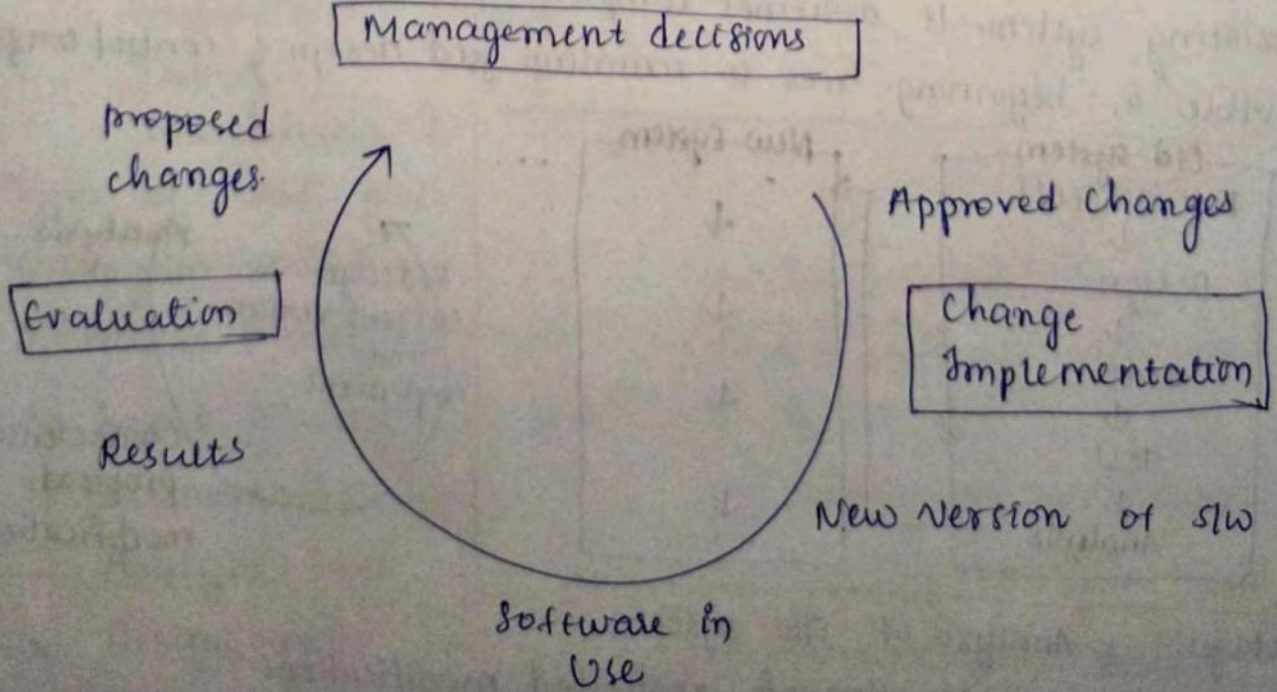
Re-Use repository plays an imp. role.

1. Indentify parts of old system that are candidates for reuse
2. Understand these system parts
3. Modification of old system parts appropriate to new requirments.
4. Integrate modified parts into new system.

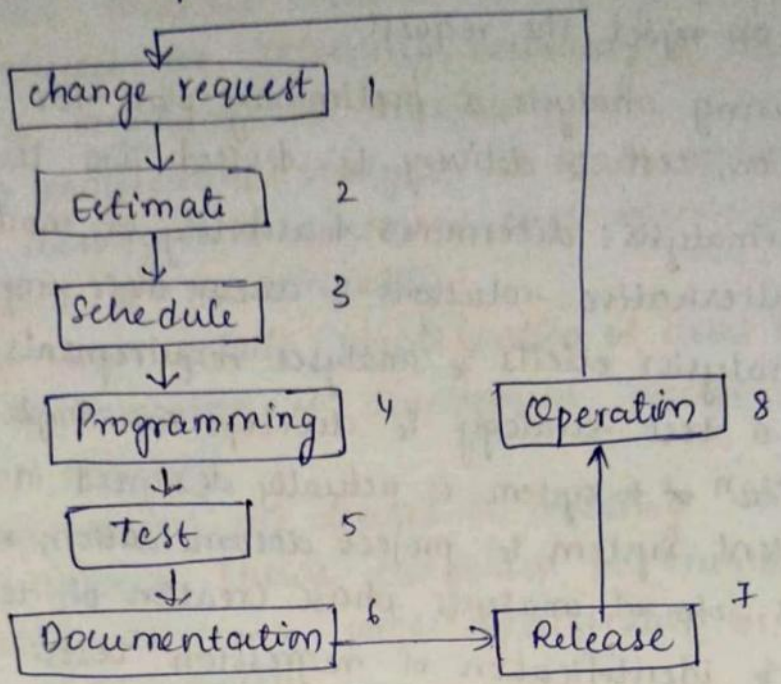


Boehm's Model: Boehm proposed a model for maintenance process based upon economic model & principles.

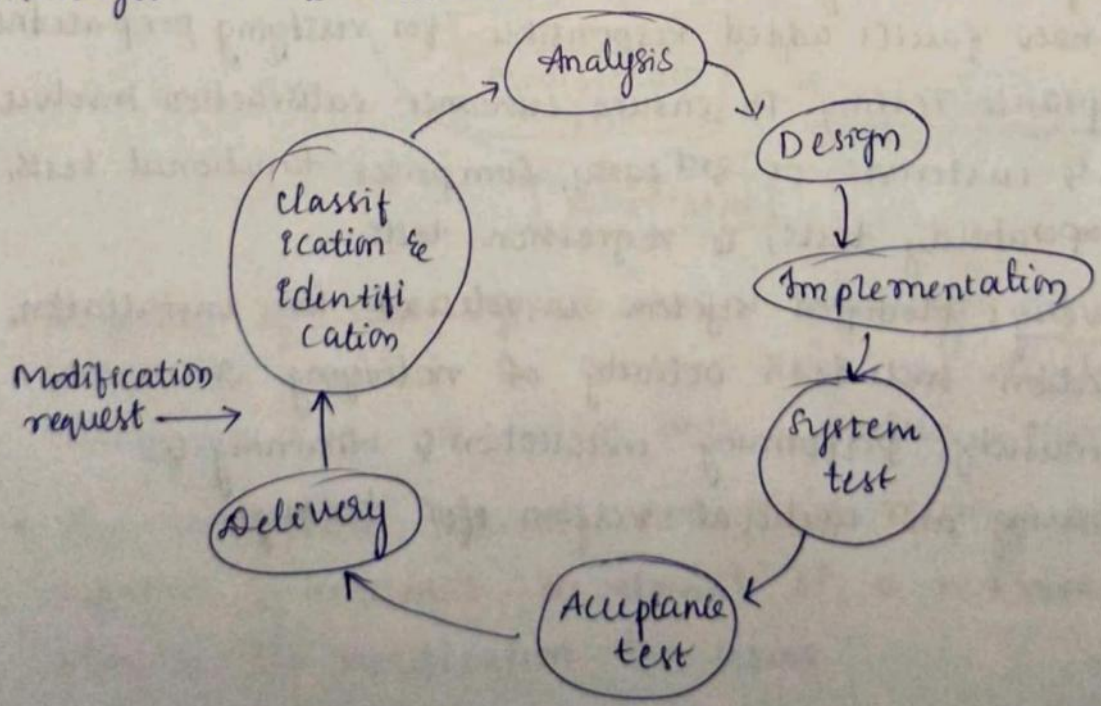
Represents maintenance process ~~models~~ as closed loop & cycles where in changes are suggested & approved first & then are executed



Taute Maintenance: Typical maintenance model that consists of 8 phases in cycle fashion. Begins by requesting change & ends with its operation.



IEEE 1219 model: The IEEE standard organises the maintenance process in 7 phases. Initially modification requests are generated by user, customer, programmer or manages of the maintenance team



Classification & Identification: There r sevrl requsts in single project. Each modification request has a unique identification no. & category of modification. This also includes to determine if to accept or reject the request.

Analysis: During analysis, a preliminary plan for design, implementation, test & delivery is devised. Two levels of analysis.

Feasibility Analysis: determines feasibility of modification & identifies alternative solutions & assess their impacts & costs.

Detailed Analysis: elicits & analyses requirements for modificn. It devises a test strategy & develops an implementation plan.

Design: Modificn of to system is actually designed in this phase. It uses current system & project documentation, existing slw & databases, o/p of analysis phase. Creation of test cases for new design & identification of regression tests.

Implementation: Includes coding & unit testing, integratn of modified code, regression testing, risk analysis & review.

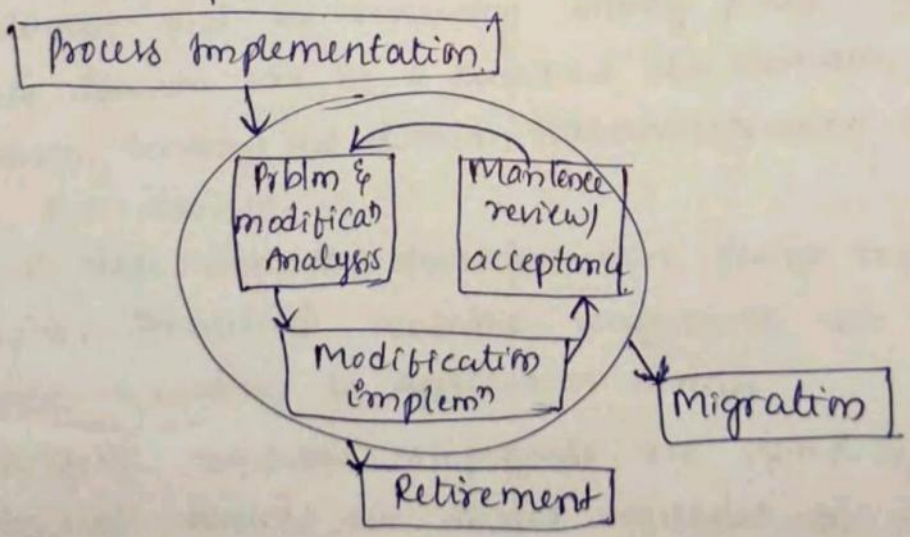
System/Regression Testing: Performed to ensure proper functioning of system along with modifications. It assures thes are no new faults added. Responsible for verifying preparedness.

Acceptance Testing: To ensure customer satisfaction. Involves, users, customers, or 3rd party. Comprises functional tests, interoperability tests, & regression tests.

Delivery: Modified system is released for installation & operation. Includes activity of notifying user community, performing installation & training & preparing an archival version of for backup.

ISO-12207 model: ISO-12207 standard organises maintenance process in 6 phases.

- Implementation includes tasks for developing plans & procedures, creating produces for receiving, recording & tracking & establishing organizational interface
- In problem & modification analysis, the main concern is to analyze request, to classify it & determine scope interms of size, costs & time required & its criticality.
- Modification impln entails identification of items needed to be modified & invocation of development process to actually implement changes.
- Maintenance review are devoted to assessing integrity of modified system & ensure completion of maintenance request. Uses quality assurance process, verification & validation & joint review process.



- Migration of slw system means ~~involving~~ moving it frome one evirnmt to another. Plans are developed to reengineer the old system onto a modern platform.
- The retirement activity consists of retiring a slw system & requires develpmnt of a retirement plan & its notification to users.

Osborne's Model: Is concerned with reality of maintenance environment. Assumes that technique problems arise due to poore management communication & control.

→ Maintenance requirements need to be included in change specification, quality assurance program is required to include in assurance requirements & metric needs to be developed in order to verify that maintenance goals have been met.

Date / / Page

Software Maintenance is the process of modifying a software product after it ha been delivered to the customer.

purpose :- is to modify and update software application after delivery to correct faults and to improve performa

Need for M... :- SM must be perform in order to :

- correct faults
- Improve design
- Implement enhancements
- Interface with other systems.
- Migrate legacy software
- Retire software.
- Accomodate

Categories

1] corrective maintenance
2] Adaptive maintenance
3] perfective maintenance
4] preventive maintenance

1] → corrective · M of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of system.

2] This includes modifications and updations when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software..

3] A software product needs maintenance to support new features that users want or to change different types of functionalities of system acc. to customer demands.

4] This type of maintenance includes MO and up to prevent future problems of softwde.

software configuration management

Page

whenever a software is build, there is always scope for improvement. changes may be required to modify or update any existing solution & to create a new solution for a problem. changes need to be analyzed before they are made to existing system, recorded before they are implemented, reported to have details of before and after. are controlled in a manner that will improve quality and reduce error. This is where we need of SCM comes.

Processes involved in SCM

1. Identification and Establishment.
2. version control
3. change control
4. configuring auditing
5. Reporting.

Cost estimation in Software Engineering

1. Analysis effort method.
2. Parametric Estimating
3. ITK method, also known as Method CETIN.
4. Planning Game
5. Putnam-model also known as SLIM
6. Proxy base estimating

→ PRICE systems.
→ UCP (The use case points method).

Cost estimation in Software engineering is typically concerned with financial spend on the effort to develop and test the software. This can also include requirements reviews, maintenance, training, managing and buying extra equipment, servers and software. Many methods have been developed for estimating software cost for a given project.

Software Reuse

Introduction:

The development of faster (in a small cycle time), better (a good quality product), and cheaper (with small budget) software is a significant challenge for the software industry. A possible way to produce software products in this way is development with software reuse.

Software Reuse:

Software Reuse is the process of producing software systems from existing software systems rather than building software from scratch.

What can be reused ?

Reuse-based software engineering allows producing reusable components only once and saves development effort multiple times. Reusable components are self contained entities which may be any software document or work product generated during the software development process.

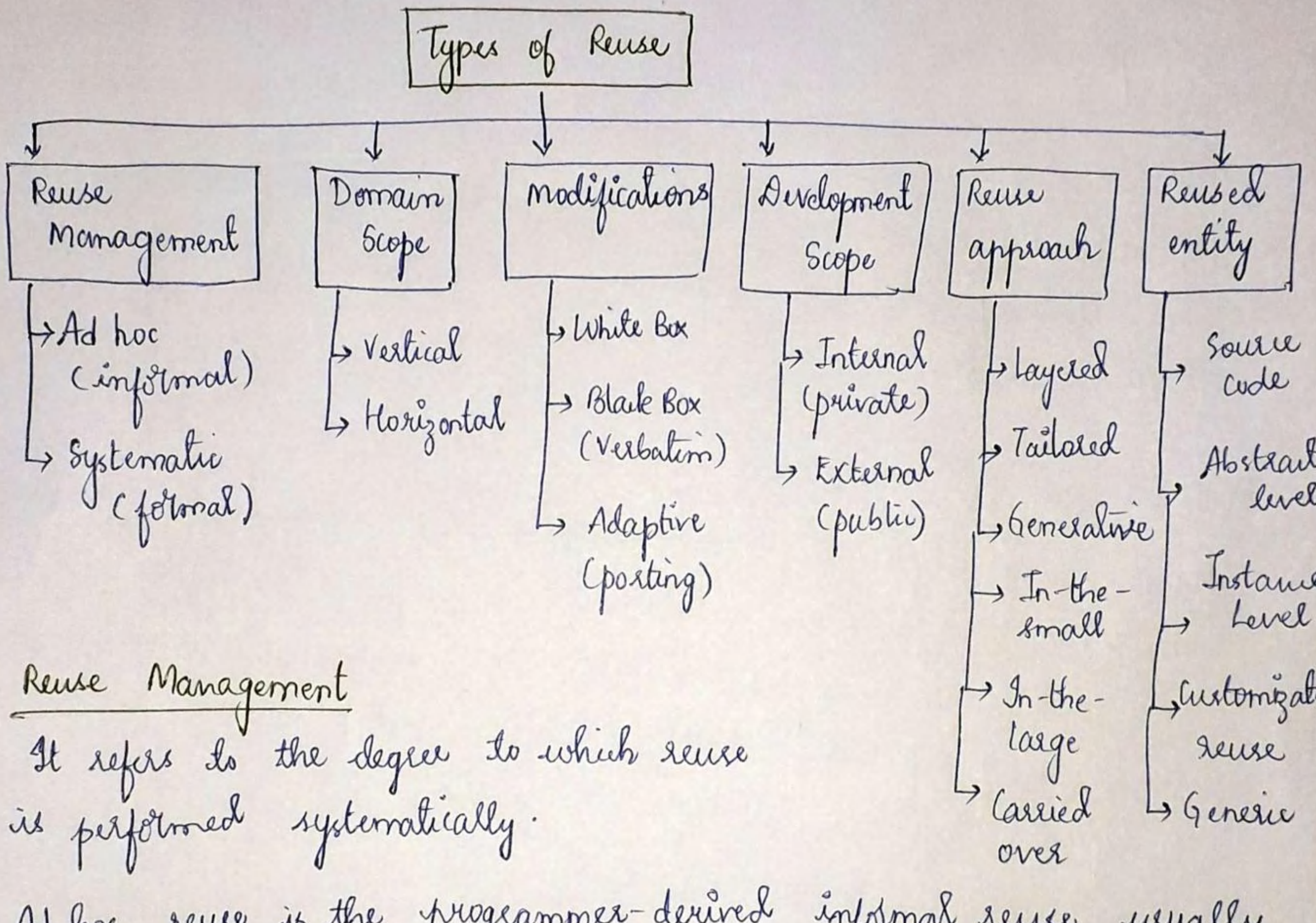
Reusable components are as follows:

- Requirement analysis document (software requirements, subsystems)
- Design models (architectural design, component-level design, user interface design, database design)
- Software architectures (frameworks, component interface, platform configuration)
- Programs (classes, function, exceptions, packages, constants)

- Design patterns
- Data
- Documentation
- Test cases & test plans, expertise
- Algorithms
- Data models
- Domain descriptions
- Semantics, proofs & logic
- Information products (texts, graphics, hypermedia)

Types of Reuse:

The success of reuse depends on the type of reuse strategies implemented in the organisation.



1. Reuse Management

It refers to the degree to which reuse is performed systematically.

→ Ad hoc reuse is the programmer-derived informal reuse, usually ill-documented code used by particular programmers from their own libraries.

→ Systematic reuse is a process driven activity that requires common standards, procedures, and practices applied consistently across a given domain.

2. Domain Scope: refers to the occurence of reuse within a family of systems or between families of systems.

→ Horizontal reuse involves those software components that are used for a wide variety of applications such as email packages; word processing program; huge library of component such as stack class, GUI functions, character handling functions, etc.

→ Vertical Reuse refers to the functional areas or domains that may be utilized by a family of systems with similar functionality.

3. Reuse Modification refers to the extent to which reusable components are changed within the reuse repository of system development.

→ Black Box reuse is the reuse of reusable components without any modification such as standardization, wrapping techniques, etc.

→ White Box reuse is the reuse of components by modification and adaption such as copy & paste, inheritance, etc.

→ Adaptive reuse strategy uses large software structures as invariants and restricts variability to low-level, isolated locations.

4. Development Scope means the reusable components are available either from a ext source external or internal to a project.

→ Internal reuse level is the ratio b/w the number of low level items and the total number of lower-level items.

→ External reuse level is the ratio between the number of lower-level items from an external repository that are used in a higher-level item and the total number of lower-level items that are used in higher-level item.

5. Reuse Approach: refers to different technical methods for implementing reuse.

→ Layered reuse is used to describe the case where reusable functions or operations are viewed as abstract primitives.

→ In tailored reuse approach, the configuration of reusable components is required to interoperate properly with the client software.

→ Generative reuse occurs when the reusable components are used as a generator program rather than being incorporated directly into the final application system.

→ Reuse in-the-large is the use of large, self-contained packages such as spreadsheets and operating systems.

→ Reuse in-the-small is the reuse of components that are dependent upon the environment of the application for full functionality.

→ Carryover reuse occurs when one version of a software component is considered to be used as it is in a subsequent version of the same system.

6. Reused entity is the type of reused objects used for application development.

→ Source Code reuse is the low level modification of an existing source to change its performance characteristics.

→ Abstract reuse is the use of high level abstractions within an object oriented inheritance structure as a foundation for new ideas & additional classification schemes.

→ Instance-level reuse can be defined as simply creating an instance of an existing class.

→ Customization reuse is the use of object-oriented inheritance to support incremental development.

→ Generic reuse is reuse of generic packages such as templates for packages or subprograms.

Reuse Benefits:

Adopting reuse-based software development process attracts a number of well-recognized economic and psychological benefits to, both, end users and developers.
These include the following:

- Savings in development costs and time
- Improvement in productivity of the organization
- Increase in reliability
- Increase in quality
- Increase in the ease-of maintainance
- Improvement in documentation and testing
- High-speed and low-cost replacement of ageing systems
- Improvement in the predictability of the process.

Why almost no REUSE so far?

There are no fully fledged reuse industry so far to develop software through reuse. Some common problems observed in establishing reuse as follows:

- Lack of mature software process
- Needs organizational changes & team building
- Lack of library support for diff kind of problems such as mathematical, scientific etc.
- Needs huge initial investment & efforts in institutionalization of reuse program
- Creating reusable components is diff & time taking process
- Designing a standard & customized component is very difficult
- Developers hesitate to have faith on reusing program designed by others
- Difficulty in recruiting skilled & experience programmers
- Difficulty in finding appropiate reusable component & integrating them in new software development which is a time consuming job & can leads to late delivery & over budget.

Basic Issues in Reuse Program

Imp issues in establishing reuse program are as follows:

Institutionalization of reuse program: The institutionalizing of reuse in an organization for product development focuses

on using reusable assets. It helps to produce quality software product rapidly & effectively. It needs huge investment. Therefore reuse program is established in an incremental manner. & it requires systematic approach for software development.

Technology transfer: Traditional programming languages (C, cobol etc) do not facilitate reuse program. Reuse base software development promote the use of component based technologies. Object (c++, java) & component based (CORBA, .NET, COM etc) technologies is widely used in software reuse.

Component development & customization: It is the process of creating reusable components or assets in a customized manner. It includes activities such as inventory & domain analysis of existing applications & assets, market & technology trends, component construction. Components are implemented using templates, inheritance, variability, scripts etc.

Effective reuse repository: The reusable assets are managed in a centralized reuse repository. Reusable assets are managed in order to easily locate reusable assets, enhance searching power & manage version ctrl. These components are classified & managed into the repository by the reuse manager. Designing effective a reuse repository is a challenging issue in reuse program.

Software development with reuse: It mainly focuses on component integration with software development. It requires skilled programmers who can develop software with reuse facilities.

Searching & retrieval: The effective design of reuse repository reduces the searching & retrieval effort. Signature matching process is effectively used in searching the components.

Process focus: A reuse process makes explicit guidelines that will ensure successful reuse. Its process seperates the component developers & reusers. Therefore the systematic reuse needs to be be adopted rather than the ad-hoc process.

Reuse & coreporate strategy: There should be standard guidelines to reuse, certify, packaging & application development reusable components. The uniform polivies & guidelines required for both reusers & creators.

Organizational issues: Reuse program requires organizational changes in reuse based development. The reuse manager controls & coordinates activities of creators, reusers, & support manager. It is imp to have executive support in a top-down manner while implementing assets bottom-up

A REUSE APPROACH

A typical reuse approach focuses on software development with reuse by reusing the already developed components.

It involves ,

- Domain Engineering
- Reuse Repository
- Component Searching and Retrieval
- Refinement and Integration
- Reuse Measurement

Domain Engineering

- Domain Engineering identifies reusable artefacts and operations in a set of related systems having similar functionalities in specific functional areas, along with the roles and responsibilities.
- The reusable components under domain analysis can be identified by the domain concepts, techniques, technologies, terminologies, and by reviewing among peers.
- The systematic organization of reusable components within the reuse repository system can reflect in the development activity.
- There have been several reuse classification and retrieval approaches designed depending upon the developer’s requirements, available technologies, and reuse software systems.

Reuse Repository

- A software repository contains a large number of reusable components over a wide range of application domains for software

reuse.

- A reuse repository system is a kind of software library where reusable components are managed systematically for easy access for future development.
- The connectivity among reusable components and their retrieval for

reuse depends on the effectiveness of repository.

- One of the simplest repository are software libraries like C++, Java, Ada, etc. The identification and classification component is responsible for the generation of reusable components using domain analysis, classification of reusable components, and defining interfaces.

Component Searching and Retrieval

- The searching process allows developers to put queries in natural languages, including keywords. The searching and matching of reusable

  artefacts according to the application domain make the retrieval process faster.

- A major part of this is **Signature Matching. Signature Matching** is the process of checking the compatibility of two components in terms of their signatures. It defines interfaces of the components.
- Signature may include the component name, number and order of arguments, type of arguments, return types, etc.
- The most popular are the simple keyword and string match, faceted classification and retrieval, signature matching, and behaviour matching.

Refinement and Integration

- Component retrieval from reuse repository is not enough for development with reuse. Instead developers emphasize finding the relevance and applicability of the delivered components with the working context of software development.
- Refinement of reusable components deals with the relevance and applicability of the retrieved components with the working context.

  Refinement of the retrieved components is needed for specialisation, restriction, extension, optimization, implementation, and convenience.
- The selection of right kind of mechanism for each component helps to build a small and coherent set of appropriately generic components.
- The integration of reusable components highly reflects from the type of interfaces and their signatures.

Reuse Measurement

- Reuse metrics measure the performance of reusable things and identify highly reusable components and the specific business area for providing greater benefits to the organization.
- Reuse metric is a quantitative measurement of an attribute of a reusable component.
- Component level measures include component size, component quality, cost of customizing components, difficulties in integrating components, component complexity, and cohesion.
- It is also reported that reuse can be measured in terms of reuse level and size, cost benefit analysis, and reuse investment analysis.

REUSE AT ORGANIZATION LEVEL

- The traditional software development process does not support reuse. In order to promote software reuse, the software development

process must evolve to include reuse activities.

- Organizations perform domain engineering to produce a variety of software products in similar domain. This way is known as Product Line Approach.
- There are following issues in order to perform reuse at organization level:

1)Inventory analysis is performed to find the resuable components in the repository or other sources and assess their scope for reuse.

2)Effective organizational structure for maintaining the product line.

3)Process for creating components of the product line from the core assets as per customer specific requirements.