Chapter 10

SOFTWARE EVOLUTION & MAINTENANCE

CHAPTER OUTLINE

After studying this chapter, the reader will be able to understand the

- Software Evolution
- Program Evolution Dynamics
- Software Maintenance
- Conducting Maintenance
- Types of Maintenance
- The Cost of Maintenance
- Maintenance Effort Distribution
- Maintenance Prediction
- Software Re-engineering
- Configuration Management

SOFTWARE EVOLUTION

Software development process does not end when systems is delivered but continue through the lifetime of system. After a system has been deployed, it inevitably has to change if it is to remain useful. Similarly, business changes and changes to user exceptions generate new requirements for the existing software, to adopt these requirements evolution is required. Software evolution is referred to as the process of developing, maintaining and updating software for various reasons. Software changes are inevitable because there are many factors that change during the life cycle of a piece of software. Some of these factors include:

- Requirement changes
- Environment changes
- Errors or security breaches
- New equipment added or removed
- Improvements to the system

Software evolution is important because organizations have invested large amount money in their software and are now completely dependent on these systems.

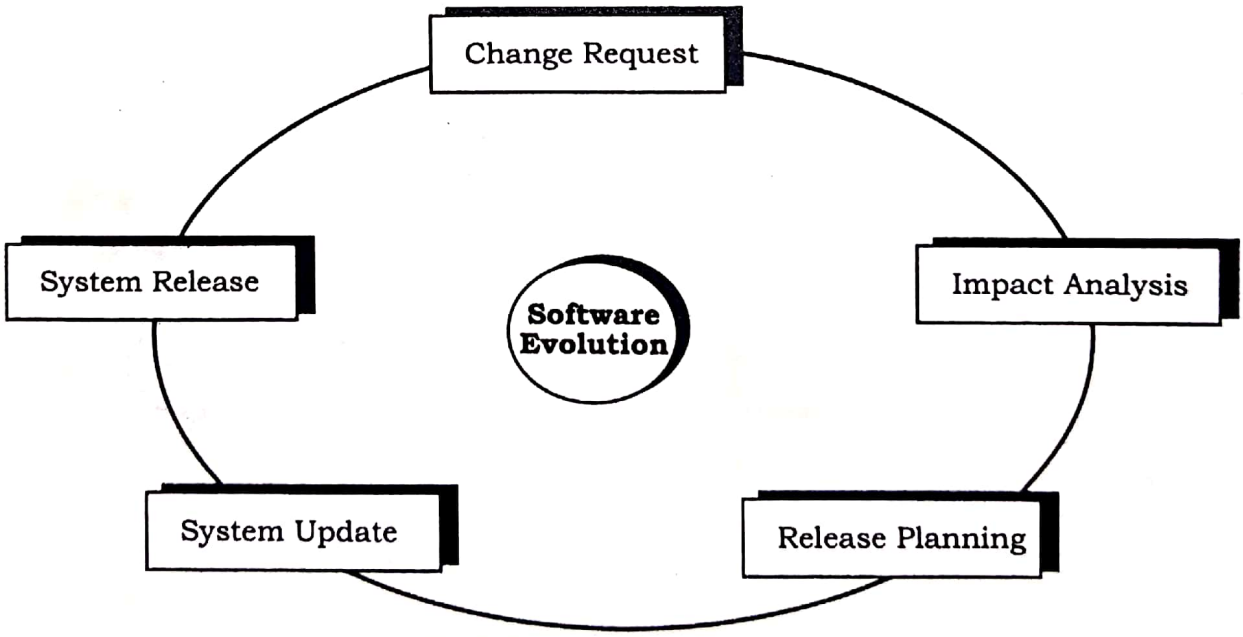

Fig: Software Evolution Process

Software evolution processes vary depending on type of software being maintained, the development process used in an organization and the skills of the people involved. However evolution process includes the fundamental activities of change analysis, release planning, change implementation and release a system to customer.

Evolution process starts from the change request process. After which the cost and impact of these changes are assessed to see how much of the system is affected by the change and how much it might cost to implement the change. If the proposed changes are accepted a new release of the system is planned. A decision is then made on which changes to implement in the next version of the system. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.

Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

PROGRAM EVOLUTION DYNAMICS

Program evolution dynamics is the study of system change. In the 1970s and 1980s, Lehman and Belady, carried out several empirical studies of system change with a view to understanding more about characteristics of software evolution. These are shown in the following table.

Law	Description
1: Continuing change	A program that is used in a real-world environment must necessarily change or else become progressively less useful in that environment.
2: 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.
3: Large program evolution	Program evolution is a self-regulating process. System attributes such as size; time between releases; the number of reported errors is approximately invariant for each system release.
4: Organizational stability	Over a program's lifetime its rate of development is approximately constant and independent of the resources devoted to system development.
5: Conservation of familiarity	Over the lifetime of a system the incremental change in each release is approximately constant.
6: Continuing growth	The functionality offered by systems has to continually increase to maintain user satisfaction.
7: Declining quality	The quality of systems will decline unless they are modified to reflect changes in their operational environment.
8: Feedback system	Evolution processes incorporate multi-agent and multi-loop feedback systems. You have to treat them as feedback systems to achieve significant product improvement.

SOFTWARE MAINTENANCE

Once software system is put into use or installed, new requirements emerges and existing requirements change as the business running that system changes and the system is essentially in the maintenance phase. Software maintenance is an activity which includes optimization, error correction, and deletion of discarded features and enhancement of existing features. Since these changes are necessary, a mechanism must be created for estimation, controlling and making modifications. System maintenance is the general process of changing a system after it has been delivered. The changes may be Simple changes to correct coding errors, more extensive changes to correct design errors or significant enhancement to correct specification errors or accommodate new requirements. It is mostly used for changing custom software and changes are implemented by modifying existing components and adding new components to the system.

There are four major activities occur within maintenance:

- Obtaining maintenance requests
- Transforming requests into changes
- Designing changes
- Implementing changes

1. **Obtaining Maintenance Requests:** In this step a formal process be established whereby users can submit system change requests. When developing the procedures for obtaining maintenance requests, organizations must also specify an individual within the organization to collect these requests and manage their dispersal to maintenance personnel.
2. **Transforming Request into Changes:** Once a request is received, analysis must be performed to identify the of the scope of the request. It must be determined how the request will affect the current system and how long such a project will take. As with the initial development of a system, the size of a maintenance request can be analyzed for risk and feasibility.
3. **Designing changes:** Next, a change request can be transformed into a formal design change, which can then be fed into the maintenance implementation phase.
4. **Implementing Changes:** In this activity once the change design is approved, proposed changes are implemented in respective components of the system.

CONDUCTING MAINTENANCE

A significant portion of the expenditures for information systems within organizations does not go to the development of new systems but to the maintenance of existing systems. We will describe various types of maintenance, factors influencing the complexity and cost of maintenance, and alternatives for managing maintenance.

Maintenance processes vary considerably depending on the type of system being maintained, the development process used in the organization and the people involved in the process. However, the maintenance process is triggered by a set of change requests from system users, management or customers. The cost and impact of these changes are assed to see how much f the system is affected by the change and how much it might costs to implement the change. If the proposed changes are accepted, new release of the system is planned. During release planning, all the proposed changes are considered. A decision is then made on which changes to implement in the next version of the system. The changes are implemented and validated.

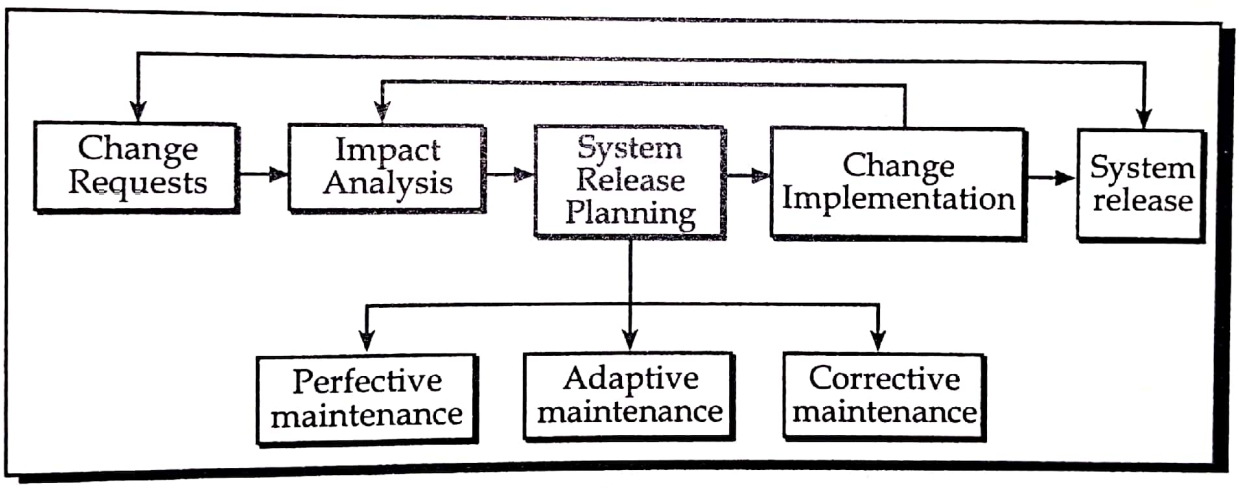

Fig: Software Maintenance Process

TYPES OF MAINTENANCE

We can perform several types of **maintenance** on an information system. By maintenance, we mean the fixing or enhancing of an information system.

1. **Corrective maintenance:** Corrective maintenance deals with the repair of faults or defects found in day-today system functions. It refers to changes made to repair defects in the design, coding, or implementation of the system. This type of maintenance implies removing errors in a program, which might have crept in the system due to faulty design or wrong assumptions. Thus, in corrective maintenance, processing or performance failures are repaired.

For example, if you had recently purchased a new home, corrective maintenance would involve repairs made to things that had never worked as designed, such as a faulty electrical outlet or a misaligned door. Most corrective maintenance problems surface soon after installation. When corrective maintenance problems surface, they are typically urgent and need to be resolved to curtail possible interruptions in normal business activities.

2. **Adaptive maintenance:** Adaptive maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Adaptive maintenance consists of adapting software to changes in the environment such as the hardware or the operating system. In adaptive maintenance, program functions are changed to enable the information system to satisfy the information needs of the user. It involves making changes to an information system to evolve its functionality to changing business needs or to migrate it to a different operating environment. Within a home, adaptive maintenance might be adding storm windows to improve the cooling performance of an air conditioner. Adaptive maintenance is usually less urgent than corrective maintenance because business and technical changes typically occur over some period of time. Contrary to corrective maintenance, adaptive maintenance is generally a small part of an organization's maintenance effort, but it adds value to the organization.

   This type of maintenance may become necessary because of organizational changes which may include:

   - Change in the organizational procedures,
   - Change in organizational objectives, goals, policies, etc.
   - Change in forms,
   - Change in information needs of managers.
   - Change in system controls and security needs, etc.

3. **Perfective maintenance:** Perfective maintenance involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the users' changing needs. It is for adding new programs or modifying the existing programs to enhance the performance of the information system. This type of maintenance undertaken to respond to user's additional needs which may be due to the changes within or outside of the organization. Outside changes are primarily environmental changes, which may in the absence of system maintenance; render the information system ineffective and inefficient. These environmental changes include:

   - Changes in governmental policies, laws, etc.,
   - Economic and competitive conditions, and
   - New technology.

4. **Preventive maintenance:** Preventive maintenance involves performing activities to prevent the occurrence of errors. It tends to reduce the software complexity thereby improving program understandability and increasing software maintainability. It comprises documentation updating, code optimization, and code restructuring. It refers to changes made to increase the understanding and maintainability of your software in the long run. Preventive changes are focused in decreasing the deterioration of your software in the long run. Restructuring, optimizing code and updating documentation are common preventive changes. Executing preventive changes reduces the amount of unpredictable effects software can have in the long term and helps it become scalable, stable, understandable and maintainable. It **involves** changes made to a system to reduce the chance of future system failure. An example of preventive maintenance might be to increase the number of records that a system can process far beyond what is currently needed or to generalize how a system sends report information to a printer so that the system can easily adapt to changes in printer technology. In our home example, preventive maintenance could be painting the exterior to better protect the home from severe weather conditions. As with adaptive maintenance, both perfective and preventive maintenance are typically a much lower priority than corrective maintenance.

THE COST OF MAINTENANCE

The cost of maintenance represent a large proportion of the budget of most organization that use the software system. For some organizations, as much as 60 to 80 percent of their information systems budget is allocated to maintenance activities. These huge maintenance costs are due to the fact that many organizations have accumulated more and more older legacy systems that require more and more maintenance. More maintenance means more maintenance work for programmers. In addition, about one-third of the costs of establishing and keeping a presence on the Web go to programming maintenance.

Maintenance cost as a proportion cost of development costs vary from one application domain to another. For business application system, maintenance cost were broadly comparable with system development cost. For embedded real time systems, maintenance cost may be up to four times higher than development cost.

So while developing system, good software engineering techniques such as precise specification, use of object oriented development and configuration management are used that contribute to maintenance cost reduction.

Factors Influencing Maintenance Cost

Numerous factors influence the **maintainability** of a system. These factors, or cost elements, determine the extent to which a system has high or low maintain- ability. Of these factors, three are most significant: the number of latent defects, the number of customers, and documentation quality. The others—personnel, tools, and software structure—have noticeable, but less, influence.

- **Latent defects:** This is the number of unknown errors existing in the system after it is installed. Because corrective maintenance accounts for most maintenance activity, the number of latent defects in a system influences most of the costs as- sociated with maintaining a system.
- **Number of customers for a given system:** In general, the greater the number of customers, the greater the maintenance costs. For example, if a system has only one customer, problem and change requests will come from only one source. Also, training, error reporting, and support will be simpler. Maintenance requests are less likely to be contradictory or incompatible.
- **Quality of system documentation:** Without quality documentation, maintenance efforts can increase exponentially. High-quality documentation leads reduction in the sys- tem maintenance effort when compared with average-quality documentation. In other words, quality documentation makes it easier to find code that needs to be changed and to understand how the code needs to be changed. Good documentation also explains why a system does what it does and why alternatives were not feasible, which saves wasted maintenance efforts.
- **Maintenance personnel:** In some organizations, the best programmers are assigned to maintenance. Highly skilled programmers are needed because the maintenance programmer is typically not the original programmer and must quickly understand and carefully change the software.
- **Tools:** Tools that can automatically produce system documentation where none exists can also lower maintenance costs. Also, tools that can automatically generate new code based on system specification changes can dramatically reduce maintenance time and costs.
- **Well-structured programs:** Well-designed system is easier to understand and fix.

Software maintenance is the general process of changing a system after it has been delivered and put into use. The changes made to the software may be simple changes to correct coding errors, more extensive changes to correct design errors or accommodate new requirement.

The term is mostly used for changing customer software and changes are implemented by modifying existing components and adding new component to the system.

MAINTENANCE EFFORT DISTRIBUTION

It is usually cost effective to invest effort in designing and implementing a system to reduce the costs of future changes. Adding new functionality after delivery of system is more expensive because you have to spend time to learn the system and analyzing the impact of the proposed changes. So, good software engineering techniques such as specification methods, design methodology, object oriented development and configuration management tools contribute to maintenance costs. The effort of 21% is spent is corrective maintenance, 25% cost is contributed to the functionality addition i.e. adaptive maintenance, 4% effort is spent in preventive maintenance and 50% of effort is contributed to perfective maintenance.

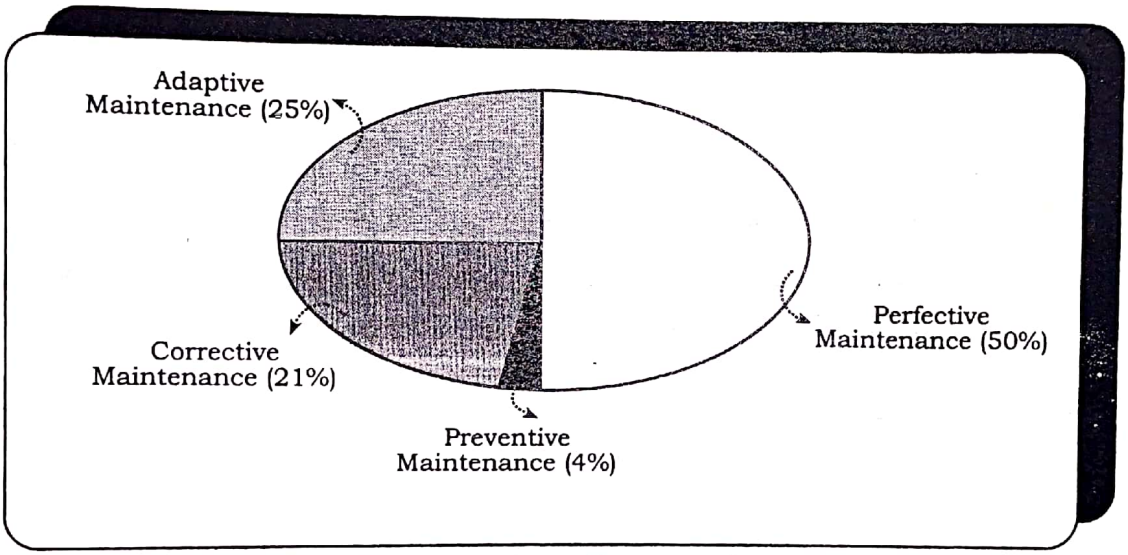

Fig: Maintenance Effort Distribution Pie-chart

Maintenance Cost

The relative cost of maintenance and new development vary from one application domain to another. It is affected by both technical and non-technical factors. It is usually more expensive to add functionality after system is in operation than it is to implement the same functionality during development. The reasons for this are:

1. **Team stability:** After a system has been delivered, it is normal for development to be broken up and for people to work in new project. The new team or the individual responsible for maintenance do not understand the system. They need to spend time to understanding the system before implementing changes to it.

2. **Poor development practice:** The contracts of maintain a system is usually separate from the system development contract. The maintenance contract may be given to a different company rather than original system developer. If the development team can cut corners to save effort during development then this means that software is more difficult to change in the future.

3. **Staff skills:** Maintenance staff is often relatively inexperienced and unfamiliar with the application domain. Furthermore, old systems may be written in obsolete programming language. The maintenance staff may not have much experience of development in these languages and most learn these languages is to maintain the system.

4. **Program age and structure:** As changes are mode to programs their structure tends to degrade, as programs age, they become harder to understand and change.

MAINTENANCE PREDICTION

Since unexpected maintenance costs may lead to an unexpected increase in costs, it is important to predict the effect of modifications in the software system. Software maintenance prediction refers to the study of software maintainability, the modifications in the software system, and the maintenance costs that are required to maintain the software system. It emphasize on predicting what system changes might be proposed what part of the system are likely to be most difficult to maintain. Various predictions are closely related and specify the following.

1. The decision to accept a system change depends on the maintainability of the system components affected by that change up to a certain extent.

2. Implementation of changes results in degradation of system structure as well as reduction in system maintainability.

3. Costs involved in implementing changes depend on the maintainability of the system components.

To predict the number of changes requested for a system, the relationship between the system and its external environment should be properly understood. To know the kind of relationship that exists, organizations should assess the following.

1. Number and the complexity involved in the system interface. More interfaces mean more complexity, which in turn means more demand for change.

2. Number of system (volatile) requirements. Changes required in organizational policies and procedures tend to be more volatile than the requirements based on a particular domain.

3. Number of business processes in which the system operates. More business processes implies more demands for system change.

To predict maintainability of a software system, it is important to consider the relationship among the different components and the complexity involved in them. Generally, software system having complex components is difficult and expensive to maintain. The complexity in a

software system occurs due to the size of procedures and functions, the size and the number of modules, and the nested structures in the software code. On the other hand, a software system developed by using good programming practices reduces not only the complexity but also the effort required in software maintenance. As a result, such software systems minimize the maintenance cost. For maintaining the individual components in software systems, it is essential to identify the complexity measurements of components.

After a system has been put into operation, several process metrics are used to predict the software maintainability. Process metrics, which may be useful for assessing maintainability, are listed below.

1. **Corrective maintenance:** Sometimes, more errors are introduced rather than being repaired during the maintenance process. This shows decline in maintainability.
2. **Average time required for impact analysis:** Before starting the software maintenance process, it is essential to analyze the impact of modifications in the software system. This is known as impact analysis, which reflects the number of components affected by the change.
3. **Number of outstanding change requests:** If the number of outstanding change requests increases with time, it may imply decline in maintainability.
4. **Average time taken to implement a change request:** This involves activities concerned with making changes to the system and its documentation rather than simply assessing the components which are affected. If the time taken to implement a change increases, it may imply a decline in maintainability.

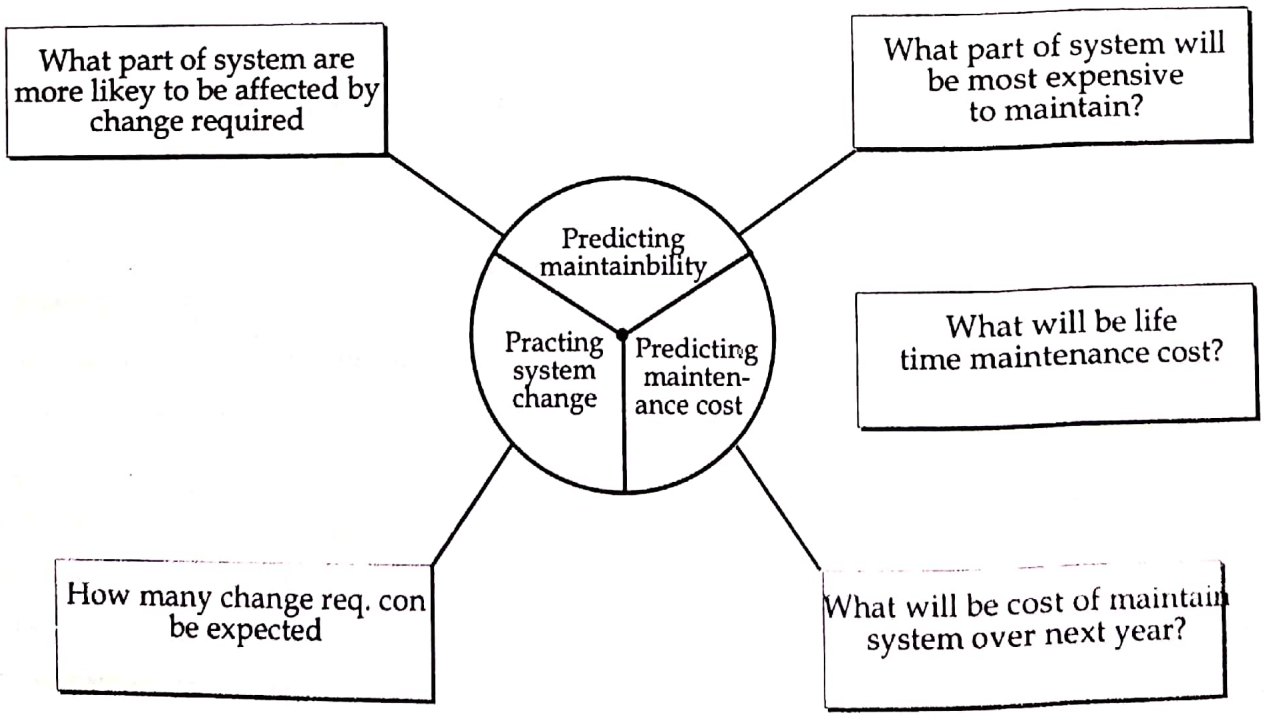

Fig: Maintenance Prediction

LEGACY SYSTEM

Because of time and effort required to develop a complex socio technical system, large computer based system usually have a long life time. It is sometimes too expensive and too risky to discard such business system after few years of use so, their development continues throughout their lifetime with changes to accommodate new requirements, new operating platform and so on. A legacy system is a computer system, programming language, software application, process or other technology that is outdated or that can no longer receive support and maintenance but is essential for organizations or companies and cannot be replaced or updated easily for different reasons.

Legacy systems are socio technical computer based systems that have been developed in past often using obsolete or older technology. They include application software together with business processes, support software, system hardware etc. Legacy systems associated with terminology or processes that are no longer applicable to current contexts or content so they are difficult to change, changes to one part of system inevitably involve changes to other components.

For example: in a bank, banking management system was one of their earliest systems. Organization policies and procedures may rely on this system. If we replace the banking management system there would be a serious business risk. If the replacement system did not work properly. So, they are maintained because it is too risky to replace.

Legacy System Components

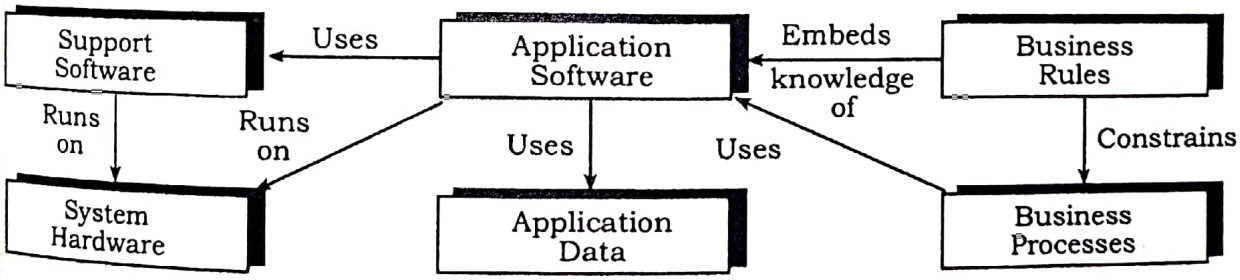

Fig: Legacy System Components

There are many legacy system that are critical business systems and these have to be extended and adapted to changing business practices. The organizations have to decide how to get the best return investment from such legacy systems. This involves making a realistic assessment of their legacy systems and then deciding on the most appropriate strategies for evolving these systems there are four strategic options:

a. **Scrap the system completely:** In this method, when the system is not making an effective contribution to business processes due to change in business process then system will be scrapped.

b. **Reengineer the system to improve its maintainability:** this method is valid when the system quality has been degraded by change and where a new change to the system is still being proposed.

c. **Leave the system unchanged and continues with regular maintenance:** This option is chosen when the system is still required but is fairly stable and the system users make relatively few change requests.

d. **Replace all or Part of the system with a new system:** This option should be chosen when factors, such as new hardware, mean that the old system cannot continue in operation or where off-the shelf systems would allow the new system to be developed at a reasonable cost.

SOFTWARE RE-ENGINEERING

Any software gets out of date over time and, if the organization's performance depends on it, then they can be prevented, but for this it is necessary to initiate some measures which involve software reengineering process. The process of system evolution involves understanding the program that has to be changed and them implementing these changes. However older legacy system are difficult to understand and change or overtime, the initial program structure may have been corrupted by a series of changes. To make a legacy software system easier to maintain, you can re-engineer these system to improve their structure and understandability. A software reengineering is software upgrading procedure or its migration to a more advanced technology platform. At the same time, its current functionality is either saved or undergoes slight modification. Re-engineering concerned with restructuring and re-documenting software to make it more maintainable. Reengineering is the process of examination and alteration of a software system to improve the maintainability and to reconstitute it in a new form. Reengineering may involve re documenting the system, refactoring the system architecture, translating programs to a modern programming language and modifying and updating the structure and values of system's data.

Necessity of Software Reengineering

The reengineering process will be an appropriate option in the following cases:

- When the programming language or platform is no longer supported
- There is a drastic change in technology
- Business processes in the company are changing
- If the software is initially poor
- A technology appears which is perfect for your goals

Advantages of Software Re-engineering

At a certain stage, the organization is faced with the choice of creating a new system from scratch or upgrading an existing one. In most cases, it is software reengineering process that will be the right choice, as it provides a number of significant advantages:

- **Reduced Risk:** There is a high risk in redeveloping business critical system. Errors may be made in the system specification or these may be development problem. Delays in introducing new software may mean that business is lost and extra costs are incurred.
- **Reduced cost:** The cost of reengineering may be significantly less than cost developing new software.
- **Productivity increase:** By optimizing the code and database the speed of work is increased.
- **Improvement opportunity:** You can not only refine the existing product; but also expand its capabilities by adding new features;
- **Time saving:** Instead of starting development from scratch, the existing solution is simply transferred to a new platform, saving all business-logic;
- **Optimization potential:** You can refine the system functionality and increase its flexibility, ensuring better compliance with the enterprise's current objectives;
- **Processes continuity:** The old product can be used while testing the new system until all work is completed.

Software Reengineering Process

Software reengineering process allows modernizing the used system and eliminating technical problems, which reduces the cost of service and expands its capabilities in terms of meeting business needs. The input to reengineering process is a legacy program and the output is an improved and restructured version of the same program.

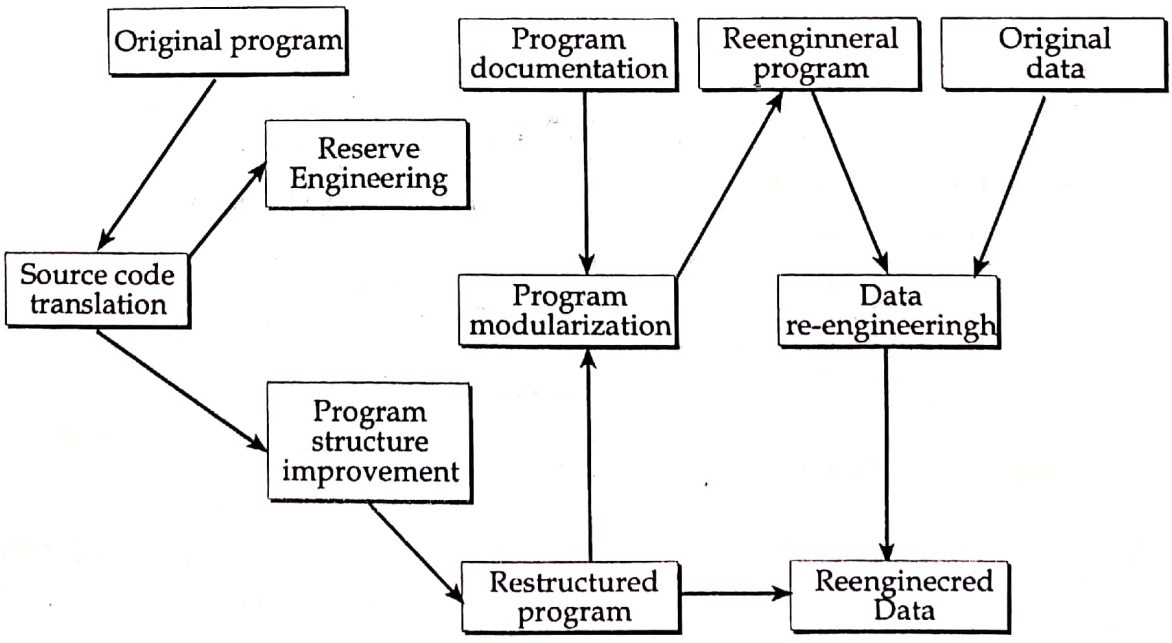

Fig: Software Reengineering Process Activities

The activities in reengineering process are:

1. **Source code translation:** Source code translation is the automatic conversion of a program written in one programming language to another language. It may be necessary when the original programming language is obsolete. Using a translation tool, the program is converted from old programming language to a more modern version of the same language or to a different language e.g FOTRAN to C.
2. **Program structure improvement:** It focuses on design details of individual modules and on local datav structures defined within modules. In this, the control structure of the program is analyzed and modified to make it easier to read and understand.
3. **Program modularization:** In this method, repeated parts of the program are grouped together and redundancy is removed where ever it is appropriate. System that uses several different data stores may be re factored to use a single repository.
4. **Data reengineering:** Data reengineering is the process of changing the data structure organization without changing the data values. It is necessary because of inconsistent data management. It involves analysis and reorganizing the data structure (Sometimes data values) in a program. May be part of the process of migrating from a file based system to a DBMS based system or changing from one DBMS to another. Different approaches to data reengineering are: Data Clean up, Data extension and Data migration.
5. **Reverse engineering:** Reverse engineering is the process of deriving the system design and specification from it source code. The purpose of reverse engineering is to facilitate the maintenance work by improving the understandability of a system and to produce the necessary documents for a legacy system. In this, software is analyzed with a view to understanding its design and specification but may also be used to re-specify a system for re-implementation. This process is usually completely automatic. The design and specification of a system may be reverse engineered so that they can be an input to the requirement specification process for the systems replacement.

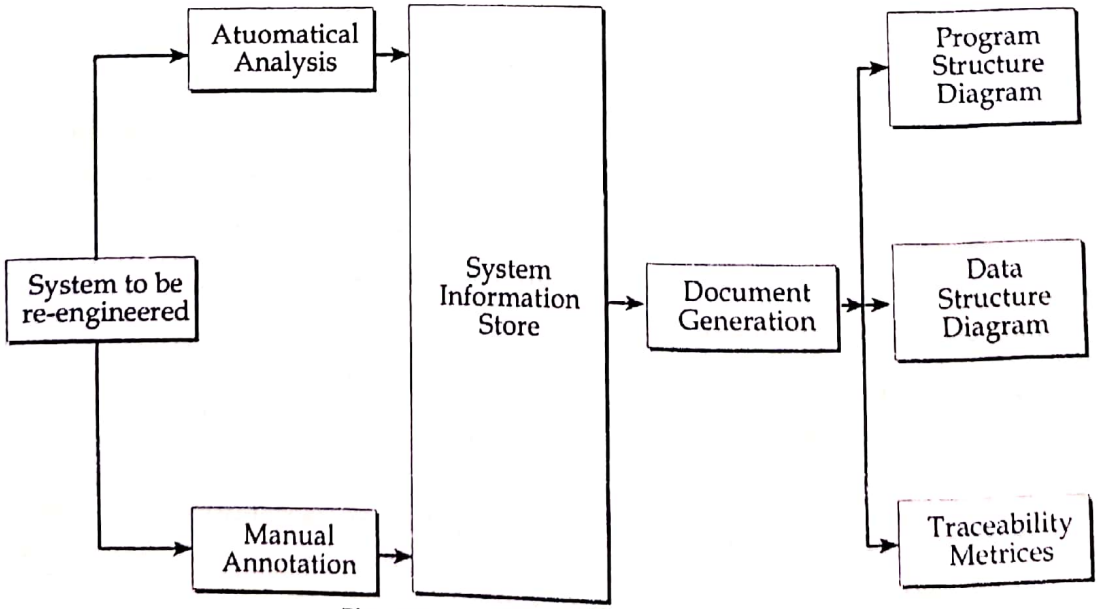

Fig: Reverse Engineering Process

CONFIGURATION MANAGEMENT

When we develop software, the product (software) undergoes many changes in their maintenance phase; we need to handle these changes effectively. Several individuals (programs) works together to achieve these common goals. This individual produces several work product (SC Items) e.g., Intermediate version of modules or test data used during debugging, parts of the final product. The elements that comprise all information produced as a part of the software process are collectively called a software configuration.

Software Configuration Management is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. It is abbreviated as the SCM process in software engineering. The primary goal is to increase productivity with minimal mistakes. It is a software engineering discipline consisting of standard processes and techniques often used by organizations to manage the changes introduced to its software products. It is a process that concerned with the policies, processes and tools for managing changing software systems. Configuration Management helps organizations to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle.

When multiple people are working on software which is consistently updating, it may be a method where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently. It changes in user requirements, and policy, budget, schedules need to be accommodated. SCM helps in identifying individual elements and configurations, tracking changes, and version selection, control, and base lining.

It is practical in controlling and managing the access to various SCIs e.g., by preventing the two members of a team for checking out the same component for modification at the same time. **It provides the tool to ensure that changes are being properly implemented.** It has the capability of describing and storing the various constituent of software. SCM is used in keeping a system in a consistent state by automatically producing derived version upon modification of the same component.

The primary reasons for Implementing Software Configuration Management System are:

- The primary goal of the SCM process is to increase productivity with minimal mistakes
- The main reason behind configuration management process is that there are multiple people working on software which is continually updating. SCM helps establish concurrency, synchronization, and version control.
- There are multiple people working on software which is continually updating

- It may be a case where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, and schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders
- SCM process is also beneficial to control the costs involved in making changes to a system

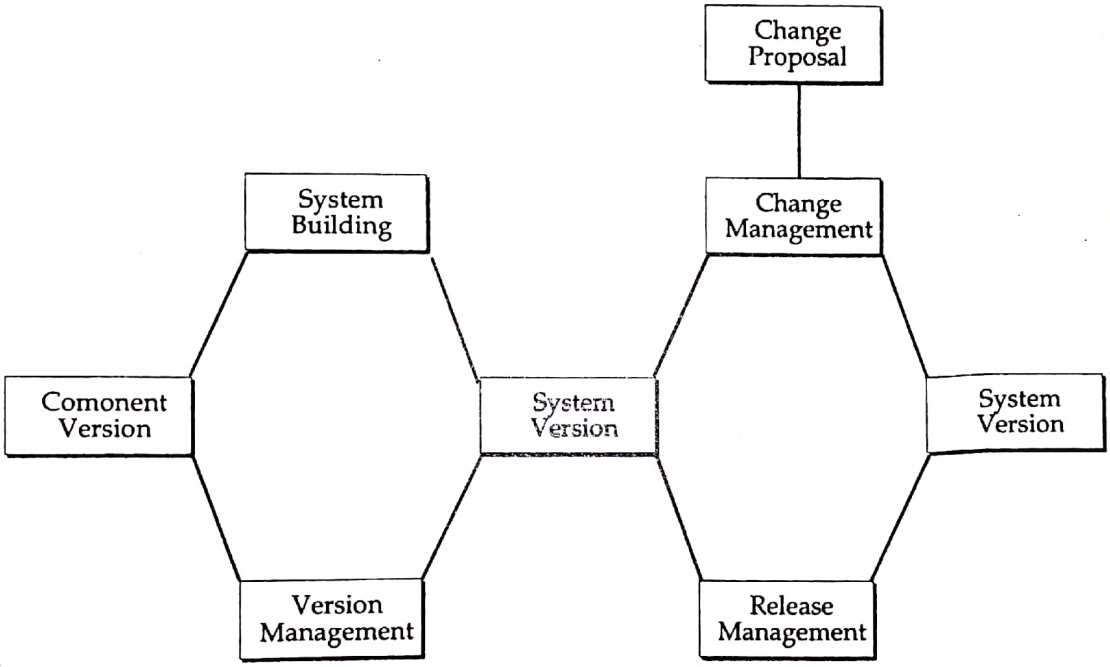

Fig: Software Configuration Management Process

The configuration management of a s/w system product involves four closely related activities.

1. **Change management:** This involves keeping track of request for changes to the s/w from customer and developers working out the costs and impact of making these changes, and deciding if and when the changes should be implemented.
2. **Version management:** This involves keeping track of the multiple version of system components and ensuring that changes made to components by different developers do not interfere with each other.
3. **System Building:** This is the process of assembling program components, data and libraries and then compiling and linking these to create an executable system.
4. **Release Management:** This involves preparing s/w for external release and keeping track of the system version that have been released for customer use.

Change Management

Change is a fact of life for large s/w system. Organizational needs and requirements change during the lifetime of a system. This requires corresponding changes to be made to the s/w. To ensure that the changes are applied to the system in a controlled way, need a set of tools supported and change management process. Change management is a systematic approach to dealing with the transition or transformation of an organization's goals, processes or technologies. Software change management is the process of keeping track of request for changes to the software from customers and developers, working out the costs and impact of making these changes and deciding when the changes should be implanted. The purpose of change management is to implement strategies for effecting change, controlling change and helping people to adapt to change. Such strategies include having a structured procedure for requesting a change, as well as mechanisms for responding to requests and following them up.

The change management process is initiated when a customer completes and submits a change request deserting the change required to the system. Request change by completing a change request form

analyze change request

If change is valid the

Assess how change might be implemented

Assess change cost

Record change request in database.

Submit request to change control board.

If change is accepted then

Repeat:

Make change to s/w

record change and link to associated change req.

Submit change s/w for quality approval.

Until: s/w is adequate.

Create new system

Version

else

reject change request

else

reject change request.

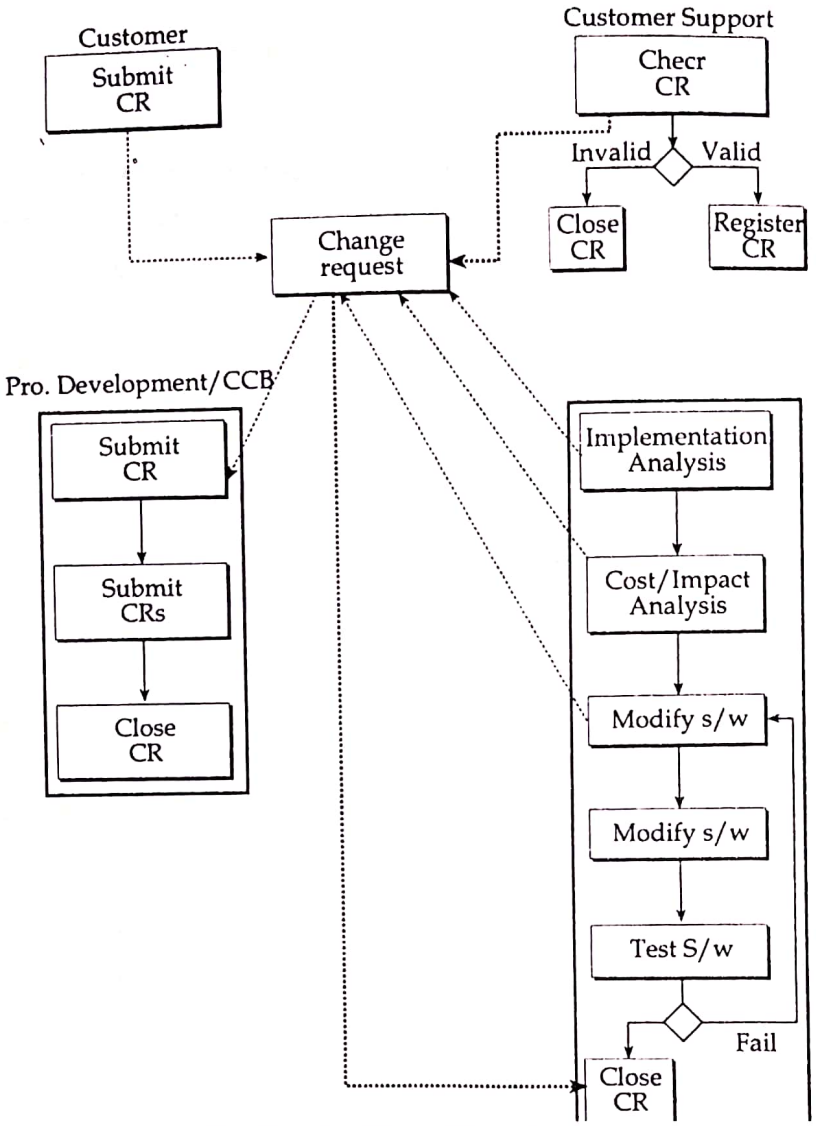

Fig: Change Management Process

Factors that should be taken into account in deducing whether or not a change should e approved one.

- The consequence of not making the change.
- The benefits of the changes.
- The no. of users affected by the change.
- The costs of making the change.
- The product release cycle.

Version Management

Version Management also called Version Control or Revision Control is a means to effectively track and control changes to a collection of related entities. It is the process of keeping track of different version of software components or configuration items and the system in which these

components are used. It also ensure that change made by different developers to these version do not interfere with each other.

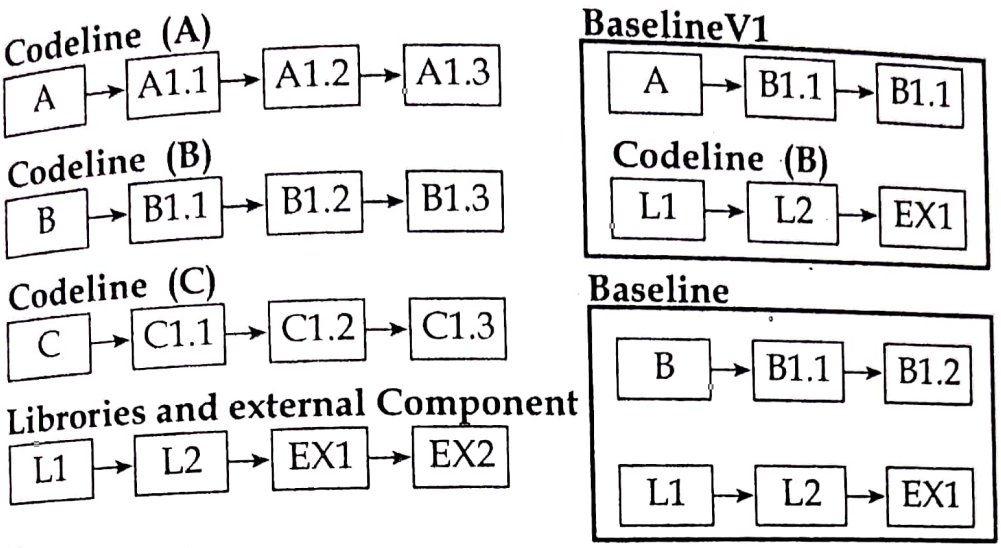

Fig: Change Request Process

Version management systems normally provide a range of features:

1. **Versions and release identification:** Managed versions are assigned identifiers when they are submitted to the system. These identifiers are usually based on the name of configuration item followed by one or more numbers e.g. 1.3 means third version in code line 1.
2. **Strong Management:** To reduce the storage space required by multiple version of components that differ only slightly, VM system usually provide strong management that stores list of differences between versi8on.
3. **Change history recording:** All of the changes made to the code of system or components are recoded and listed.
4. **Independent development:** Different developers may be working on the same content at the same time without interfere.
5. **Project support:** A version management system may support the development of several projects, which share components factors that should be taken into account in deducing whether or not a change should be approved are:
   - The consequence of not making the change.
   - The benefits of the change.
   - The no. of users affected by the change.
   - The costs of making the change.
   - The product release cycle.

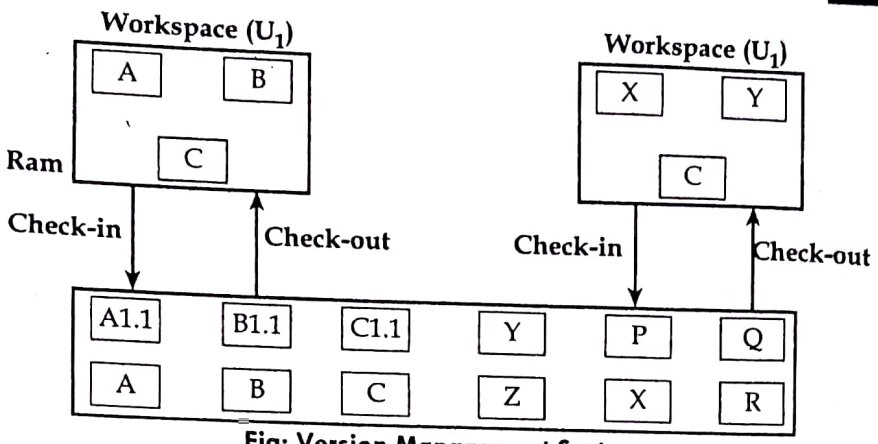

Fig: Version Management System

Release Management

A system release is a version of the system that is distributed to customers. System release includes excitable code data files, configuration files and documentation Release management involves making decision on system release dates, preparing all information for distribution and documenting each system release. Release Management is the process that handles software deployments and change initiatives. Across an organization, it schedules the relevant tasks (internal and external), assigns the physical and human resources needed to carry them out, and oversees the execution. It starts with planning what will be contained within a release, managing the software build through different stages and environments, testing stability and finally, deployment.

A system release is not just the executable code of the system. The release may also include.

- Configuration files defining how the release should be configured for particulars installation.
- Data files that are needed for successful system operations.
- An installation program that is used to help install the system on target hardware.
- Electronic and paper documentation describing the system.
- Packaging and associated publicity that have been designed for that release.

Factors Influencing Release Strategy

- **Technical quality of software:** If serious system faults are reported which affect the way in which many customers use the system, it may be necessary to issue a fault repair release.
- **Platform changes:** New release of software application is created in when a new version of the operating system platform is released.
- **Platform changes:** New release of software application is created in when a new version of the operating system platform is release.

- **Competition:** A new system release may be necessary if a competing product is available.
- **Marketing requirements:** The marketing department of an organization may have made a commitment for release to be available at a particular date.
- **Customer change proposals:** For Customized systems, customers may have made and paid for a specific set of system change proposals, and they expect a system release as soon as these have been implement.

System Building

Software building is the activity of combining the correct versions of software configuration items, using the appropriate configuration data, into an executable program for delivery to a customer or other recipient, such as the testing activity. For systems with hardware or firmware, the executable program is delivered to the system-building activity. Build instructions ensure that the proper build steps are taken in the correct sequence. System building is the process of compiling and linking s/w components into a program that executes on a particular target configuration. It is the process of assembling system components into an executable program to run on some target computer system. Building a system from its components include.

- Have all components that make up a system been included in the buildings instruction?
- Has the appropriate version of each required component been included in the building instruction are all required dates, available
- It the appropriate version of the complier and other required tools available?

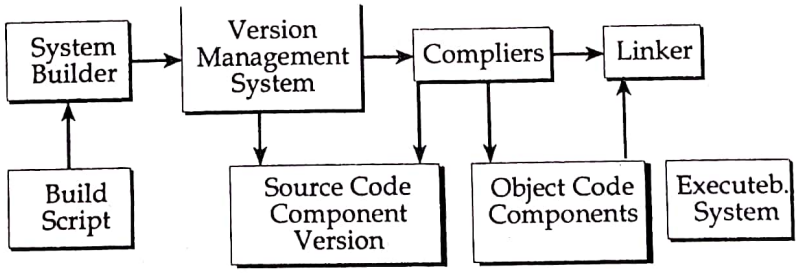

Fig: System Building process

1. What do you mean by software evolution? Explain software evolution process.
2. Define software maintenance. Explain different types of software maintenance.
3. What is software reengineering? Explain different reengineering approaches.
4. What do you mean by software configuration management? Explain configuration management activities in brief.
5. Explain different program evolution dynamics.
6. What is maintenance prediction? Explain the factors that we should assess the relationship between system and its environment?

❑❑❑