Unit-7
Software Maintenance

Software is an evolutionary Entity/Evolving nature of Software

- **Software Evolution** is a term which refers to the process of developing software initially, then timely updating it for various reasons, i.e., to add new features or to remove obsolete functionalities etc. The evolution process includes fundamental activities of change analysis, release planning, system implementation and releasing a system to customers.
- **Software evolution** is the set of activities, both technical and managerial, that ensures that software continues to meet organizational and business objectives in cost effective way.
- It is all programming activity that is intended to generate a new software version from an earlier operational version.
- The application of software maintenance activities and processes that generates a new operational software version with a changed customer experienced functionality or properties from a prior operational version together with the associated quality assurance activities and processes ,and with the management of activities and processes.
- A software product must change continually or become progressively less useful-**Lehman's first law.**
- The structure of program tends to degrade as more and more maintenance is carried out on it.-**Lehman's second law**
- Over program's lifetime, its rate of development is approximately constant.-**Lehman's third law**

Software Change

- Software change is inevitable
  – New requirements emerge when the software is used;
  – The business environment changes;
  – Errors must be repaired;
  – New computers and equipment is added to the system;
  – The performance or reliability of the system may have to be improved.
  • A key problem for all organizations is implementing and managing change to their existing software systems.

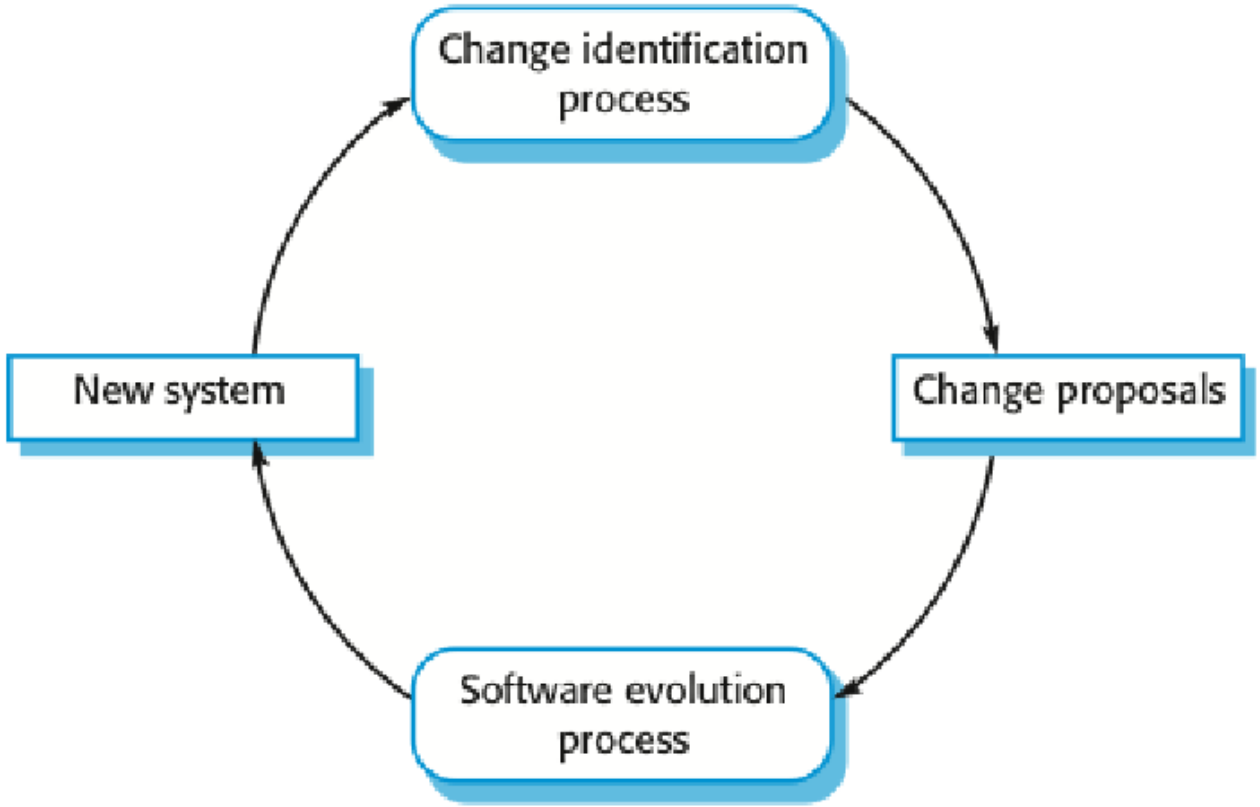
Importance of Software Evolution

- Organizations have huge investments in their software systems - they are critical business assets.
- To maintain the value of these assets to the business, they must be changed and updated.
- The majority of the software budget in large companies is devoted to changing and evolving existing software rather than developing new software.

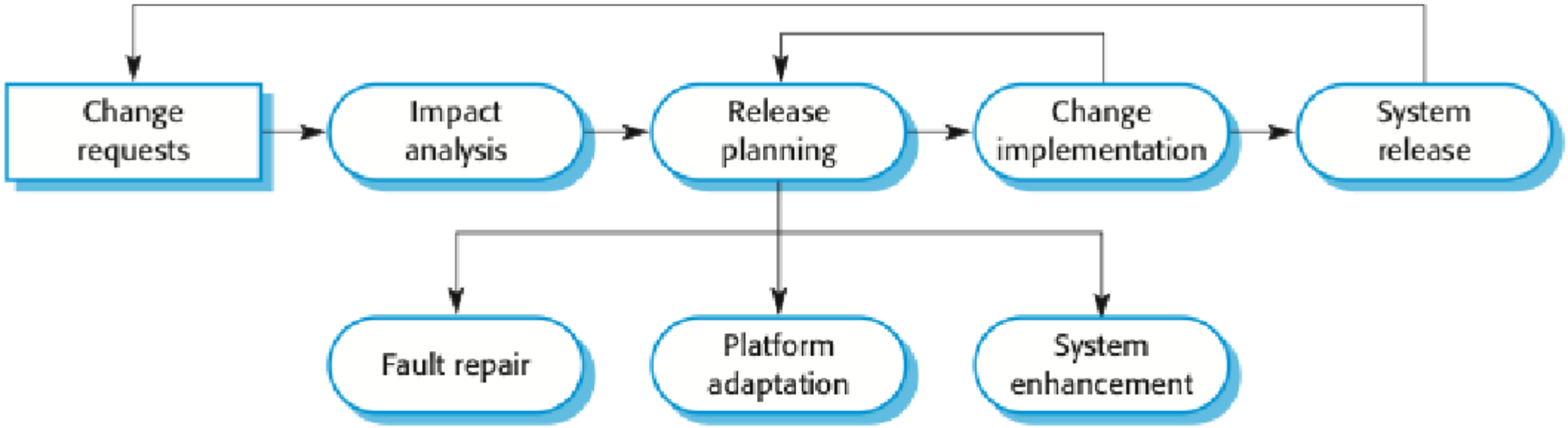
Evolution Process

- Software evolution processes depend on
  - The type of software being maintained;
  - The development processes used;
  - The skills and experience of the people involved.
- Proposals for change are the driver for system evolution.
  - Should be linked with components that are affected by the change, thus allowing the cost and impact of the change to be estimated.
- Change identification and evolution continues throughout the system lifetime.

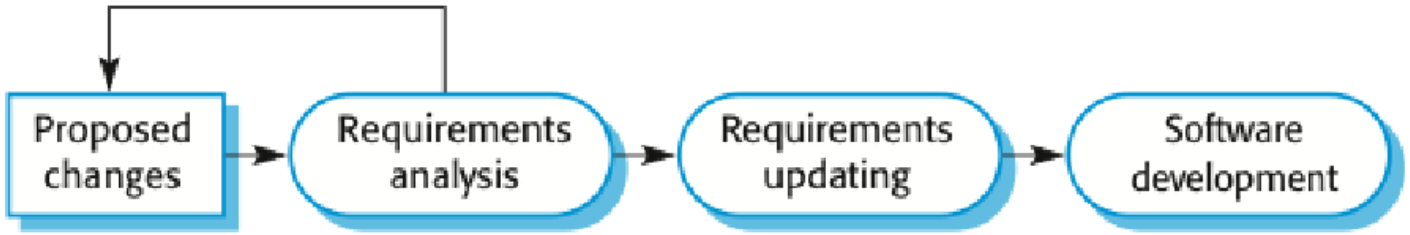
Change identification and evolution processes



The software evolution process



Change Implementation



- Iteration of the development process where there visions to the system are designed, implemented and tested.
- A critical difference is that the first stage of change implementation may involve program understanding, especially if the original system developers are not responsible for the change implementation.
- During the program understanding phase, you have to understand how the program is structured, how it delivers functionality and how the proposed change might affect the program.

Urgent Change Request

Urgent changes may have to be implemented without going through all stages of the software engineering process
– If a serious system fault has to be repaired to allow normal operation to continue;
– If changes to the system's environment (e.g. an OS upgrade) have unexpected effects;
– If there are business changes that require a very rapid response (e.g. the release of a competing product).

Software Maintenance

- Software Maintenance is the process of modifying a software product after it has been delivered to the customer.
- The main purpose of software maintenance is to modify and update software application after delivery to correct faults and to improve performance.
- Changes made to a system to fix or enhance its functionality. Software Maintenance is a very broad activity that includes error corrections, enhancements of capabilities, deletion of obsolete capabilities, and optimization.
- The deliverables and outcomes from the process are the development of a new version of the software and new versions of all design documents created or modified during the maintenance effort.
- Software modifications or maintenance is required to address the following reasons:
  - **Market Conditions** - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification.
  - **Client Requirements** - Over the time, customer may ask for new features or functions in the software.
  - **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.
  - **Organization Changes** - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.
- **Why Maintenance is Needed?**
- Software Maintenance must be performed in order to:
  - Correct faults.
  - Improve the design.
  - Implement enhancements.
  - Interface with other systems.
  - Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
  - Migrate legacy software.
  - Retire software.

Causes of software maintenance problems

- **Lack of Traceability**
  - Codes are rarely traceable to the requirements and design specifications.
  - It makes it very difficult for a programmer to detect and correct a critical defect affecting customer operations.
  - Like a detective, the programmer pores over the program looking for clues.
  - Life Cycle documents are not always produced even as part of a development project.
- **Lack of Code Comments**
  - Most of the software system codes lack adequate comments. Lesser comments may not be helpful in certain situations.
- **Obsolete Legacy Systems**
  - In most of the countries worldwide, the legacy system that provides the backbone of the nation's critical industries, e.g., telecommunications, medical, transportation utility services, were not designed with maintenance in mind.
  - They were not expected to last for a quarter of a century or more!
  - As a consequence, the code supporting these systems is devoid of traceability to the requirements, compliance to design and programming standards and often includes dead, extra and uncommented code, which all make the maintenance task next to the impossible.
- **Problem during Maintenance**
  - Often the program is written by another person or group of persons.
  - Often the program is changed by person who did not understand it clearly.
  - Program listings are not structured.
  - High staff turnover.
  - Information gap.
  - Systems are not designed for change.

Types of Maintenance

- **Corrective maintenance(fault-repair maintenance):**
  - Corrective maintenance 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 the system.
- **Adaptive maintenance(software adaptation):**
  - This includes modifications and updating 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.
- **Perfective maintenance(functionality addition or modification):**
  - A software product needs maintenance to support the new features that the users want or to change different types of functionalities of the system according to the customer demands.
- **Preventive maintenance(predictive maintenance):**
  - Maintenance carried out **at predetermined intervals** or according to **prescribed criteria**, aimed at **reducing the failure risk or performance degradation** of the equipment.
  - This type of maintenance includes modifications and updating to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.

Software Maintenance Plan

- An integral part of software is the maintenance one, which requires an accurate maintenance plan to be prepared during the software development.
- It should specify how users will request modifications or report problems. The budget should include resource and cost estimates.
- A new decision should be addressed for the developing of every new system feature and its quality objectives. The maintenance plan must includes the actual timing requirements, total duration requirements, staff requirements as well as cost requirements in details.
- Software Maintenance planning includes following activities:
  - **Preparation** – Describe software preparation and transition activities including the conception and creation of the maintenance plan; describe how to handle problems identified during development and configuration management.
  - **Modification** – After the application has become the responsibility of the maintenance team, explain how to analyze each request; confirm and check validity; investigate and propose solutions; document the proposal and get the required authorizations to apply the modifications.
  - **Implementation** – Describe the process for considering the implementation of the modification itself.
  - **Acceptance** – Describe how the modification is accepted by the maintenance team.
  - **Migration** – Describe any migration tasks that need to be executed. If the software needs to be moved to another system, outline the steps to do so without impacting its functionality.
  - **Transition** – Document the sequence of activities to transition the system from Development to Maintenance.
  - **Service Level Agreements** – Document SLAs and maintenance contracts negotiated by Maintenance.
  - **Change Request** – Outline the problem-handling process to prioritize, documents and route change and maintenance requests.
  - **Modification Request acceptance/rejection** – Describe the request including details of the size/effort/complexity. If this is too complex to resolve, outline the steps to route the issue back to the software team.
  - **Retirement** – This is the final stage in the lifecycle. Describe how to retire the software and the steps to archive any data that may be a by-product of the system.

Cost of maintenance

- Many organizations allocate eighty percent of information systems budget to maintenance.
- Usually greater than development costs (2* to 100* depending on the application)
- Affected by both technical and non-technical factors
- Increases as software is maintained. Maintenance corrupts the software structure so makes further maintenance more difficult.
- Ageing software can have high support costs (e.g. old languages, compilers etc.)
- Various types of factors that influence system maintainability are:
  - Latent (hidden) defects.
  - Number of customers for a given system.
  - Quality of system documentation.
  - Maintenance personnel.
  - Tools.
  - Well structured programs.
- **Maintance Cost Factors:**
- Team stability
  - Maintenance costs are reduced if the same staff are involved with them for some time
- Contractual responsibility
  - The developers of a system may have no contractual responsibility for maintenance so there is no incentive to design for future change
- Staff skills
  - Maintenance staff are often inexperienced and have limited domain knowledge
- Program age and structure
  - As programs age, their structure is degraded and they become harder to understand and change

Characteristics of Maintainable software

- **Maintainability has "abilities". Including, but not limited to**
  - readability
  - understandability
  - changeability
  - testability
  - Reliability etc.
- Maintainable software exhibits effective **modularity**
- It makes use of design patterns that allow **ease of** understanding.
- It has been constructed using well-defined coding standards and conventions, leading to source code that **is self-documenting and understandable.**
- It has undergone a variety of **quality assurance** techniques that have uncovered potential maintenance problems before the software is released.
- It has been created by software engineers who recognize that they may not be around when changes must be made.
- **How do we attain these abilities?**
- Through application of all those software principles and guidelines you've ever or maybe never heard of. For example:
  - Good comments
  - descriptive variable, method, class names
  - Limit methods to no more that a page in length
  - Maximize cohesion and minimize coupling
  - encapsulate that which stays the same
  - encapsulate that which changes
  - code layout/formatting guidelines applied consistently
  - One entry point only, and one exit point only.

Reengineering

- *Reengineering* is most commonly defined as the redesign of business processes—and the associated systems and organizational structures—to achieve a dramatic improvement in business performance
- Software Reengineering is rorganising and modifying existing software systems to make them more maintainable.
- System reengineering consists of Re-structuring or re-writing part or all of a legacy system without changing its functionality
- Applicable where some but not all sub-systems of a larger system require frequent maintenance
- Re-engineering involves adding effort to make them easier to maintain. The system may be re-structured and re-documented.

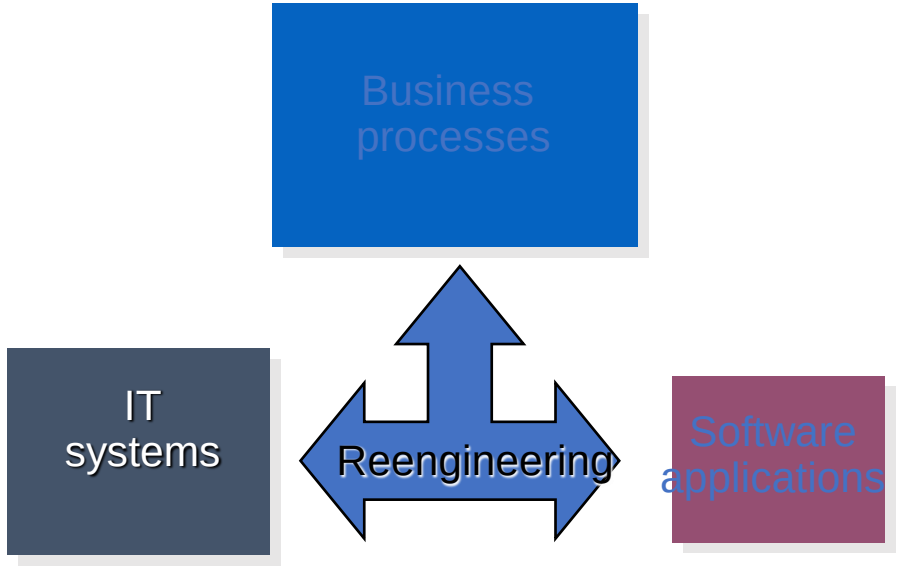


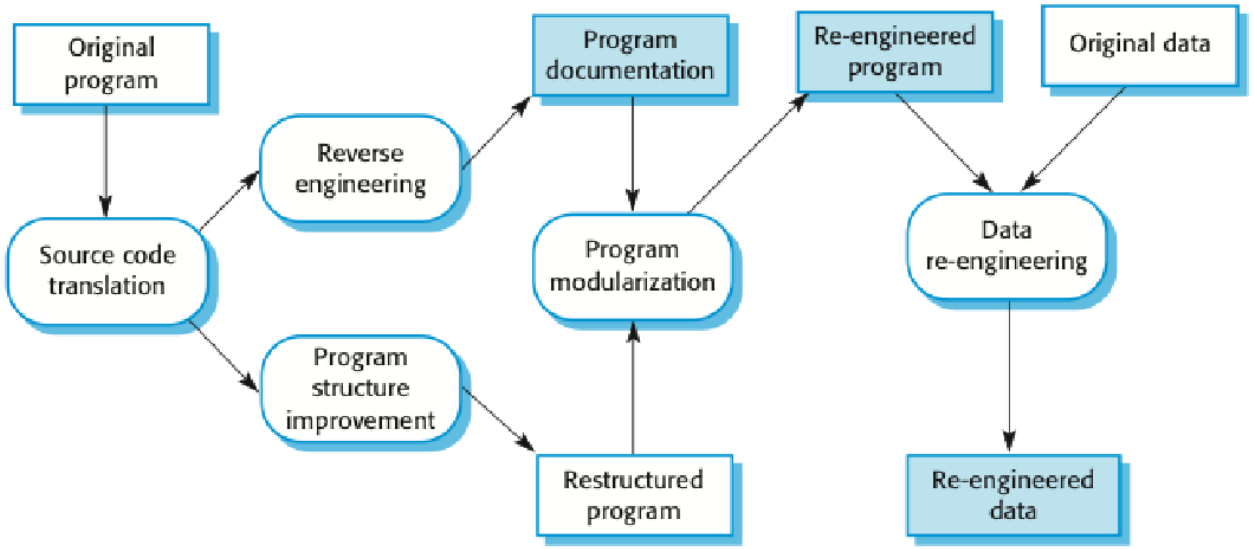
Fig:Reengineering

Motivation for reengineering/When and why reengineering ?

- Reduce costs/expenses (the most cited business-driven reengineering project goal) and reduce risks
- Improve financial performance
- Reduce external competition pressure
- Reverse erosion of market share
- Respond to emerging market opportunities
- Improve customer satisfaction
- Enhance quality of products and services
- When system changes are mostly confined to part of the system then re-engineer that part
- When hardware or software support becomes obsolete
- When tools to support re-structuring are available

Software Reengineering

- **Software Re-engineering** is a process of software development which is done to improve the maintainability of a software system.
- Re-engineering is the examination and alteration of a system to reconstitute it in a new form.
- This process encompasses a combination of sub-processes like reverse engineering, forward engineering, reconstructing etc.



- Source code translation
  - Convert code to a new language.
- Reverse engineering
  - Analyse the program to understand it;
- Program structure improvement
  - Restructure automatically for understandability;
- Program modularisation
  - Reorganise the program structure;
- Data reengineering
  - Clean-up and restructure system data.

Introduction to Configuration Management

- Configuration management (CM) is a systems engineering process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life.
- Configuration management (CM) is an effective strategy designed to help organizations govern control policies and maintain server and data integrity.
- it’s the discipline of ensuring that all software and hardware assets which a company owns are known and tracked at all times—any future changes to these assets are known and tracked.

- **Software Configuration Management(SCM)**
- SCM is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle.
- The primary goal is to increase productivity with minimal mistakes.
- SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

Tasks in SCM

- Configuration Identification
  - determining the scope of the software system
- Baselines
  - A baseline is a formally accepted version of a software configuration item
- Change Control
  - Change control is a procedural method which ensures quality and consistency when changes are made in the configuration object
- Configuration Status Accounting
  - Configuration status accounting tracks each release during the SCM process
- Configuration Audits and Reviews
  - Software Configuration audits verify that all the software product satisfies the baseline needs. It ensures that what is built is what is delivered.

Importance of Configuration Management

- **Disaster Recovery**
- **Uptime and Site Reliability**
- **Easier Scaling**
- **Reduced risk** of outages and security breaches through **visibility** and tracking of the changes to your systems.
- **Cost reduction** by having detailed knowledge of all the elements of your configuration, avoiding wasteful duplication of your technology assets.
- **Improved experience** for your customers and internal staff by rapidly detecting and correcting improper configurations that could negatively impact **performance**.
- **Strict control of your processes** by defining and enforcing formal policies and procedures that govern asset identification, status monitoring, and auditing.
- **Greater agility and faster problem resolution**, enabling you to provide a higher quality of service and reduce software engineering costs.
- Efficient **change management** by knowing your baseline configuration, and having the visibility to design changes that avoid problems.
- Quicker **restoration of service**. In an outage, you'll be able to recover quickly as your configuration is documented and automated.
- Better **release management** and clear status accounting.

Configuration items(CI)

- A configuration item (CI) is any service component, infrastructure element, or other item that needs to be managed in order to ensure the successful delivery of services.
- Each CI has several characteristics:
  - A classification, or type, which indicates what kind of item it is.
  - Attributes, which vary by classification and describe the characteristics of the individual CI.
  - A status value, which represents the CI's state in the lifecycle used for CIs of this classification.
  - Relationships, which indicate how the CI is related to other CIs.
  - An owner, the person who is responsible for the CI.

Versioning

- Versioning is the creation and management of multiple product releases, all of which have the same general function, but are improved, upgraded or customized.
- While many developers and vendors use the term in different contexts, versioning most often applies to operating systems, software artifacts and web services.
- **Software versioning** is the process of assigning either unique *version names* or unique *version numbers* to unique states of computer software.
- Within a given version number category (e.g., major or minor), these numbers are generally assigned in increasing order and correspond to new developments in the software.

Most common methods of software version numbering

- Semantic numbering – Three-digit numbering technique based on Major.Minor.Patch
  - For Example:software version 1.3.1 means that the software is in its first major version, with three additional functionalities added, and one patch fix implemented. When a developer fixes a bug in that version, the next version release is named version 1.3.2.
- Date-of-release – The software version number is the date of the release. For example, 20.06 (June 2020)
- Unary numbering
- Alphanumeric codes: Lotus 1-2-3 Release 1a
- Sequential numbering

Version Control

- At a fine-grained level, revision control is often used for keeping track of incrementally-different versions of information, whether or not this information is computer software.
- Version control, also called source control, allows you to manage changes to files over time, storing these modifications in a database. You can use version control to version code, binary files, and digital assets
- Version control (also known as revision control or source control) is a category of processes and tools designed to keep track of multiple different versions of software, content, documents, websites and other information in development.
- Any system that provides change tracking and control over programming source code and documentation can be considered version control software.
- The purpose of version control is ensuring that content changes under development go as planned.

Benefits of Version Control

1. Enhances the project development speed by providing efficient collaboration,
2. Leverages the productivity, expedite product delivery, and skills of the employees through better communication and assistance,
3. Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
4. Employees or contributor of the project can contribute from anywhere irrespective of the different geographical locations through this **VCS,**
5. For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is **Git, Helix core, Microsoft TFS,**
6. Helps in recovery in case of any disaster or contingent situation,
7. Informs us about Who, What, When, Why changes have been made.