Software Life Cycle Models

Course Instructor: Saurabh Kumar
Assistant Professor
CSE, LNMIIT, Jaipur

Semester: V
CSE: 0326 Software Engineering

October 28, 2020

Outline

- Waterfall model
- Incremental model
- Spiral model
- WINWIN Spiral model
- RAD model
- Prototyping model
- Object-oriented model
- Agile Development model
- Formal Methods
- Client-Server model

Waterfall Model

- It is the simplest model, which states that the phases are organized in a linear order.
- The model was originally proposed by *Royce*.
- The various phases in this model are
  - Feasibility Study: determine whether it would be financially and technically feasible to develop the product.

Waterfall Model

- It is the simplest model, which states that the phases are organized in a linear order.
- The model was originally proposed by *Royce*.
- The various phases in this model are
  - Feasibility Study: determine whether it would be financially and technically feasible to develop the product.
  - Requirement Analysis: understand the exact requirements of the customer and to document them properly.

Waterfall Model

- It is the simplest model, which states that the phases are organized in a linear order.
- The model was originally proposed by *Royce*.
- The various phases in this model are
  - Feasibility Study: determine whether it would be financially and technically feasible to develop the product.
  - Requirement Analysis: understand the exact requirements of the customer and to document them properly.
  - Design: architectural and detailed.

Waterfall Model

- It is the simplest model, which states that the phases are organized in a linear order.
- The model was originally proposed by *Royce*.
- The various phases in this model are
  - Feasibility Study: determine whether it would be financially and technically feasible to develop the product.
  - Requirement Analysis: understand the exact requirements of the customer and to document them properly.
  - Design: architectural and detailed.
  - Implementation: translation of the design specifications into source code, with activities like debugging, documentation and unit testing of the source code.

Waterfall Model

- It is the simplest model, which states that the phases are organized in a linear order.
- The model was originally proposed by *Royce*.
- The various phases in this model are
  - Feasibility Study: determine whether it would be financially and technically feasible to develop the product.
  - Requirement Analysis: understand the exact requirements of the customer and to document them properly.
  - Design: architectural and detailed.
  - Implementation: translation of the design specifications into source code, with activities like debugging, documentation and unit testing of the source code.
  - Testing: integration and acceptance.

Waterfall Model

- It is the simplest model, which states that the phases are organized in a linear order.
- The model was originally proposed by *Royce*.
- The various phases in this model are
  - Feasibility Study: determine whether it would be financially and technically feasible to develop the product.
  - Requirement Analysis: understand the exact requirements of the customer and to document them properly.
  - Design: architectural and detailed.
  - Implementation: translation of the design specifications into source code, with activities like debugging, documentation and unit testing of the source code.
  - Testing: integration and acceptance.
  - Maintenance: corrective, perfective and adaptive.

Waterfall Model

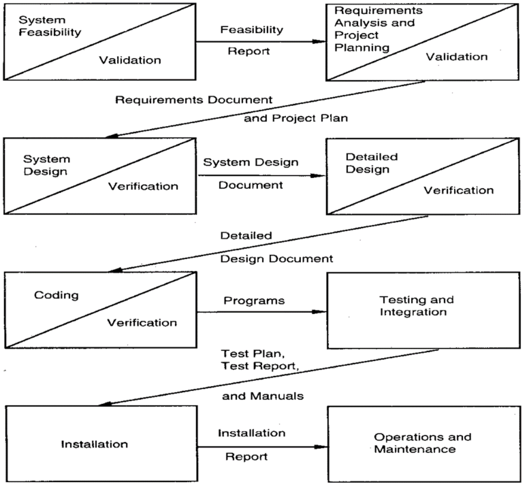

Figure 1: Waterfall Model

Waterfall Model

- Advantages:
  - All phases are clearly defined.
  - One of the most systematic methods for software development.
  - Being oldest, this is one of the time tested models.
  - It is simple and easy to use.

Waterfall Model

- Advantages:
  - All phases are clearly defined.
  - One of the most systematic methods for software development.
  - Being oldest, this is one of the time tested models.
  - It is simple and easy to use.
- Disadvantages:
  - Real Projects rarely follow sequential model.
  - It is often difficult for the customer to state all requirements explicitly.
  - Poor model for complex and object oriented projects.
  - Poor model for long and ongoing projects.
  - High amount of risk and uncertainty.
  - Poor model where requirements are at a moderate to high risk of changing.
  - This model is suited to automate the existing manual system for which all requirements are known before the design starts.

Incremental Model

- It combines elements of the linear sequential model with the iterative philosophy of prototyping.
- It is iterative in nature.
- It focuses on the delivery of operational product with each increment
- The process is repeated until the complete product is produced.
- Example: word-processing software.
- Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.

Incremental Model

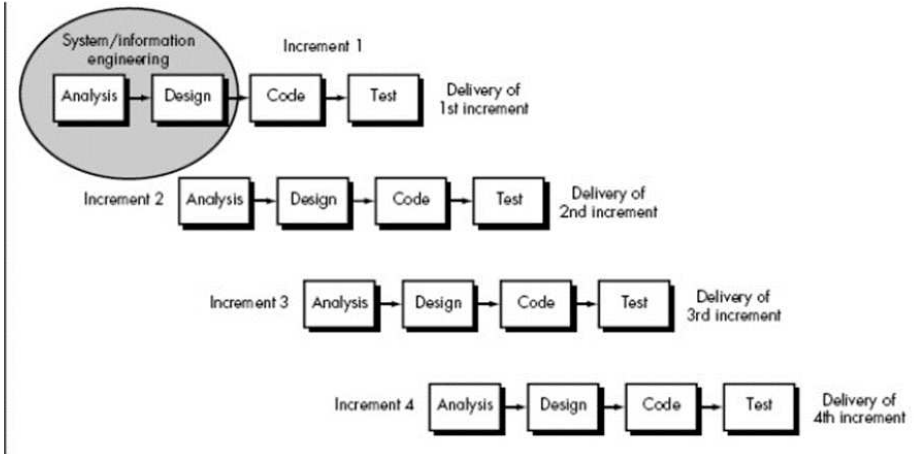

Figure 2: Incremental Model

Incremental Model

- Advantages:
  - It provides on the rigid nature of sequential approach.
  - This method is of great help when organization is low on staffing.
  - Generates working software quickly and early during the software life cycle.
  - More flexible – less costly to change scope and requirements.
  - Easier to test and debug during a smaller iteration.
  - Easier to manage risk because risky pieces are identified and handled during its iteration.
  - Each iteration is an easily managed milestone.

Incremental Model

- Advantages:
  - It provides on the rigid nature of sequential approach.
  - This method is of great help when organization is low on staffing.
  - Generates working software quickly and early during the software life cycle.
  - More flexible – less costly to change scope and requirements.
  - Easier to test and debug during a smaller iteration.
  - Easier to manage risk because risky pieces are identified and handled during its iteration.
  - Each iteration is an easily managed milestone.
- Disadvantages:
  - This model could be time consuming.
  - Each phase of an iteration is rigid and do not overlap each other.
  - Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

Spiral Model

- Evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model.
- It provides the potential for rapid development of incremental versions of the software.
- The software is developed in a series of incremental releases.
  - During early iterations, the incremental release might be a paper model or prototype.
  - During later iterations, increasingly more complete versions of the engineered system are produced.
- A spiral model is divided into framework activities or task regions.
  - Customer communication
  - Planning
  - Risk analysis
  - Engineering
  - Construction and release
  - Customer evaluation

Spiral Model

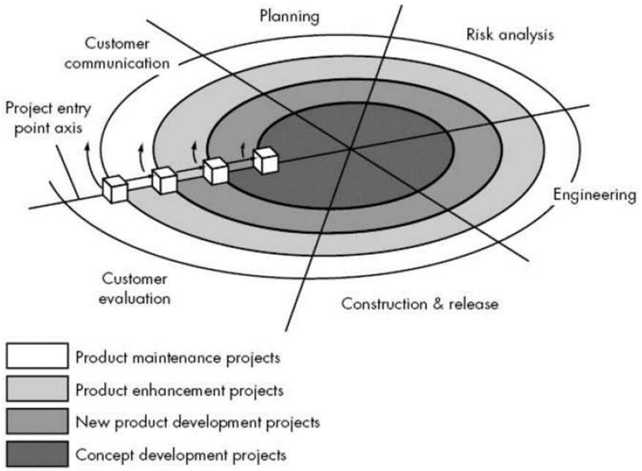

Figure 3: Spiral Model

Spiral Model

- Advantages:
  - It is a realistic approach for development of large scale system.
  - High amount of risk analysis
  - Good for large and mission-critical projects.
  - Software is produced early in the software life cycle.

Spiral Model

- Advantages:
  - It is a realistic approach for development of large scale system.
  - High amount of risk analysis
  - Good for large and mission-critical projects.
  - Software is produced early in the software life cycle.
- Disadvantages:
  - It is not widely used.
  - It may be difficult to convince customers that the evolutionary approach is controllable.
  - It demands considerable risk assessment expertise and relies on this expertise for success. If a major risk is not uncovered and managed, problems will undoubtedly occur.
  - Can be a costly model to use.
  - Risk an alysis requires highly specific expertise.
  - Project's success is highly dependent on the risk analysis phase.
  - Does not work well for smaller projects.

WINWIN Spiral Model

- First WIN is for customer and second WIN is for developer.
- The best negotiations strive for a WIN-WIN result.
- the following activities are defined.
  - Identification of the system or subsystem's key Stake holders.
  - Determination of the Stake holder's WIN conditions.
  - Negotiations of the Stake holder's WIN conditions to reconcile them into a set of WIN-WIN conditions for all concerned.
- Anchor points
  - Life Cycle Objectives (LCO): defines a set of objectives for each major software engineering activity.
  - Life Cycle Architecture (LCA): establishes objectives that must be met as the system and software architecture is defined.
  - Initial Operational Capability (IOC): represents a set of objectives associated with the preparation of the software.

WINWIN Spiral Model

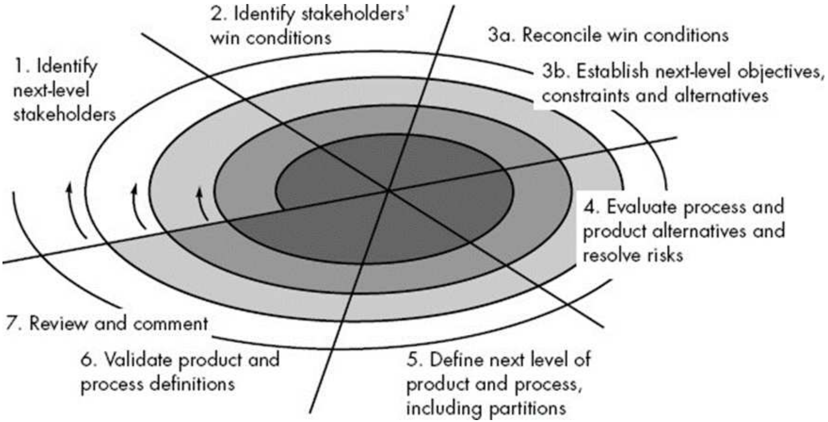
2. Identify stakeholders' win conditions
3a. Reconcile win conditions
3b. Establish next-level objectives, constraints and alternatives
1. Identify next-level stakeholders
4. Evaluate process and product alternatives and resolve risks
7. Review and comment
6. Validate product and process definitions
5. Define next level of product and process, including partitions

Figure 4: WINWIN Spiral Model

WINWIN Spiral Model

- Advantages:
  - Flexibility: The model let the teams adapt to accompanying risks and uncertainties, such as a rapid project schedule and changing team composition.
  - Discipline: The modeling framework was sufficiently formal to maintain focus on achieving three main anchor-points.
  - Trust Enhancement: The model provides a means for growing trust among the project stakeholders, enabling them to evolve from adversarial, contact-oriented system development approaches.

RAD Model

- Rapid application development (RAD) is an incremental software development process model that emphasizes an extremely short development cycle.
- The RAD model is a high-speed adaptation of the linear sequential model in which rapid development is achieved by using component-based construction.
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within very short time periods (e.g., 60 to 90 days).
- Used primarily for information systems applications.
- RAD approach encompasses the following phases:
    - Business Modeling
    - Data Modeling
    - Process Modeling
    - Application generation
    - Testing and turnover

RAD Model

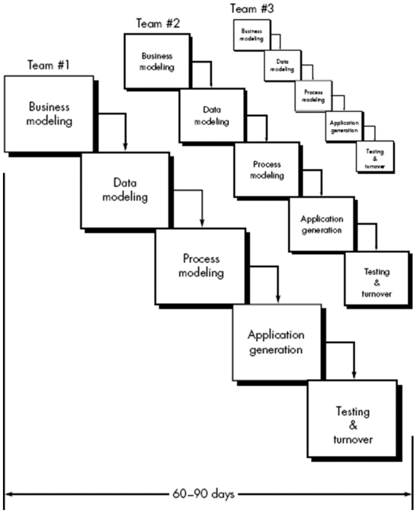

Figure 5: RAD Model

RAD Model

- Advantages:
  - As RAD stresses on reuse of components testing time may be very less in certain cases.
  - RAD model can be most efficient when it comes to overall development time.

RAD Model

- Advantages:
  - As RAD stresses on reuse of components testing time may be very less in certain cases.
  - RAD model can be most efficient when it comes to overall development time.
- Disadvantages:
  - For large but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
  - RAD requires developers and customers who are committed to the rapid-fire activities necessary to get a system complete in a much abbreviated time frame.
  - If commitment is lacking from either constituency, RAD projects will fail.
  - Not all types of applications are appropriate for RAD.
    - If a system cannot be properly modularized, building the components necessary for RAD will be problematic.
    - If high performance is an issue and performance is to be achieved through tuning the interfaces to system components, the RAD approach may not work.
  - RAD is not appropriate when technical risks are high.
    - This occurs when a new application makes heavy use of new technology or when the new software requires a high degree of interoperability with existing computer programs.

Prototyping Model

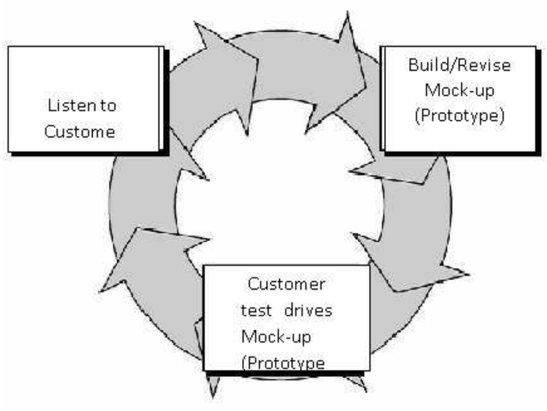
Listen to Custome
Build/Revise Mock-up (Prototype)
Customer test drives Mock-up (Prototype

Figure 6: Prototyping Model

Prototyping Model

- Advantages:
  - It serves as the mechanism for identifying the requirements.
  - The developer use the existing program fragment means it helps to generate the software quickly.
  - Continuous developer – Consumer communication is available.

Prototyping Model

- Advantages:
  - It serves as the mechanism for identifying the requirements.
  - The developer use the existing program fragment means it helps to generate the software quickly.
  - Continuous developer – Consumer communication is available.
- Disadvantages:
  - Customer considers the prototype as an original working version of the software.
  - Developer makes implementation compromise in order to get prototype working quickly.
  - This model is time consuming.

Object-Oriented Model

- Also known as component assembly model.

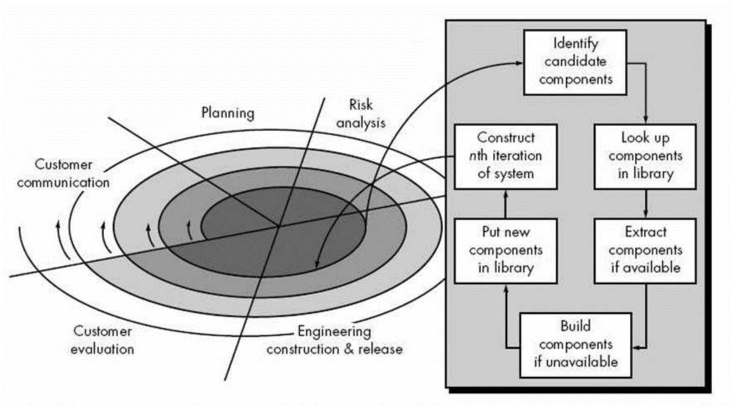
Planning
Risk analysis
Customer communication
Customer evaluation
Engineering construction & release
Identify candidate components
Construct nth iteration of system
Look up components in library
Put new components in library
Extract components if available
Build components if unavailable

Figure 7: Object-oriented Model

Agile Development Model

- In earlier days, iterative waterfall model was very popular to complete a project.
- Nowadays, the developers face many problems in using it such as
  - handling change requests from customers during project development; and
  - high cost and time required to incorporate the changes.
- To overcome the mentioned drawbacks, the Agile software development model was proposed in the mid 1990s.
- Agent model was primarily designed to help a project to adapt to change requests quickly.
- The main aim of Agile model is to facilitate quick project completion.
- To accomplish this task, agility is required.
- Agility is achieved by fitting the process to the project, removing activities that may not be essential for a specific project.
- Agility avoids anything that is wastage of time and effort.

Agile Development Model

- Agile model refers to a group of development processes.
- These processes share some basic characteristics but do have certain subtle differences among themselves.
- A few Agile SDLC models are given below:
  - Crystal
  - Atern
  - Feature-driven development
  - Scrum
  - Extreme Programming (XP)
  - Lean development
  - Unified process
- In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed.
- The Agile model is the combination of adopts iterative and incremental process models.
- Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only.
- At a time one iteration is planned, developed and deployed to the customers. Long-term plans are not made.

Agile Development Model

- Steps involve in agile SDLC models are:
  - Requirement gathering
  - Requirement analysis
  - Design
  - Coding
  - Unit testing
  - Acceptance testing
- The time to complete an iteration is known as a Time Box.
- Time-box refers to the maximum amount of time needed to deliver an iteration to customers.
- The end date for an iteration does not change.
- Though the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time.
- The central principle of the Agile model is the delivery of an increment to the customer after each Time-box.

Agile Development Model

- Principles of Agile model
  - To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
  - Agile model relies on working software deployment rather than comprehensive documentation.
  - Frequent delivery of incremental versions of the software to the customer representative in intervals of few weeks. Requirement change requests from the customer are encouraged and efficiently incorporated.
  - It emphasizes on having efficient team members and enhancing communications among them is given more importance. It is realized that enhanced communication among the development team members can be achieved through face-to-face communication rather than through the exchange of formal documents.

Agile Development Model

- It is recommended that the development team size should be kept small (5 to 9 peoples) to help the team members meaningfully engage in face-to-face communication and have collaborative work environment.
- Agile development process usually deploy Pair Programming. In Pair programming, two programmers work together at one work-station. One does coding while the other reviews the code as it is typed in. The two programmers switch their roles every hour or so.

Agile Development Model

- Advantages
  - Working through Pair programming produce well written compact programs which has fewer errors as compared to programmers working alone.
  - It reduces total development time of the whole project.
  - Customer representative gets the idea of updated software products after each iteration. So, it is easy to change any requirement if needed.

Agile Development Model

- Advantages
  - Working through Pair programming produce well written compact programs which has fewer errors as compared to programmers working alone.
  - It reduces total development time of the whole project.
  - Customer representative gets the idea of updated software products after each iteration. So, it is easy to change any requirement if needed.
- Disadvantages
  - Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
  - Due to absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

Verification & Validation

- Goal: asses and improve the quality of the work products generated during development and modification of software.
- Qualitative attributes
  - Correctness
  - Completeness
  - Consistency
  - Reliability
  - Usefulness
  - Conformance to standards
  - Overall cost effectiveness
- Assessment of work products to determine conformance to the specifications.
- Specifications include
  - Requirements specification
  - The design documentation
  - Various stylistic guidelines
  - Implementation language standards
  - Project standards
  - Organizational standards
  - User expectations

Verification

- Verification is done to ensure that the work product of a given phase of the development cycle fulfill the specifications established during prior phase.
- According to Boehm, Verification: Are we producing the product right?
- Verification ensures that the product is designed to deliver all functionality to the customer.
- It typically involves reviews and meetings to evaluate documents, plans, code, requirements and specifications.
- This can be done with checklists, issues lists, walkthroughs and inspection meetings.
- In verification, uncovering of defects will be done in primary ways.
- Here no code will be executed.
- Before building actual system this checking will be done. This process will be called Quality Assurance.
- Verification is nothing but the Static Testing.
- Inputs: check list, issue list, walkthroughs and inspection meetings, reviews and meetings.
- Output: nearly a perfect set of documents, plans, specifications and requirements document.

Validation

- Validation is the process of evaluating software at the end of the software development process to determine compliance with the requirements.
- According to Boehm, Validation: Are we producing the right product?
- Validation ensures that the functionality, as defined in requirements, is the intended behavior of the product.
- Validation typically involves actual testing and takes place after verifications are completed.
- Validation concern, checking will be done by executing code for errors (defects.). This can be called as Quality Control.
- Validation is nothing but the Dynamic Testing.
- Inputs: actual testing of an actual product.
- Output: a nearly perfect, actual product.

Software Configuration Management

- **SCM** is an umbrella activity that is applied throughout the software process, because change can occur at any time.
- SCM activities are developed to
  - Identify change,
  - Control change,
  - Ensure that change is being properly implemented, and
  - Report changes to others who may have an interest.
- SCM has a clear distinction from software support.
- Output of software process has three broad categories:
  - Computer programs (both source level and executable forms);
  - Documents that describe the computer programs (targeted at both technical practitioners and users), and
  - Data (contained within the program or external to it).
- Software Configuration: collection of the items that comprise all information produced as part of the software process.

Software Configuration Management

- Four fundamental sources of change:
  - New business or market conditions dictate changes in product requirements or business rules.
  - New customer needs demand modification of data produced by information systems, functionality delivered by products, or services delivered by a computer-based system.
  - Reorganization or business growth/downsizing causes changes in project priorities or software engineering team structure.
  - Budgetary or scheduling constraints cause a redefinition of the system or product.
- SCM: a set of activities that have been developed to manage the change throughout the life cycle of software process.

Software Configuration Management

- Baseline: A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures. (IEEE Definition)

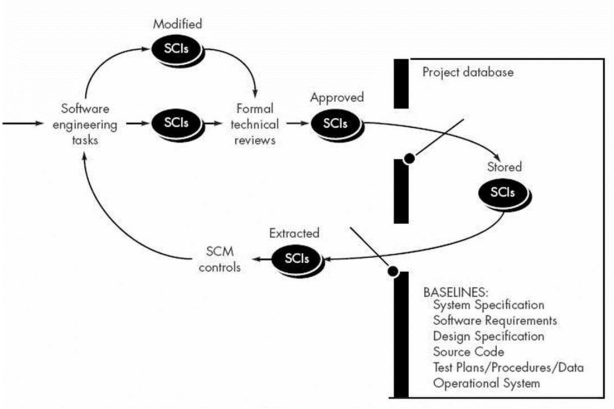

Figure 8: Baselined SCIs and the project database

SCM Process

- Identification of objects:
  - To control and manage SCIs, each of them must be separately named and organized using object-oriented approach.
  - Categories of objects
    - Basic Object: a **unit of text** created by a software engineer during analysis, design, coding or testing phases. Example: a section of a requirements specification, a source listing for a component, or a suite of test cases that are used to exercise the code.
    - Aggregate Object: a collection of basic objects and other aggregate objects. Example: Design Specification is an aggregate object.
  - Each object should have
    - A unique name
    - A description of the object including its SCI type (eg – Document, program or data, a project identifier and version information)
    - A list of resources, which are entities that are processed, provided or required by the object.

SCM Process

- Version Control:
  - It combines various tools and procedures to manage and control different versions of SCI objects (as a result of change) that are created during the software engineering process.
  - Clemm describes version control in the context of SCM:
    Configuration management allows a user to specify alternative configurations of the software system through the selection of appropriate versions. This is supported by associating attributes with each software version, and then allowing a configuration to be specified (and constructed) by describing the set of desired attributes.
  - The attributes can be as simple as a specific version number that is attached to each object.

SCM Process

- Change Control:
  - For a large system development project, uncontrolled change rapidly leads to confusion and inconsistencies.
  - Change control includes human procedures and automated tools to control the reasons for change.
  - The following processes take place when a situation of change occurs:
    - Change request: Technical merit, Potential side effects, subsystems and the cost for implementing the change.
    - Change report: change report is submitted to the Change Control Authority/Board (CCA or CCB).
    - Engineering Change Order (ECO): The description of the change to be made, Constraints that has to be taken care of, and Criteria for review and audit.
    - Check out & check in: The object to be changed is checked out of the project database, the decided changes are made and appropriate Software Quality Assurance (SQA) activities are performed. The object is then checked in the project database and appropriate version control mechanisms are used to create the next version of the software.

SCM Process

- Configuration Audit: Two activities are performed:
  - Formal Technical Review (FTR)
    - To uncover errors in functions, logic or implementation.
    - To verify that the software under review should meet its requirement.
    - To ensure that the representation of the software is according to the standards.
    - To make the project more manageable.
    - It consists of walkthroughs, inspections and round-robin reviews.
  - Software Configuration Audit
    - To check whether the changes specified in the ECO has been properly made and to check if any additional modules are added.
    - To check whether formal technical reviews are conducted for the assessment of technical correctness.
    - To check whether SE standards are properly followed.
    - To check whether the change is highlighted in the SCI documentation and also the attributes of the configuration object should reflect the change.
    - To check whether SCM procedures like noting change, recording it and reporting it has been properly done.
    - To check whether all the related SCIs are properly updated.

SCM Process

- Reporting: summarizes the activities done so far which includes the following
  - The report of all the activities done.
  - The report on the persons involved in the above reported activities.
  - The report on the time and date when the reported activities were accomplished.
  - The report on various other factors or objects that will be affected due to the reported activity.
- Following are the situations where the CSR needs updating
  - Each time when a SCI is assigned a new or updated identification.
  - Each time ECO is issued, i.e when a change is approved by the CCA/CCB.
  - Each time a configuration audit is conducted.
- The CSR is made available online and is updated occasionally in order to keep the management and concerned individuals updated on the changes.
- It improves the communication among all the people involved.