UNIT-I

21CSC303J

SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

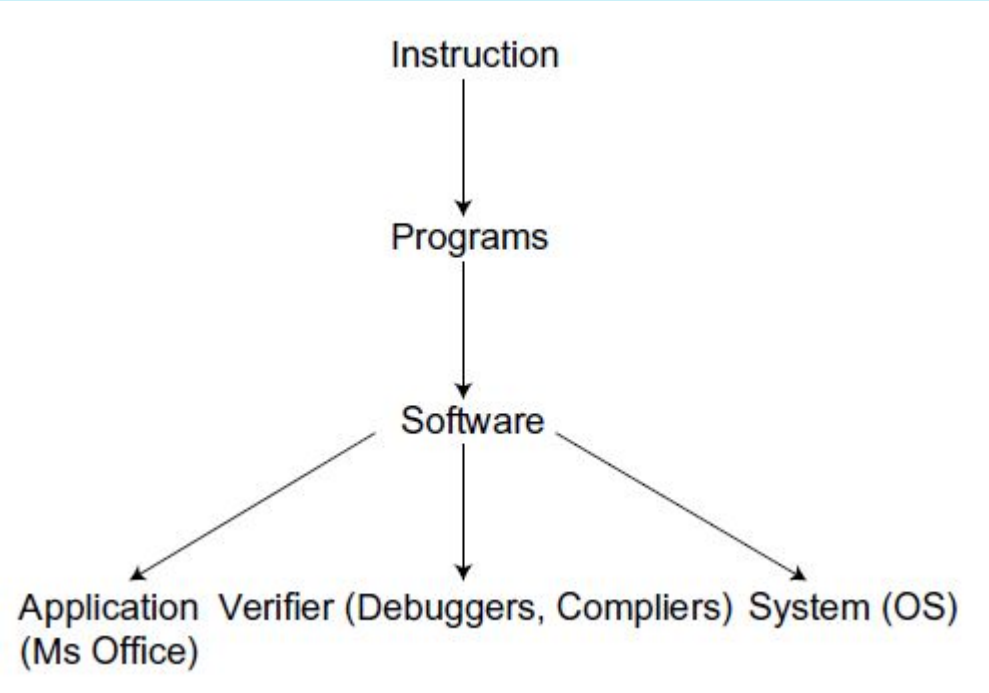
Unit 1- Syllabus

- Introduction to Software Engineering: The evolving role of software,
- Changing nature of software
- Generic view of software engineering
- Software engineering- A layered technology,
- A process framework
- Software Project Management - life cycle activities
- Process models:
  - Waterfall model,
  - Evolutionary models
  - Incremental models
- The unified process
- Agile- XP, Scrum
- Project Initiation management
- Project Charter,
- Project Scope,
- Project Objectives,
- Practical considerations.

Introduction to Software Engineering

- Software is a collection of similar tasks that provides a single interface to the user to obtain the results for a variety of requests.
- Software is the controlling unit of a hardware component.
- The software defines the functions of data transfer, path of control messages and transfer of input and output between the hardware.
- Software engineering deals with the terminologies of designing and delivering software to the client's satisfaction.
- Software = Computer Programs (Instructions) + Procedures + Rules + Associated documents + Data
- Procedures and rules include step-by-step instructions of how to install (deploy) and use the software.
- Documents include requirement documents, design documents and testing-related documents.
- To run the software basic data needs to be fed in.

Introduction to Software Engineering



The evolving role of software

Changing Perceptions of Computers and Software (1970s to 2000s)

1970s - 1980s: Early Predictions of Transformation

- **1979:** Osborne [OSB79] described the computer revolution as a "new industrial revolution."
- **1980:** Toffler [TOF80] introduced the concept of the "third wave of change," highlighting the rise of microelectronics.
- **1982:** Naisbitt [NAI82] predicted a societal shift from an industrial society to an "information society."
- **1983:** Feigenbaum and McCorduck [FEI83] emphasized the importance of information as the new source of power.
- **1989:** Stoll [STO89] discussed the growing importance of electronic communities in knowledge exchange.

The evolving role of software

1990s: The Rise of Software and Internet Culture

- **1990:** Toffler [TOF90] noted a "power shift" due to the democratization of knowledge through computers and software.
- **1992:** Yourdon [YOU92] warned of the decline of American software competitiveness, predicting the fall of the American programmer.
- **1993:** Hammer and Champy [HAM93] argued that information technologies would play a key role in reengineering corporations.
- **Mid-1990s:** Neo-Luddite authors criticized the growing influence of computers (e.g., *Resisting the Virtual Life* and *The Future Does Not Compute*).
- **1996:** Yourdon [YOU96] reversed his pessimistic outlook, suggesting a "rise and resurrection" of the American programmer as the internet grew.

The evolving role of software

Late 1990s: Y2K and the Rise of Ubiquitous Computing

- **1998-1999:** Writings on the Y2K "time bomb" (e.g., [YOU98b], [DEJ98], [KAR99]) underscored the growing dependence on software, although the predicted crisis never materialized.
- **1998-1999:** The concept of "ubiquitous computing" [NOR98] and connected devices became prominent, suggesting a future of constant web connectivity [LEV99].

2000s and Beyond: Software's Expanding Role

- Software evolved into both a product and a delivery mechanism for various technologies, transforming data management, business processes, and global connectivity.

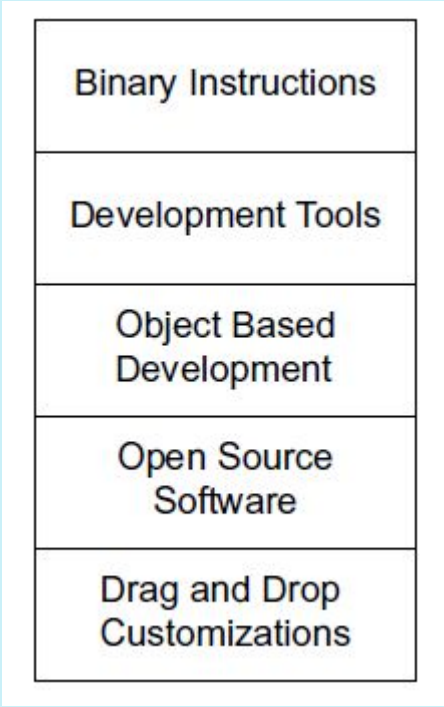
The evolving role of software

- The shift from lone programmers to specialized software development teams introduced new challenges:
  - High development costs
  - Long project timelines
  - Difficulty in error detection and progress measurement

This evolving landscape led to the adoption of **software engineering practices** to address these issues and manage the increasing complexity of modern computer-based systems.

Changing nature of software

- In the beginning, binary instructions were fed into the computer machine.
- Now-a-days Object-based software development is in use.
- The source code was earlier visible only to the development team but is now available to the customers as well.
- Open source software is a big hit in the market and anybody can change the code while using and customizing it as per the need.
- Tools are now developed to facilitate the users to design a software product themselves by dragging and dropping the components (contents) at their intended positions.
- Proper knowledge of coding and testing is not required to design and code the software.
- Software development tools simplifi ed the design and implementation phases drastically.
- The process of developing software becomes simple.



Changing nature of software

In-built Software

- Various mechanisms of automobiles, planes and robots employed for various operations in the fields of military and research involve dedicated software with complex operations and decision-making skills.
- These devices need embedded software as they involve serious analysis and take appropriate actions at the right time.
- The in-built software are deeply sensitive and efficient, prescribing the intelligence functions of the devices.
- The intelligence includes decision-making skill sets because no human can be practically involved to command the robot, which reports on the atmospheric conditions of the planet Mars.
- Engineering and scientifi c research use this type of software mostly, reducing the participation of the mankind in in-human conditions.

Changing nature of software

System-based Software

- The system-based software has also some prescribed function, but they are limited to simpler devices.
- Home computers or computers of any kind necessitate an operating system and additional driver softwares for the functioning of the entire system.
- The system-based software is related to programs,which define the functionality of the hardware components of a particular computer.
- The operation of a computer as a whole is facilitated by the system software containing the basic and advanced modes of operations at different instances of the end user.
- Without this software, the hardware is merely a machine. Compliers, linkers and debuggers are also some of the important system software for testing the programs.

Changing nature of software

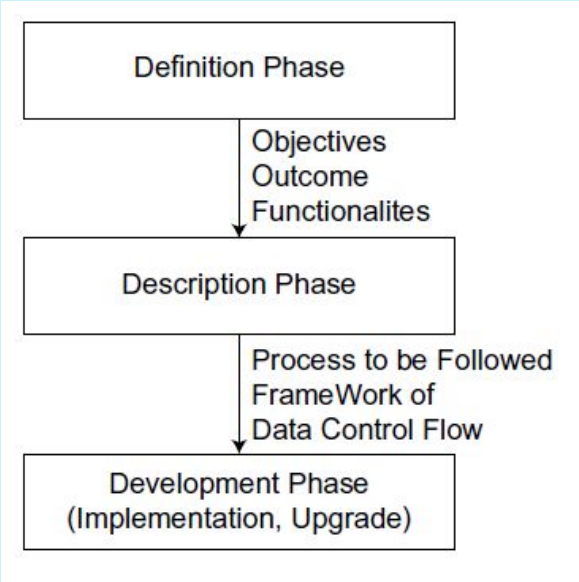
Application-based software

- These kinds of software delivers functionality or service to the requesting users.
- They obtain an input condition or a query from the end user, analyze it and display the results on the monitor.
- They are not as complex as the above-mentioned type.
- The structure and the language used for the applications vary by a large extent, such as for a web-based application the HTML and XML languages are used while for a database application, an ORACLE program is used.
- Different applications possess distinguishable styles and frameworks.
- Every field has its very own requirements, area of interests, functional aspects and thus different application softwares belong to different category and need not be the same.

Generic View of Software Engineering

- The sequential steps that describe the stages of a product being developed are the generic view of software engineering.
- Every action required for the development is categorized into three generic stages (phases).
- **Definition Phase:** The desired outcomes, functionalities and objectives are obtained from the customers and documented. These objectives are the driving force for a product to be developed. The team of developers and customers agree upon the final and feasible list of goals to be implemented into the product.
- **Description Phase:** The next generic phase takes the necessary steps for further development;the desired outcomes are framed with optimal preceding and succeeding processes. The framework of data and control flow in a product, how the processes are ordered and how the functionalities are defined are described in this phase.
- **Deployment Phase:** After the processes are described, they have to be transformed into the corresponding codes and implemented in the real-world environments. Implementation of the software product also includes the training provided to the customers and end users, maintenance and upgrade at regular intervals.

Generic View of Software Engineering



Software engineering- A layered technology

- SE methods provide the technical how-to's for building software.
- Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing, and support.
- SE methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.
- SE tools provide automated or semi-automated support for the process and the methods.
- When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided SE, is established.
- CASE combines software, hardware, and a SE database to create a SE environment analogous to CAD/CAE (computer-aided design/engineering) for hardware.

Software engineering- A layered technology

Software engineering- A layered technology

- Software engineering is a layered technology.
- Total quality management and similar philosophies foster a continuous process improvement culture, this leads to the development of increasingly more mature approaches to software engineering.
- The foundation for software engineering is the process layer.
- Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.
- Process defines a framework for a set of key process areas (KPAs) that must be established for effective delivery of software engineering technology.
- KPA form the basis for management control of software projects and establish the context in which technical methods are applied, work products are produced, milestones are established, quality is ensured, and change is properly managed.

Software engineering- A layered technology

The layered model of software engineering divides the activities into four broad categories:

- **Quality Process:** For the successful engineering process there should be a strong base of quality Processes.
- **Process**: The software engineering processes ensure that all the technology layers work together and enable the timely development of the software system.
- **Methods**: The methods in software engineering provide the technical capabilities to the system. The requirement analysis, design, modeling, development, testing, etc. are the tasks related to the methods.
- **Tools**: The tools provide the support to the process and methods by making the tasks automated or semi-automated.

Process Framework

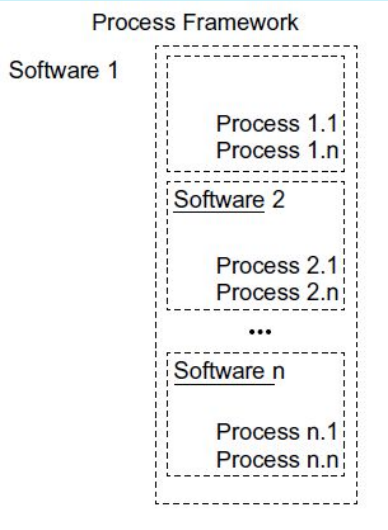
- Framework activities are processes of basic functionalities and are common to almost all software products. These actions can be applied to every product without any further modifications.

Analysis on the Process Framework Activities

- **Gathering the requirements:** The customary process framework activities begin with the analysis and observations made on the requirements. This process called as communication phase involves intended customers and the team of developers. Concluding the communication process produces a list of gathered and approved requirements.
- **Scheduling and staff allocation:** Based on the gathered information and the plans the model is selected for development, processes are framed and the analysis on the feasible risks are made. In addition, the schedule for the development is also agreed. Appropriate members are assigned depending on the knowledge possessed for every task.
- **Design and Deployment:** The team members start designing and developing the coding for their assigned task. The design process may be eliminated by reusing a module. Efficient modeling and processes ease the risks at the later stages of a product.

Process Framework

- **Supplementary Framework Activities:** The product being developed has to be checked at regular intervals to ensure that it is proceeding in the right path. Slight deviations from the intended path would result in starting over the whole process again. Risk analysis is also another major course of action to prevent disastrous outcomes at the end. Detection and mitigation of risks at the earlier stages would shorten the final stages.



Software Project Management – life cycle activities

- Software project management, has four major phases,:
  - Planning
  - Organizing
  - Monitoring
  - Adjusting
- The POMA management process, starts with the planning of tasks and moves through the remaining categories of project management activities.
- Unlike software engineering, which begins with a development and support process and then applies management to that process, POMA starts with management and applies software engineering's domain-specific knowledge, such as a requirements solicitation method or software measurement method, at various stages along the way.
- POMA models the software management life cycle much as a waterfall software process models the software development life cycle.
- Note that the activities in the four process categories of POMA are not necessarily sequential.
- Some activities within each category may overlap, and the categories themselves may overlap.

Software Project Management – life cycle activities

Planning

- Software project planning consists of a set of activities that will develop a plan of attack for the project.
  - The description of the software product in form of software artifact contents & deliverables
  - The software product attributes
  - The schedule required to complete the project
  - The types of and amount of resources needed to meet the project schedule
  - The relevant measurements that would be used to gauge the status of the software project and to assess the final project “success”
  - The risks associated with the project
- Project planning includes a time-consuming and important set of tasks unfortunately rushed.
- It is much wiser to spend the appropriate time needed to develop a good plan initially than to have to make multiple and costly adjustments later.
- Even with a well-conceived plan, it is unusual not to encounter some conditions that require unexpected changes during the project.
- However, having a well-thought-out plan facilitates making project adjustments even at a much later phase.

Software Project Management – life cycle activities

Organizing

- Software project organization seeks to construct a software development,support, and service organization based on the project plan.
- To build and implement the software project organization, acquiring the various skilled individuals needed for the project, defining a process and a set of methodologies that will be utilized for the software project, obtaining a set of tools that will support the process and the methodologies, and creating a well-defined set of metrics that will be used to track and gauge the project.
- A significant portion of the task of organizing activities is ensuring that all personnel brought on board are properly equipped to perform their designated tasks.
- This equipping of personnel includes obtaining needed tools and preparing facilities for initiation of the project, and it also includes educating the personnel in using those tools, the methodology selected, and the metrics chosen.
- As part of the organizing activities, project managers need to ensure that adequate financial funding has been set aside and will be made available in a timely manner.
- Thus, the software project management team must either include financial management and personnel management members or have well-defined interfaces with these other organizations.
- Just like any organizational interface, the software project organization relies on human relationships that need to be established, nurtured, and maintained.

Software Project Management – life cycle activities

Monitoring

- Software project monitoring focuses on the following activities:
  - Consistently and regularly collecting measurements
  - Analyzing the data
  - Representing and presenting the data for a defined set of reports
  - Making projections and making recommendations based on the analysis of the data
- Software project management, like any other project management situation, involves a heavy dosage of “people” management.
- Therefore, the project monitoring component must include the soft art of both physically and virtually “walking around the hallway” and tapping into day-to-day issues, concerns, and morale.

Software Project Management – life cycle activities

Adjusting

- Adjustments and changes are a very important set of activities for software project managers.
- It is a rare situation in which we can develop a perfect plan and put together a perfect organization.
- Most of the time, we will need to make mid-course adjustments, sometimes several times.
- The monitoring of projects ensures that the correct adjustments can be made at a relatively early stage in the development process.
- In a situation that is often encountered, the design of a software artifact is found to have a high number of errors that are attributable to unclear requirements.
- In this situation, the software project managers might need to commission a rework on the requirements and ask for adjustments in the project schedule, resources, or content.
- For a software project manager to make adjustments, he or she must first recognize the need for making a change—and the risks of not doing so.
- Then a potential set of adjustments must be available, along with some prioritization scheme.

Software process models

- To solve actual problems in an industry setting, a software engineer must incorporate a development strategy that encompasses the process, methods, and tools layers. This strategy is often referred to as a process model or a software engineering paradigm.
- A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

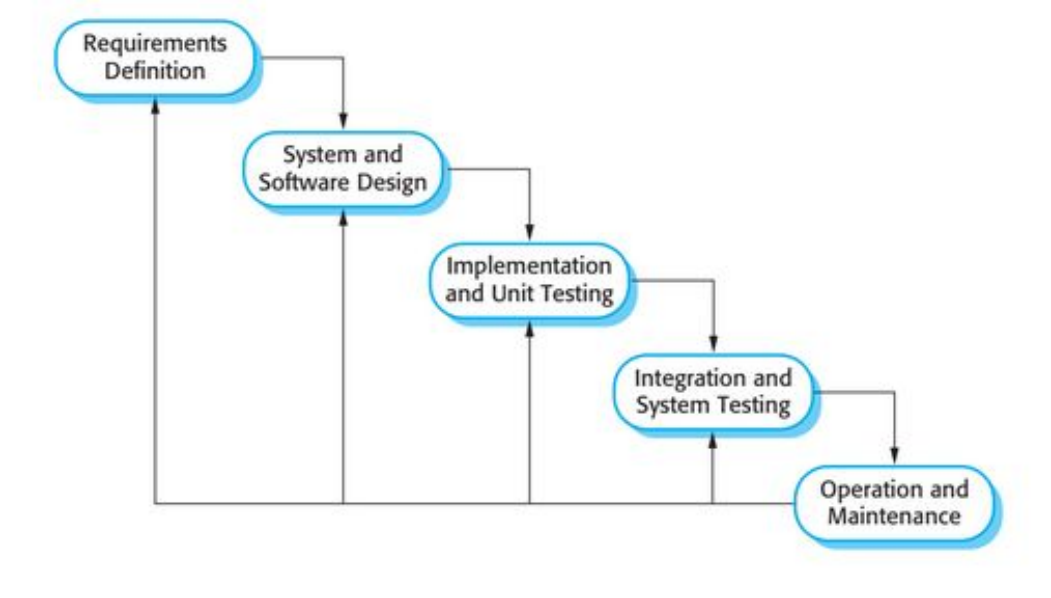
Software process models

The waterfall model

- The waterfall model is an example of a plan-driven process—in principle, must plan and schedule all of the process activities before starting work on them.
- This takes the fundamental process activities of specification, development, validation, & evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.
- The waterfall model is consistent with other engineering process models and documentation is produced at each phase.
- In principle, the waterfall model should only be used when the requirements are well understood and unlikely to change radically during system development.
- However, the waterfall model reflects the type of process used in other engineering projects.
- As is easier to use a common management model for the whole project, software processes based on the waterfall model are still commonly used.

Software process models

The waterfall model



Software process models

The waterfall model

The principal stages of the waterfall model directly reflect the fundamental development activities:

1. Requirements analysis and definition The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

2. System and software design The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. Implementation and unit testing During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

4. Integration and system testing The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

5. Operation and maintenance Normally, this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

Software process models

The waterfall model

Advantages of Waterfall Model

1. Easy to understand
2. Easy for implementation
3. Widely used and known; hence, all stakeholders understand this
4. Identifi es deliverables and milestones upfront

Disadvantages of Waterfall Model

1. Does not match well with reality
2. It is unrealistic to freeze the requirement upfront
3. Software is delivered only at the last phase
4. Difficult and expensive to make changes

Software process models

Evolutionary software process models

- Evolutionary models are iterative.
- They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.
- The Evolutionary process models smoothen the progress of a software development in a number of ways.
- These strategies are like - creating prototypes and making steps overlapping and autonomous.
- The demerits of linear sequential models are eliminated and these models stand among the best process models in software development.
- Some of the evolutionary process models are described as follows:
    - Spiral Model
    - Components-based Development Model
    - Formal Methods Model

Software process models

Evolutionary software process models

Spiral Model

- The spiral model was proposed by Boehm as an evolutionary model; it introduced the concept of risk analysis in the earlier stages of development process.
- Spiral model is also called Risk Spiral Model.
- This model combines the software development life cycle with Risk Management principles to control the risks at early stage of the project.
- A spiral model encompasses the iterative steps in a spiral representation
- Each circle in the model denotes a succeeding level of the previous state.
- These are divided task regions, which represent the selective process.
- The task regions may vary from the complexity level of different products.
- The planning part includes the tasks for designing an efficient framework as a solution to the objectives.
- Planning also happens in steps:
    - Life Cycle Plan is prepared first;
    - Development Plan is prepared in the next level; and
    - Integration Plan is in the last level.

Software process models

Evolutionary software process models

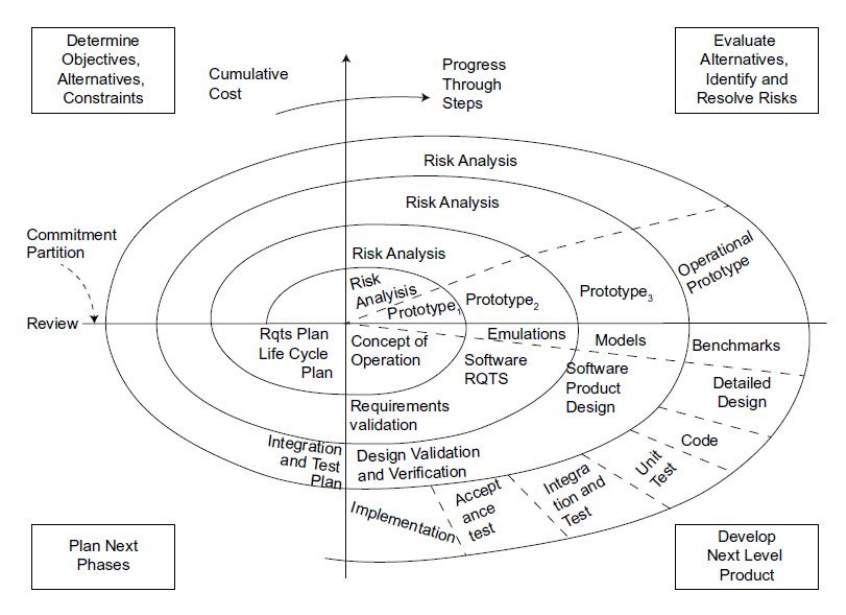
Spiral Model

- In the Risk Analysis Phase the prototype is being developed.
- The risk analysis would report the identification and effects of the present risks in the prototype.
- Assessments of risks at every rotation of the spiral evolution reduce the problems and the time required for solving at the final stage.
- The risk analysis strategies have to be stringent enough such that even small errors are not left out.
- Prototyping, incremental and premature risks detection proves this to be an efficient method over the other process models.
- Elimination of risks occurs in next iteration.
- The feedback from the customers is noted for updating the next prototype.
- The final prototype is developed after stated suggestions are subjected to risk analysis process.
- Construction and release also happens in iterative fashion based on the base prototype developed.
- At every level it is also being strengthened and the final level is the final code to be delivered to the customer.

Software process models

Evolutionary software process models

Spiral Model



Software process models

The Incremental development model

- The incremental model combines elements of the linear sequential model (applied repetitively) with the iterative philosophy of prototyping.
- Each linear sequence produces a deliverable “increment” of the software.
- When an incremental model is used, the first increment is often a core product.
- That is, basic requirements are addressed, but many supplementary features (some known, others unknown) remain undelivered.
- The core product is used by the customer (or undergoes detailed review).
- As a result of use and/or evaluation, a plan is developed for the next increment.
- The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality.
- This process is repeated following the delivery of each increment, until the complete product is produced.

Software process models

The Incremental development model

- This approach interleaves the activities of specification, development, and validation.
- The system is developed as a series of versions (increments), with each version adding functionality to the previous version.
- Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed.
- Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities.
- Incremental software development, which is a fundamental part of agile approaches, is better than a waterfall approach for most business, e-commerce, and personal systems.
- Incremental development reflects the way that we solve problems.
- We rarely work out a complete problem solution in advance but move toward a solution in a series of steps, backtracking when we realize that we have made a mistake.
- By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed.

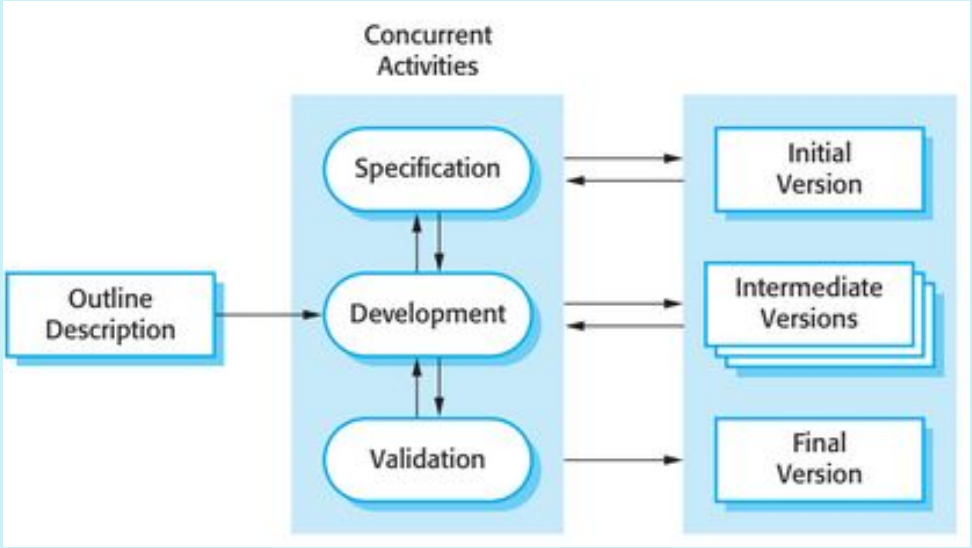
Software process models

The Incremental development model

- Incremental development in some form is now the most common approach for the development of application systems.
- This approach can be either plan-driven, agile, or, more usually, a mixture of these approaches.
- In a plan-driven approach, the system increments are identified in advance; if an agile approach is adopted, the early increments are identified but the development of later increments depends on progress and customer priorities.
- From a management perspective, the incremental approach has two problems:
    - The process is not visible. Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
    - System structure tends to degrade as new increments are added. Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

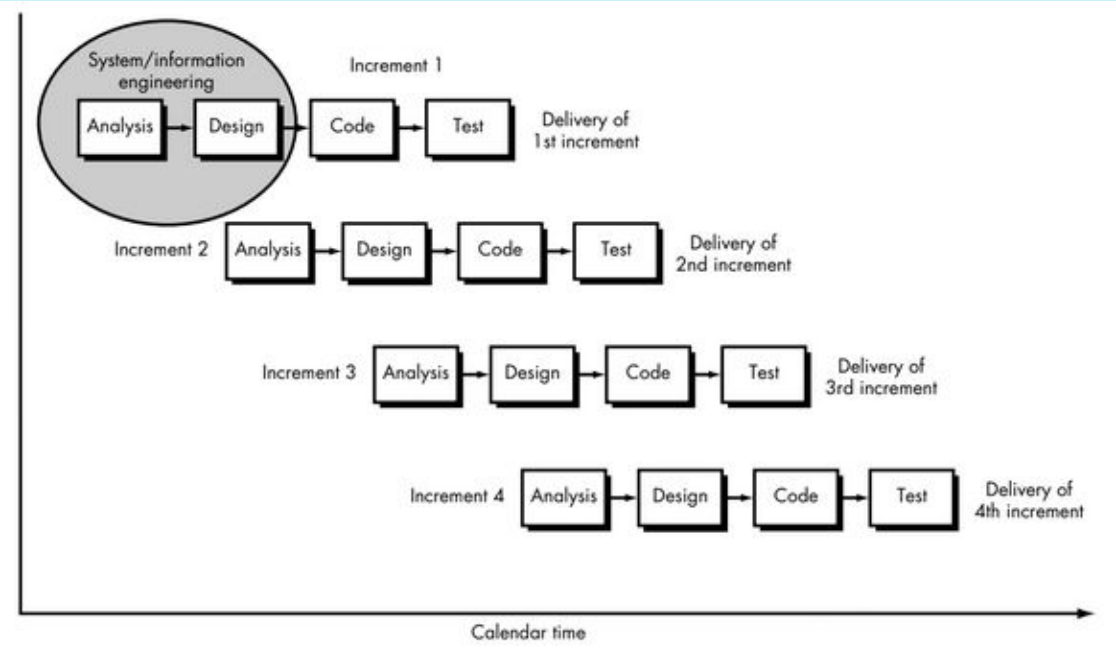
Software process models

The Incremental development model



Software process models

The Incremental development model



Software process models

The Incremental development model

- Each increment of the system incorporates some of the functionality that is needed by the customer.
- Generally, the early increments of the system include the most important or most urgently required functionality.
- This means that the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required.
- If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.
- Incremental development has important benefit:
  - The cost of accommodating changing customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
  - It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. Customers find it difficult to judge progress from software design documents.
  - More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Software process models

The Incremental development model

Advantages of the Model

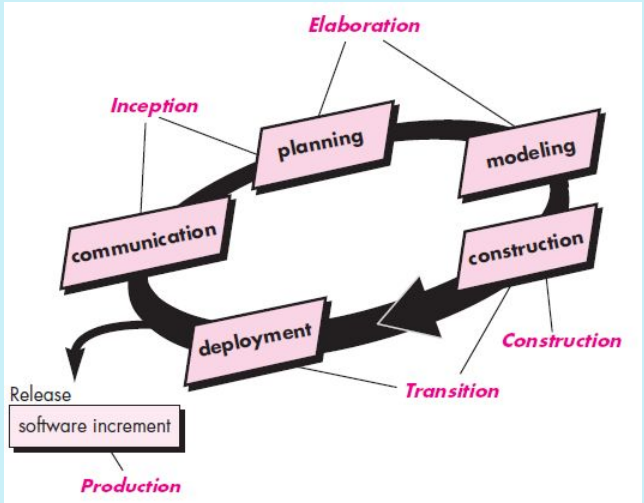
- 1. Quick feedback loop from business stakeholders
- 2. Focus on customer feedback
- 3. Customer feels the product early

Disadvantages of the Model

- 1. Scheduling of development is difficult with incremental model
- 2. Although the final product is given in pieces, tracking and monitoring is difficult
- 3. Testing needs to be exhaustive for each part

The unified process

- The Unified Process recognizes the importance of customer communication and streamlined methods for describing the customer's view of a system.
- It suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern software development.
- Jacobson, Rumbaugh, and Booch developed the Unified Process, a framework for object-oriented software engineering using UML.



The unified process

- **The inception phase** of the UP encompasses both customer communication and planning activities.
- By collaborating with stakeholders, business requirements for the software are identified; a rough architecture for the system is proposed; and a plan for the iterative, incremental nature of the ensuing project is developed.
- Fundamental business requirements are described through a set of preliminary use cases that describe which features and functions each major class of users desires.
- Architecture at this point is nothing more than a tentative outline of major subsystems and the function and features that populate them.
- Later, the architecture will be refined and expanded into a set of models that will represent different views of the system.
- Planning identifies resources, assesses major risks, defines a schedule, and establishes a basis for the phases that are to be applied as the software increment is developed.

The unified process

- **The elaboration phase** encompasses the communication and modeling activities of the generic process model .
- Elaboration refines and expands the preliminary use cases that were developed as part of the inception phase and expands the architectural representation to include 5 different views of software— use case, requirements, design, implementation, and deployment model.

- **The construction phase** of the UP is identical to the construction activity defined for the generic software process.
- Using the architectural model as input, the construction phase develops or acquires the software components that will make each use case operational for end users.
- To accomplish this, requirements and design models that were started during the elaboration phase are completed to reflect the final version of the software increment.
- All necessary and required features and functions for the software increment (i.e., the release) are then implemented in source code.
- As components are being implemented, unit tests are designed and executed for each.
- In addition, integration activities are conducted.
- Use cases are used to derive a suite of acceptance tests that are executed prior to the initiation of the next UP phase.

The unified process

- **The transition phase** of the UP encompasses the latter stages of the generic construction activity and the first part of the generic deployment (delivery and feedback) activity.
- Software is given to end users for beta testing and user feedback reports both defects and necessary changes.
- In addition, the software team creates the necessary support information (e.g., user manuals, troubleshooting guides, installation procedures) that is required for the release.
- At the conclusion of the transition phase, the software increment becomes a usable software release.

- **The production phase** of the UP coincides with the deployment activity of the generic process.
- During this phase, the ongoing use of the software is monitored, support for the operating environment (infrastructure) is provided, and defect reports and requests for changes are submitted and evaluated.

Agile

- In the agile model, the requirements are decomposed into many small parts that can be incrementally developed.
- The agile model adopts an iterative approach.
- Each incremental part is developed over an iteration.
- Each iteration is intended to be small and easily manageable and lasting for a couple of weeks only.
- At a time, only one increment is planned, developed, and then deployed at the customer site.
- Agile model emphasise face-to-face communication over written documents.
- It is recommended that the development team size be deliberately kept small (5–9 people) to help the team members meaningfully engage in face-to-face communication and have collaborative work environment.
- It is implicit then that the agile model is suited to the development of small projects.
- However, if a large project is required to be developed using the agile model, it is likely that the collaborating teams might work at different locations.
- Agile development projects usually deploy pair programming.
- In pair programming, two programmers work together at one work station.
- One types in code while the other reviews the code as it is typed in.
- The two programmers switch their roles every hour or so.

Agile

- Probably the best-known agile method is
  - Extreme programming (Beck, 1999; Beck, 2000)
  - Scrum (Cohn, 2009; Schwaber, 2004; Schwaber and Beedle, 2001),
  - Crystal (Cockburn, 2001; Cockburn, 2004),
  - Adaptive Software Development (Highsmith,2000),
  - DSDM (Stapleton, 1997; Stapleton, 2003), and
  - Feature Driven Development (Palmer and Felsing, 2002).
- The success of these methods has led to some integration with more traditional development methods based on system modelling, resulting in the notion of agile modelling (Ambler and Jeffries, 2002) and agile instantiations of the Rational Unified Process (Larman, 2002).
- Although these agile methods are all based around the notion of incremental development and delivery, they propose different processes to achieve this.

Agile

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Extreme programming (XP)

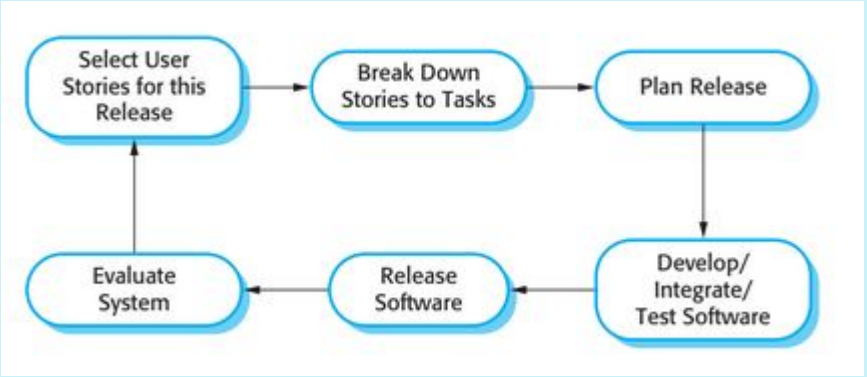
- XP is an process model under the agile umbrella and was proposed by Kent Beck in 1999.
- The name of this model reflects the fact that it recommends taking the best practices that have worked well in the past in projects to extreme Levels.
- XP is based on frequent releases (iterations), during which the developers implement “user stories”.
- User stories are similar to use cases, but are more informal and are simpler.
- A user story is the conversational description by the user about a feature of the required System.
- XP prescribes several basic activities to be part of the software development process.
- **Coding**: XP argues that code is the crucial part of any system development process, since without code it is not possible to have a working system. Therefore, utmost care and attention need to be placed on coding activity. However, the concept of code as used in XP has a slightly different meaning from what is traditionally understood.
- **Testing**: XP places high importance on testing and considers it be the primary means for developing a fault-free software.
- **Listening**: The developers need to carefully listen to the customers if they have to develop a good quality software. Programmers may not necessarily have an in-depth knowledge of the specific domain of the system under development. On the other hand, customers usually have this domain knowledge. Therefore, for the programmers to properly understand the required functionalities of the system, they have to listen to the customer.

Extreme programming (XP)

- **Designing**: Without a proper design, a system implementation becomes too complex and the dependencies within the system become too numerous and it becomes very difficult to comprehend the solution. This makes any maintenance activity prohibitively expensive. A good design should result in elimination of complex dependencies within the system. With this intention, effective use of a suitable design technique is emphasised.
- **Feedback**: It espouses the wisdom: “A system staying isolated from the scrutiny of the users is trouble waiting to happen”. It recognizes the importance of user feedback in understanding the exact customer requirements. The time that elapses between the development of a version and collection of feedback on it is critical to learning and making changes. It argues that frequent contact with the customer makes the development effective.
- **Simplicity**: A corner-stone of XP is the principle: “build something simple that will work today, rather than trying to build something that would take time and yet may never be used”. This in essence means that attention should be focused on specific features that are immediately needed and making them work, rather than devoting time and energy on speculations about future requirements.

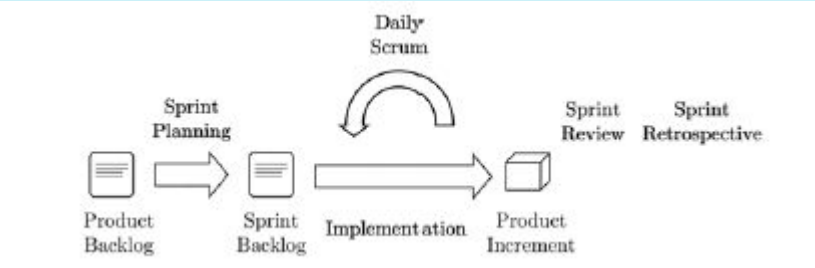
Extreme programming (XP)

Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.



SCRUM

- Scrum has relatively short iterations, called sprints, that take no more than a month.
- A product grows in increments, with each sprint resulting in a potentially deliverable version of the product.
- Scrum does not specify how the product is built and how requirements are elicited.
- In 1995, Scrum is the work of Ken Schwaber and Jeff Sutherland.
- The name Scrum comes from the sport of rugby, where a scrum is called to regroup and restart play.
- Both Schwaber and Sutherland participated in the 2001 meeting where the term agile was coined.
- Arrows represent the flow of events.
- A sprint begins at the left, with the current product backlog of potential work items.
- Think of the product backlog as a “wish list” of requirements items.



SCRUM

- The first event is sprint planning.
- The purpose of sprint planning is twofold:
  - To craft a goal for the work to be done during the sprint; and
  - To select work items for implementation during the sprint.
- The work items are selected from the product backlog and are called the sprint backlog.
- The arrow labeled implement represents the work to develop the sprint backlog.
- The circular arrow represents an event called a daily scrum.
- A daily scrum is a brief (15- minute) review of the work being done by the developers.
- During the event, developers review what was done the previous day and what is planned for the given day.
- The output from the development work is called a product increment; it consists of the functionality completed in this sprint.
- The product increment is inspected during a sprint review with stakeholders.
- The sprint ends with a sprint retrospective to explore process improvements, in readiness for the next sprint.

SCRUM

- Scrum is a framework, not a complete process.
- A framework specifies some aspects of a process and leaves other aspects to be filled in for a given project.
- Scrum specifies events and lets developers decide how they implement work items in a sprint backlog.
- Scrum defines three roles,Together, they form a scrum team.
    - **Product Owner** The product owner is responsible for the content of the product. The owner must be a single person; ownership is not to be spread across team members. In all team events, the product owner serves as the voice of the customer and sets priorities for what the team implements.
    - **Developers** Developers have sole responsibility for implementation. The development team comes up with estimates for the time and effort needed to implement a work item.
    - **Scrum Master** The scrum master acts as coach, facilitator, and moderator, responsible for arranging all events and for keeping them focused and on time. The scrum master guides people by highlighting the purpose and ground rules of an event, not by telling people how to do their jobs. An additional responsibility is to remove any external impediments for the team.

SCRUM

- Scrum defines four kinds of time-boxed events:
  - sprint planning,
  - Daily scrums,
  - sprint reviews, and
  - sprint retrospectives.

- Time-boxed means that the time interval of each event is fixed.
- All of these events occur during a sprint;One purpose of Scrum events is to plan, review, and steer development work.
- The boxes show the activities associated with the events.
- Another reason for specified or scheduled events is to foster communication within the scrum team.
- The events are time-limited, to keep down the time spent in meetings.
- Sprint Planning is to set a sprint goal and to select a sprint backlog.
- The product owner represents customer needs and priorities.
- The development team provides estimates for what it can accomplish during the sprint.
- Sprint planning is time-limited to eight hours or less for a one-month sprint.
- The attendees are the entire scrum team.

SCRUM

Sprint Goal

- The product owner proposes a goal for the functionality to be implemented during the sprint.
- The owner leads the entire scrum team in converging on the sprint goal.
- Once set, the sprint goal may not be changed during the sprint.

Sprint Backlog

- Given the sprint goal, the development team has sole responsibility for selecting product-backlog items to achieve the goal.
- The developers are accountable for explaining how completion of the selected items during the sprint will achieve the sprint goal.
- During a sprint, the developers may renegotiate with the product owner what they can accomplish.

Daily Scrum

- During a sprint, a daily scrum is a short 15-minute meeting to regroup and replan development work for the day.
- The attendees are the scrum master and the development team.
- The scrum master facilitates the meeting.

SCRUM

- Each developer addresses three questions:
  - What did I do yesterday toward the sprint goal?
  - What will I do today?
  - Are there any impediments to progress?

- These questions keep the whole development team informed about the current status and work that remains to be done in the sprint.
- The scrum master takes responsibility for addressing any impediments that are external to the scrum team.
- Daily scrums are also known as daily stand-ups because the attendees sometimes stand (to encourage observance of the 15-minute time limit).
- The 15-minute time limit works for small teams.
- The process can be scaled to larger teams by creating smaller subteams responsible for subsystems.
- The daily scrum for the larger team is then a scrum of scrums, with representatives from the subteams.
- The representatives are typically the scrum masters of the subteams.

SCRUM

- **Sprint Review** is an at most four-hour meeting at the end of a one-month sprint.
- The purpose of the review is to close the current sprint and prepare for the next.
- The review includes stakeholders and the whole scrum team.
- Closing the current sprint includes a discussion of what was accomplished, what went well, and what did not go well during the sprint.
- The development team describes the implemented functionality and possibly gives a demo of the new functionality.
- A sprint review sets the stage for the planning meeting for the next sprint.
- Preparing for the next sprint is like starting a new project, building on the current working software.
- The product owner updates the product backlog based on the latest understanding of customer needs, schedule, budget, and progress by the development team.
- The group revisits the priorities for the project and explores what to do next.
- **Sprint Retrospective** is an at most three-hour meeting for a one-month sprint.
- In the spirit of continuous improvement, the purpose of the retrospective is for the scrum team to reflect on the current sprint and identify improvements that can be put in place for the next sprint.
- The improvements may relate to the product under development, the tools used by the team, the workings of the team within the rules of Scrum, or the interactions between team members.

SCRUM

- Scrum specifies three artifacts that are visible to all team members to ensure that everyone has access to all available information.
- **Product Backlog** is an evolving prioritized list of items to be implemented. It is maintained by the product owner. It represents the requirements for the product, based on stakeholder needs, market conditions, and the functionality that has been implemented during previous sprints. At the end of each sprint, the product backlog may be updated, during the sprint review with stakeholders.
- **Sprint Backlog** consists of the work items to be implemented during the current sprint. The work items are selected from the product backlog during sprint planning, by the development team. The sprint goal drives the selection of work items for the sprint backlog. The goal cannot be changed during the sprint. The sprint backlog may, however, be changed during a sprint. Based on progress on the work items and any new information that comes to light, the developers may renegotiate the sprint backlog. Only the developers may change the sprint backlog, and the changes must preserve the sprint backlog's focus on the sprint goal. Any requirements changes have to wait until the next sprint planning event.
- **Product Increment** As with any iterative process, the product evolves with every sprint. The deliverable from a sprint is called a product increment. The added functionality in the product increment is driven by the sprint goal. The product increment is reviewed with stakeholders during the sprint review. The intent is to have a potentially releasable product at the end of each sprint.

Project Initiation management

- Project management involves application of knowledge, skills, tools, and techniques to control project activities so as to meet project requirements; thus, it converts the abstract into the concrete or an organization's vision to reality.
- The project management life cycle has five distinct phases
  - 1. Initiation
  - 2. Planning
  - 3. Execution
  - 4. Control
  - 5. Closure

- Based on the domain and technology, the lifecycle model differ, but it is always performed within the 5 stages.

-Project Initiation management

-Project charter

- Most projects start on a high note. Stakeholders have high hopes. Accordingly, project charters are made.
- Unfortunately, as the project progresses, all the enthusiasm vanishes quickly.
- The project stakeholders need to set their expectations with grounded realities.
- All their hopes should be aligned with practical limitations and achievable goals.
- If this is not done right from the inception of the project, the project is going to falter all the way.
- The project charter should include things like project goals, project objectives, major responsibilities allocation, etc.
- But a simple project charter may be a simple statement from the top management
- The project charter is the place where a big picture of the effort, even beyond the project, is captured.
- For instance, say, the project is part of a product development effort in which the product is being developed incrementally.
- The product development consists of many small projects for which a small set of features are being developed and added into the product each time a project gets completed.
- The project charter will capture information for the entire effort to build the ultimate product through these small projects, and in fact, during all of these projects, the project charter may remain the same with not many changes.

-Project Initiation management

-Project charter

- Similarly, the project charter should also include the business goals for which the software project is being initiated, and also state that the software project will help in achieving those business goals.

Table 2.1 Sample Project Charter

The project will provide a cutting edge software solution to our sales team to provide excellent customer service for our customers so that all customer issues can be solved within 24 hours of lodging of a complaint.

Project Initiation management

-Project scope

- Project scope management involves the processes required to identify and ensure that a project includes all the work and only the work required for successful project completion.
- It is primarily concerned with identification and control of the work, that is, what is and what is not a part of project requirements.
- Project failure is related to incorrect understanding of project requirements and lack of control on the already agreed upon project scope.

Types of Scopes

- **Product scope:** It refers to unalterable features and functions that are to be included in a product or service.Product scope is validated against the project requirements
- **Project scope:** It refers to the work that must be done to deliver a product with specified features and functions; project scope is validated against the project management plan.

Product Scope	Project Scope
Features and functions characterize a product or service	Work that must be done to deliver a product
Required processes, tools, and techniques vary by application area	Required processes, tools, and techniques are similar across application areas
Defined in a project life cycle	Defined in a project management life cycle
Completion of product scope is validated by measurement against product requirements	Completion of project scope is validated by measurement against project management plan

Project Initiation management

-Project scope

- After analyses of failed projects, it has been found that most of the projects fail because of an increase in the scope of the project over time.
- An increase in project scope happens primarily due to two factors.
- One factor is that as the project progresses and features are being built in the application, the user community, after seeing the partially made application, may feel that some additional functionality is also needed to do their job using the newly built application. So they keep making change requests throughout the development cycle. This not only disrupts development activity, it also makes the application susceptible to defects. But the most important impact over the project is the increase in the volume of the project work], which results in the escalation of costs and an elongation of the schedule.
- The other factor that results in change in the project scope is a poor requirements definition. A poor understanding of requirements or a poor definition of requirements leads to changes required later on in the software design or software build to rectify this problem. In any case, the project scope increases due to these factors.
- To deal with scope creep, it has to be ensured that the requirements are lucid and clear from the very start so that project effort estimation and project schedule are accurate.
- If any changes are to be made in requirements, then there should be a proper change request mechanism that will identify the impact of the change on the project and this should be communicated to the stakeholders.

Project Initiation management

-Project scope

- All these aspects should be clearly defined during the initiation stage itself.
- There is one more aspect about project scope, apart from the volume of work, in terms of the number of features.
- It is the fact that the software product to be produced needs to have a specific level of quality.
- This level of quality needs to be frozen during the project initiation phase.
- Definitely, such a software system needs to be of very high quality in terms of reliability, security, correctness, and efficiency.
- A high level of quality for such a software system translates into high effort required for building this application.
- So, a combination of a number of features and the quality level determines the total volume of work.
- It is very important, early on in the project, to clearly lay out these aspects so that the volume of work can be determined.

Table 2.2 Sample Project Scope Definition

The project will be delivered within 15 months from the date of start of the project. The software product that will be made through this project will have features for customer complaint logging, issue resolution, and issue closure. The software product should have the capability of supporting our customer base of 10,000, who will be using the service through an Internet connection by logging into our web portal.

Project Initiation management

-Project objectives

- The project should have a set of well-defined objectives that must be met.
- If any of these objectives are not met upon completion of the project, then the project will be considered to be a failure.
- The stakeholders state and set the project objectives.
- The objectives should be stated in clear language and the set of objectives should be kept as small as possible.
- If clear project objectives are set at the project initiation, it would help the project team to understand the importance of the project and will help the team to do its best to achieve the goals.

Table 2.3 Sample Project Objectives

The organization will be able to increase customer satisfaction to 99.5% from the existing level of 92%. This will help in reducing customer attrition, increasing repeat business from existing customers, and enhancing our brand value.

Project Initiation management

-Practical Considerations

- **Project size**: It is the single-most important factor that makes the approach to handling one project different from another. Smaller projects need less formal project management than the larger ones.

- **Product quality**: If the software product to be made requires stringent quality measures, then an elaborate quality control mechanism will be required throughout the project process to ensure that defects are prevented in the product at each stage of development. On the other hand, if the software product to be made does not need stringent quality norms, then a cursory quality control mechanism will be enough.

- **Technology**: Technology plays an important role in determining productivity on any project. If the platform is older technology(client server) and the programming language (Ada), then the project effort will be considerably more than if a newer and more productive technology, say Java, is used.

- **Code reuse:** Code reuse can considerably reduce the required project effort. So, the effort on two projects will be very different if one project code reuse has been extensively used, compared to some other project where code reuse has not been used.

Project Initiation management

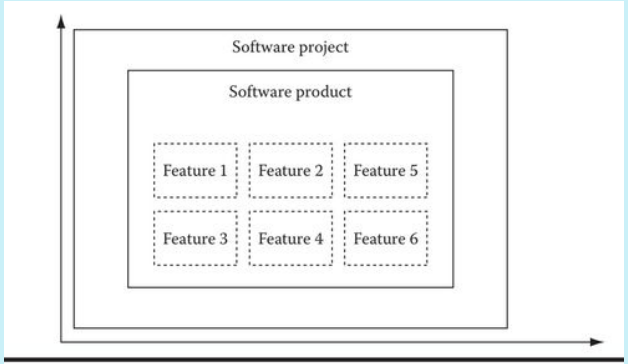
Estimate Initial Project Size

- At the project initiation stage, a rough project size should be estimated so that a sketch of the initial project plan can be realized.
- From the initial requirements (as available in a Request for Information quote), a rough design estimate can be made.
- The rough design can include details about how the product can be broken down into parts.
- These parts can be sized from estimating, either the estimated number of lines of code required to build them or by using an estimated number of function points.
- After the size of each part or module is determined this way, the complete size of the software product can be determined.
- Since at the initiation stage detailed information about project parts is not clear, the estimate of the product size is also rough.
- However, this can be taken as a starting point, and the product size estimate can be refined as the project progresses.
- Especially on outsourced projects, a rough product size estimate should be made during the initiation phase so that a general idea about the project can be made and passed on to the stakeholders.
- This information will be helpful for them to make crucial decisions about the project.

Project Initiation management

Estimate Initial Project Effort and Costs

- Initial project cost estimates can be determined from the productivity of the members of the project team, the effort estimate, the number of hours put in by software professionals, and the prevailing hourly rate of software professionals who will be working on the project as project team members.
- The cost of the project is one of the most important considerations of stakeholders.
- If the project is going to cost more than they had anticipated and budgeted for, then most probably, the project will be called off.
- In some cases, the stakeholders may agree to a reduction in the size of the project so as to reduce the cost of the project.

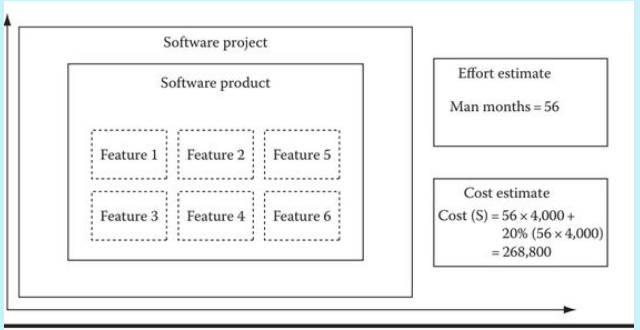


Initial software product and project size estimate.

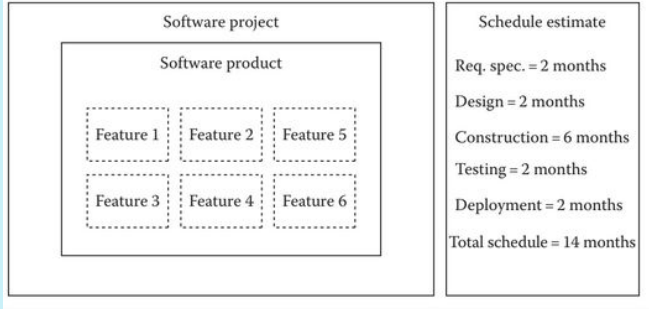
Project Initiation management

Estimate Initial Project Schedule

- Like project cost, the project schedule is one of the most critical aspects of the project.
- Stakeholders may have the objective of gaining a marketing edge over the competition by implementing the proposed software application.
- Many of such objectives are time sensitive, and the stakeholders may like to see the new system implemented before a specified date.
- During project initiation discussions, stakeholders may ask the project manager to reduce the project schedule that has been presented to them, even if project costs rise because of this.
- In such cases, the project manager may have to adjust his project schedule to suit the needs of the stakeholders.
- He will have to adjust his project plan, resource allocation, etc., accordingly.



2 Initial software project effort and cost estimate.



Initial software project schedule estimate.

Project Initiation management

- Project initiation most often happens with a kick off meeting involving the project manager,the stakeholders, and some key project members.
- They define the project charter, project scope, and project objectives.
- A preliminary effort and cost estimate is chalked out.
- A preliminary sketch is also made for the project schedule so that a tentative duration for the project can be established.
- At the initiation stage, everything about the project is tentative.
- But the goal is to see if the project is itself feasible or not.
- For this purpose, a feasibility study can also be conducted in case the confidence level for the project is still uncertain.
- If the project is found not viable after the feasibility study, it can be abandoned.
- Abandoning an unfeasible project at this stage is less costly than abandonment after investing large sums of money and effort.
- In cases when it is felt that the requirements from customers are not clear or complete, then the project can be split so that the requirements can be made clear and complete in the first phase of the project.
- In the second phase of the project, the software product can be built on the basis of complete customer requirements.