Introduction To Software Engineering

Syllabus

Software Engineering-process framework, Capability Maturity Model (CMM), Advanced Trends in Software Engineering

1.1 Nature of Software

- The nature of software has great impact on software engineering systems.
- The general nature of software may describe the important characteristic : **Reliability :** If we consider the use of software in air traffic control system and space shuttle, then there should not be any chances of failure of software.
- In these examples, if reliability is not taken into consideration, then the human life is at risk.
- In addition to this, there are four other important characteristics that describe the nature of software :

1. Absence of fundamental theory
2. Ease of change
3. Rapid evolution of technologies
4. Low manufacturing cost

1.1.1 Absence of Fundamental Theory

- If we consider the example of physics, we see there are fundamental laws of physics, but in software there are no such fundamental laws despite the researches done by the computer scientists.
- Because of this drawback, it is very difficult to do any reasoning about the software until it is developed. Thus the developers practice few software engineering standards that are not foolproof. But the codes written for developing software are following the discipline and some solid principles.

1.1.2 Ease of Change

- The software has the provision to be altered at any stage of time. Thus the software development organizations take the advantage of this feature. From the customer's point of view, the change is always required throughout its development cycle and even after its delivery to the customer.

- Since there are no basic rules of software, there is no rule available to accommodate the changes and its impact until it is developed. Thus we can say that the ease of change is a gift of God to the developers and the development organizations.

1.1.3 Rapid Evolution of Technologies

- In the modern age, the software development technologies and development environments are changing rapidly with an extreme speed.
- It becomes the need of the time to keep the software engineers updated and armed with latest technologies and skills.
- The software engineering standards must also be revised with the technology evolutions.

1.1.4 Low Manufacturing Cost

- The cost of software reproduction and installation is less as compared to a new development. Today nearly 80 percent of the software development contains only maintenance and only 20 percent is new development. This reflects that manufacturing a product is considerably involves very low cost.
- The software reusability is an important characteristic that benefits the development organizations a lot in manufacturing the software at low cost.

1.2 Software Definition

Q. Define 'Software Engineering' MU - Dec. 15, Dec.19, 1 Mark

- Computer software has become an integral part of our daily lives. It helps in all sorts of businesses and decision makings in business. The application of computer software includes : Telecommunication, transportation, military, medical sciences, online shopping, entertainment industry, office products, education industry, construction business, IT industry, banking sector and many more.
- The software applications have a great impact on our social life and cultural life as well. Due to its widespread use, it is the need of time to develop technologies to produce high quality, user friendly, efficient and economical software.
- Computer software is actually a product developed by software engineers by making use of various software engineering processes and activities.
- Software consists of data, programs and the related documents. All these elements build a configuration that is created as a part of the software engineering process. The main motive behind software engineering is to give a framework for building software with better quality.
- We can say that the software has become the key element in all computer based systems and products.

1.3 Software Engineering : A Layered Technology

Q. Explain in brief the software process framework. MU - Dec. 15, 4 Marks

- The term software engineering is defined as : "By using the principles of sound engineering and its establishment, software is developed that should be economical and should work efficiently on real machines."
- Software engineering is a systematic and disciplined approach towards developments of software. The approach must be systematic for operation of the software and the maintenance of it too.
- Software engineering is considered as a layered technology. These layers includes :

(1) Quality focus	(2) Process
(3) Methods	(4) Tools

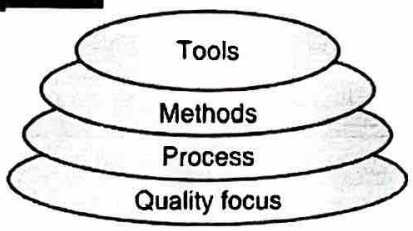


Fig. 1.3.1 : Software engineering layers

1.3.1 Quality Focus

Quality is nothing but **'degree of goodness'**. Software quality cannot be measured directly. Good quality software has following characteristics :

1. **Correctness** is degree to which software performs its required function.
2. **Maintainability** is an ease with which software is maintained. Maintenance of software implies change in software. If software is easy to maintain, then the quality of that software is supposed to be good.
3. **Integrity** is a security provided so that unauthorized user can not access data or information. e.g. password authentication.
4. **Usability** is the efforts required to use or operate the software.

1.3.2 Process

Software process defines a framework that includes different activities and tasks. In short, process defines following **'what'** activities and tasks for software development :

1. What activities are to be carried out ?
2. What actions will be taken ?
3. What tasks are to be carried out in a given action ?

1.3.3 Methods

Method provides technical way to implement the software i.e. **'how to'** implement. Methods consist of collection of tasks that include :

1. **Communication :** Between customer and developer.
2. **Requirement analysis :** To state requirements in detail.
3. **Analysis and design modelling :** To build a prototyping model of software to exhibit the requirements clearly.

4. **Program construction :** Implementation of requirements using conventional programming languages or automated tools.
5. **Testing and support :** Test for errors and expected results.

1.3.4 Tools

Software tool is an automated support for software development.

For example

1. Microsoft front page or Microsoft Publisher can be used as web designing tool.
2. Rational Rose can be used as object oriented analysis and design tool.

1.4 The Characteristics of Software

- Software is having some characteristics, those are totally different from hardware characteristics or the industrial manufacturing
- Software is developed or engineered and it is not manufactured like other hardware products.
- There exist some similarities between development of software and manufacturing of hardware
- In both the cases quality is achieved by good design but manufacturing phase of hardware can introduce quality problems that are absent in software.
- Both activities depend on people but relationship between people applied and work done is different in b cases.
- Both the cases require the 'construction of product', but approaches are different.

Software does not wear out.

- Note that, wear out means process of losing the material.
- Hardware components are affected by dust, temperature and other environmental maladies. Stated simply, hardware can wear out. However software does not influenced by such environmental means.
- When hardware component wear out, it can be replaced by spare part. But there are no spare parts for software. Any software error indicates error in design or coding of that software.
- Hence software maintenance involves more complexity than hardware maintenance.
- Mostly software is custom built rather than assembled from existing components.
- Computer hardware can be assembled from existing components as per our requirements. But that is not the case for software. Software is developed according to customer's need.

1.5 The Software Engineering Process Framework

- A software process can be illustrated by the following Fig. 1.5.1. In the Fig. 1.5.1, a common process framework is exhibited. This framework is established by dividing overall framework activities into small number of framework activities.

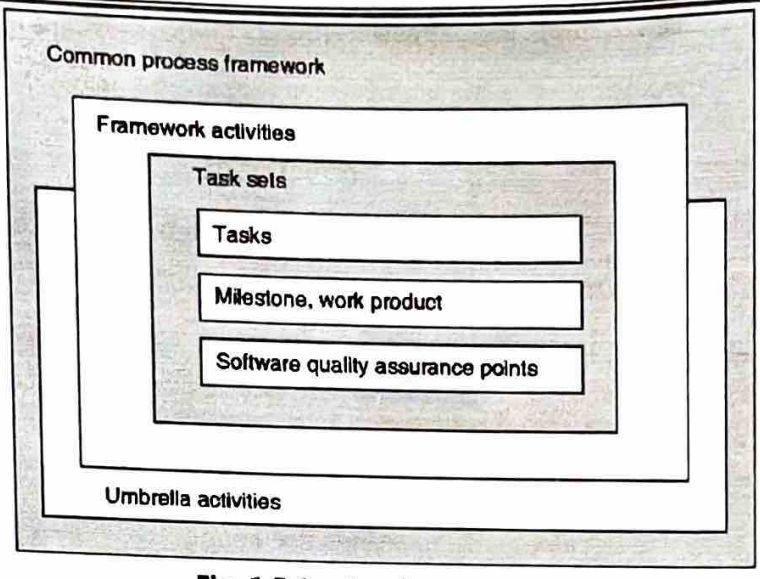


Fig. 1.5.1 : A software process

- And these small activities are applicable to the entire project irrespective of its size and complexity. A framework is a collection of task sets and these task sets consists of :
  - Collection of small work tasks
  - Project milestones, deliverable i.e. actual work product and
  - Software quality assurance points
- There are various activities called **umbrella activities** are also there and these activities are associated throughout the development process. The umbrella activities include :

(1) Software project tracking and control
(2) Risk management
(3) Software Quality Assurance (SQA)
(4) Formal Technical Reviews (FTR)
(5) Measurement
(6) Software Configuration Management (SCM)
(7) Reusability Management
(8) Work product preparation and production

- All these umbrella activities actually constitute the process model. Also the umbrella activities are independent and occur throughout the process

1.5.1 Umbrella Activities

Q. Explain Umbrella activities of software engineering. **MU - Dec. 19, 4 Marks**

The framework described in generic view of software engineering (Fig. 1.5.1 : A software process) is complemented by number of Umbrella activities. Typical **umbrella activities** are :

1. Software project tracking and control

- Developing team has to assess project plan and compare with predefined schedule.

- If project plan doesn't match with predefined schedule, necessary actions are taken to maintain schedule.

2. Risk management

Risk is event that may or may not occur. But if that event happens, it causes some unwanted outcomes. He proper management of risk is required.

3. Software Quality Assurance (SQA)

- SQA is nothing but planned and systematic pattern of activities those are required to give guarantee software quality.
- For example during software development, meetings are conducted in every stage of development to fi out defects and suggest improvement to yield good quality software.

4. Formal Technical Reviews (FTR)

- FTR is a meeting conducted by technical staff. The purpose of meeting is to detect quality problems ar suggest improvements.
- The technical staff focuses on quality from customer's point of view, i.e. what customer exactly wants.

5. Measurement

- It includes the efforts required to measure the software.
- Software can not be measured directly. It is measured by some direct measures (e.g. cost, lines of code, siz of software etc) and indirect measures (e.g. quality of software, which is measured in terms of other factors Hence it is an indirect measure of software.)

6. Software Configuration Management (SCM)

It manages the effects of change throughout the software process.

7. Reusability management

- It defines criteria for product reuse.
- If software components developed for certain application can be used in development of other applications then it's good quality software.

8. Work product preparation and production

It includes the activities required to create documents, logs, forms, lists and user manuals for develope software.

1.6 Capability Maturity Model (CMM)

Q. Explain the process of CMM. **MU - May 19, 10 Marks**

- The SEI Capability Maturity Model is a process meta-model developed by the Software Engineering Institute (SEI).
- It defines the process characteristics that should exist if an organization wants to establish a software process that is complete.
- The spirit of the SEI Capability Maturity Model should always be adopted. It argues that software development :
  - It must be taken seriously
  - It must be thoroughly.
  - It must be controlled uniformly.
  - It must be tracked accurately.
  - It must be conducted professionally.
  - It must focus on the needs of it's customers
- The SEI Capability Maturity Model presents two types of meta models:
  - As a Continuous model
  - As a Staged model

As a Continuous model

It describes the process in two dimensions as shown in Fig. 1.6.1. Each process area (e.g. project planning, requirement management) are assessed against specific goals and practices and is rated according to following capability levels :

1. **Level 0 : Incomplete**

   The process area (e.g., requirements management) is either not performed or does not achieve all goals and objectives defined by the SEI Capability Maturity Model for level 1 capability.

2. **Level 1 : Performed**

   All of the specific goals of the process area (as defined by the SEI Capability Maturity Model) have been satisfied. Work tasks required to produce defined work products are being conducted.

3. **Level 2 : Managed**

   All level 1 criteria have been satisfied. In addition, all work associated with the process area conforms to an organizationally defined policy; all people doing the work have access to adequate resources to get the job done.

4. **Level 3 : Defined**

   All level 2 criteria have been achieved. Also the process is tailored from organization's set of standard processes according to organization's guidelines.

5. **Level 4 : Quantitatively managed**

   All level 3 criteria have been achieved. Also the process is controlled and improved using measurements and quantitative assessment.

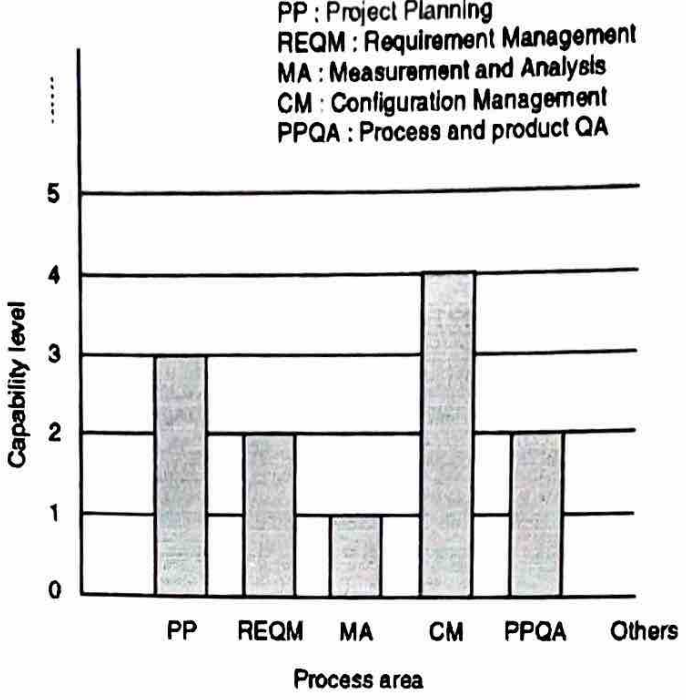


Fig. 1.6.1 : SEI Capability Maturity Model process area capability profile

6. Level 5 : Optimized

- All level 4 criteria have been achieved. Also the process area is adapted all standards to meet chang customer's needs.
- The SEI Capability Maturity Model defines each process area in terms of Specific Goals (SG) and Speci Practices (SP).
- For example, project planning is one of the process areas.

The Specific Goals (SG) and the associated Specific Practices (SP) defined for project planning are :

SG 1 : Establish estimates

SP 1.1 -1 Estimate the scope of the project

SP 1.2-1 Establish estimates of work product and task attributes

SP 1.3-1 Define project life cycle

SP 1.4-1 Determine estimates of effort and cost

SG 2 : Develop a Project Plan

SP 2.1 -1 Establish the budget and schedule

SP 2.2-1 Identify project risks

SP 2.3-1 Plan for data management

SP 2.4-1 Plan for project resources

SP 2.5-1 Plan for needed knowledge and skills

SP 2.6-1 Plan stakeholder involvement

SP 2.7-1 Establish the project plan

SG 3 : Obtain commitment to the plan

SP 3.1 -1 Review plans that affect the project

SP 3.2-1 Reconcile work and resource levels

SP 3.3-1 Obtain plan commitment

The SEI Capability Maturity Model also defines set of five Generic Goals (GG) and related Generic Practices (GP).For example : Generic Goals (GG) and related Generic Practices (GP) for project planning process area are :

GG 1 : Achieve specific goals

GP 1.1 Perform base practices

GG 2 : Institutionalize a managed process

GP 2.1 Establish an organizational policy

GP 2.2 Plan the process

GP 2.3 Provide resources

GP 2.4 Assign responsibility

GP 2.5 Train people

GP 2.6 Manage configurations

GP 2.7 Identify and involve relevant stakeholders

GP 2.8 Monitor and control the process

GP 2.9 Objectively evaluate adherence

GP 2.10 Review status with higher level management

GG 3 : Institutionalize a defined process

GP 3.1 Establish a defined process

GP 3.2 Collect improvement information

GG 4 : Institutionalize a quantitatively managed process

GP 4.1 Establish quantitative objectives for the process

GP 4.2 Stabilize sub process performance

GG 5 : Institutionalize an optimizing process

GP 5.1 Ensure continuous process improvement

GP 5.2 Correct root causes of problems

The staged SEI Capability Maturity Model defines the same process areas, goals, and practices as the continuous model. The primary difference is that the staged model defines five maturity levels, rather than five capability levels. To achieve a maturity level, the specific goals and practices associated with a set of process areas must be achieved.

1.7 Advanced Trends in Software Engineering

- It is style of software development which is focused on public availability and communication.
- Usually communication happens using internet.
- It is used in freeware, open source software and commons based peer production.
- It is also used in agile development model.
- It is generally used in development of free software.
- It is very compatible with free software.
- They meets online for software development.
- It is evolution of integrated development environment.

Typical functionalities are as follows :

- Version control system
- Bug tracking system
- To do list
- Mailing list
- Document management system
- Forum
- They also involve users in the collaborative development.
- It is used in most technological disciplines.

Model-driven development

- It is software design approach for development of software system.
- It provides guidelines for structuring of specifications.
- It is kind of domain engineering.
- It is launched by object management group.
- It defines system functionalities using platform independent model.

Related standards are as follows.

- Unified modeling language (UML)
- Meta object factory (MOF)
- XML metadata interchange (XMI)
- Enterprise distributed object computing (EDOC)
- Software process engineering meta model (SPEM)

Different tools are used in model driven architectures.

- Creation tools
- Analysis tools
- Transformation tools
- Composition tools
- Test tool
- Simulation tools
- Metadata management tools
- Reverse engineering tools
- Model driven development is shown in Fig. 1.7.1.

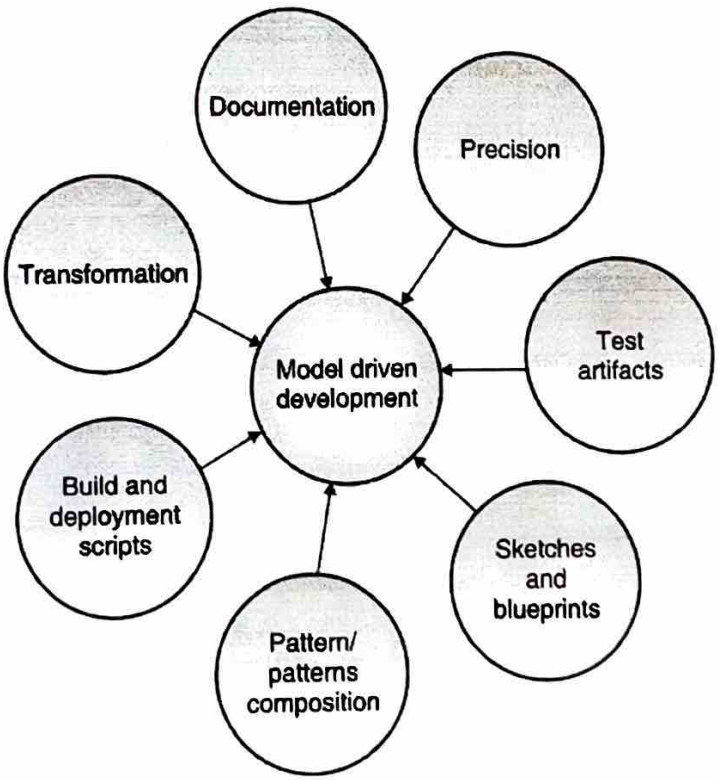


Fig. 1.7.1 : Model driven architecture

Test-driven development

- It is software development process which relies on repetition of very short development cycle.
- First, developer writes test cases which define a desired improvement.
- It is related to test first programming concept of extreme programming.
- Test driven development cycle is shown in Fig. 1.7.2
  1. Add test
  2. Run all tests
  3. Write some code
  4. Run tests
  5. Re - factor code
  6. Repeat the process.

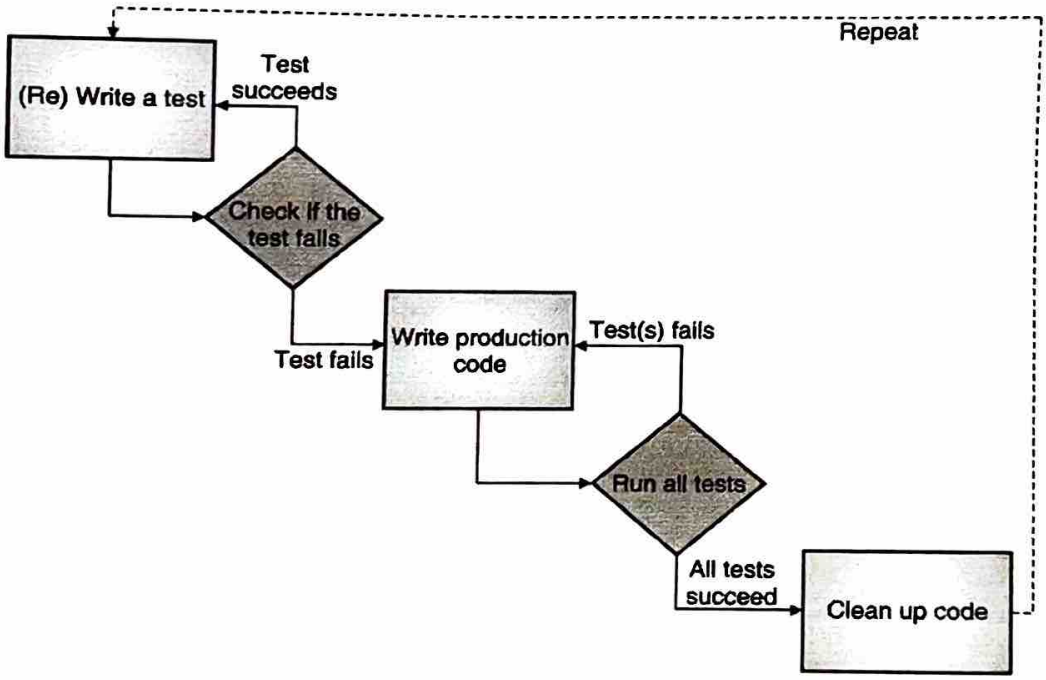


Fig. 1.7.2 : Test Driven development

Challenges of global software development

- Communication breakdown
- Coordination breakdown
- Control breakdown
- Cohesion barriers

Culture clash

Ability to gain market share.

Greater flexibility and variable staffing model.

Lower cost labor.

Access to broader set of skilled workers.

Ability to leverage outsourcing providers with specialized skill.

Possibility of establishing a presence in geographies that may become new market for product.

Ability to gain competitive edge.

Business driver and global delivery challenges are shown in Fig. 1.7.3.

usiness drivers and global delivery challenges

Business Drivers and Global Delivery Challenges

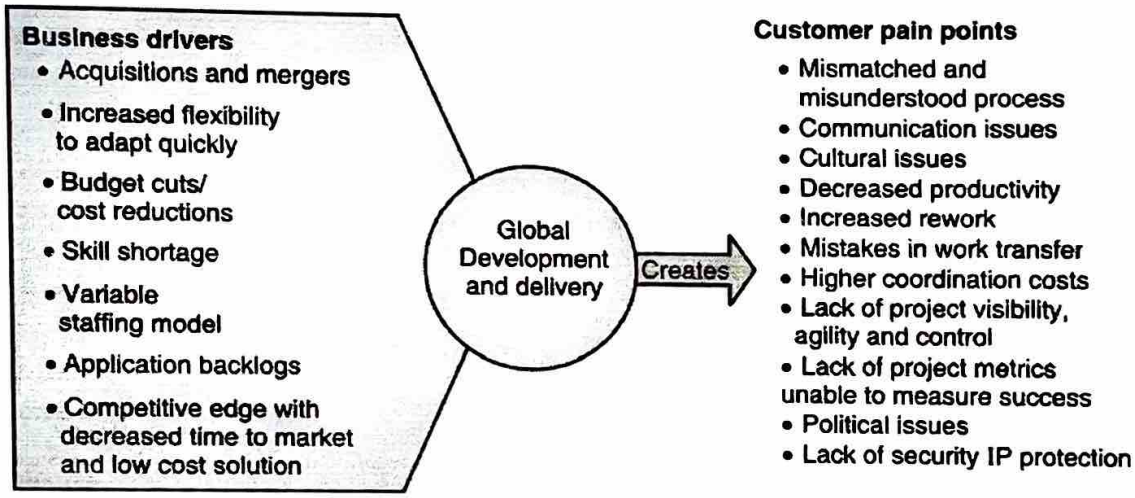


Fig. 1.7.3 : Business Driver and global delivery challenges

Misunderstood processes or mismatched processes.

Communication issues can lead to misunderstanding, omissions, errors and rework.

Review Questions

Q. 1 List and explain the characteristics that describe the nature of software.

Q. 2 Explain the term Computer Software in brief.

Q. 3 Explain the layered approach to software engineering.

Q. 4 Define Software Engineering. What are the various categories of software?

Q. 5 What are the software characteristics?

2 Process Models

Syllabus

Prescriptive Process Models : The Waterfall Model, Incremental Process Models, Evolutionary Process Models, RAD & Spiral

2.1 A Generic Process Model (or Generic Process Framework)

A software process is collection of various activities. There are five generic process framework activities :

(1) Communication (2) Planning (3) Modelling
(4) Construction (5) Deployment

2.1.1 Communication

- The software development process starts with communication between customer and developer.
- According to waterfall model customer must state all requirements at the beginning of project.

2.1.2 Planning

It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and tracking.

2.1.3 Modelling

- It includes detail requirement analysis and project design (algorithm, flowchart etc).
- Flowchart shows complete pictorial flow of program where as algorithm is step-by-step solution of problem.

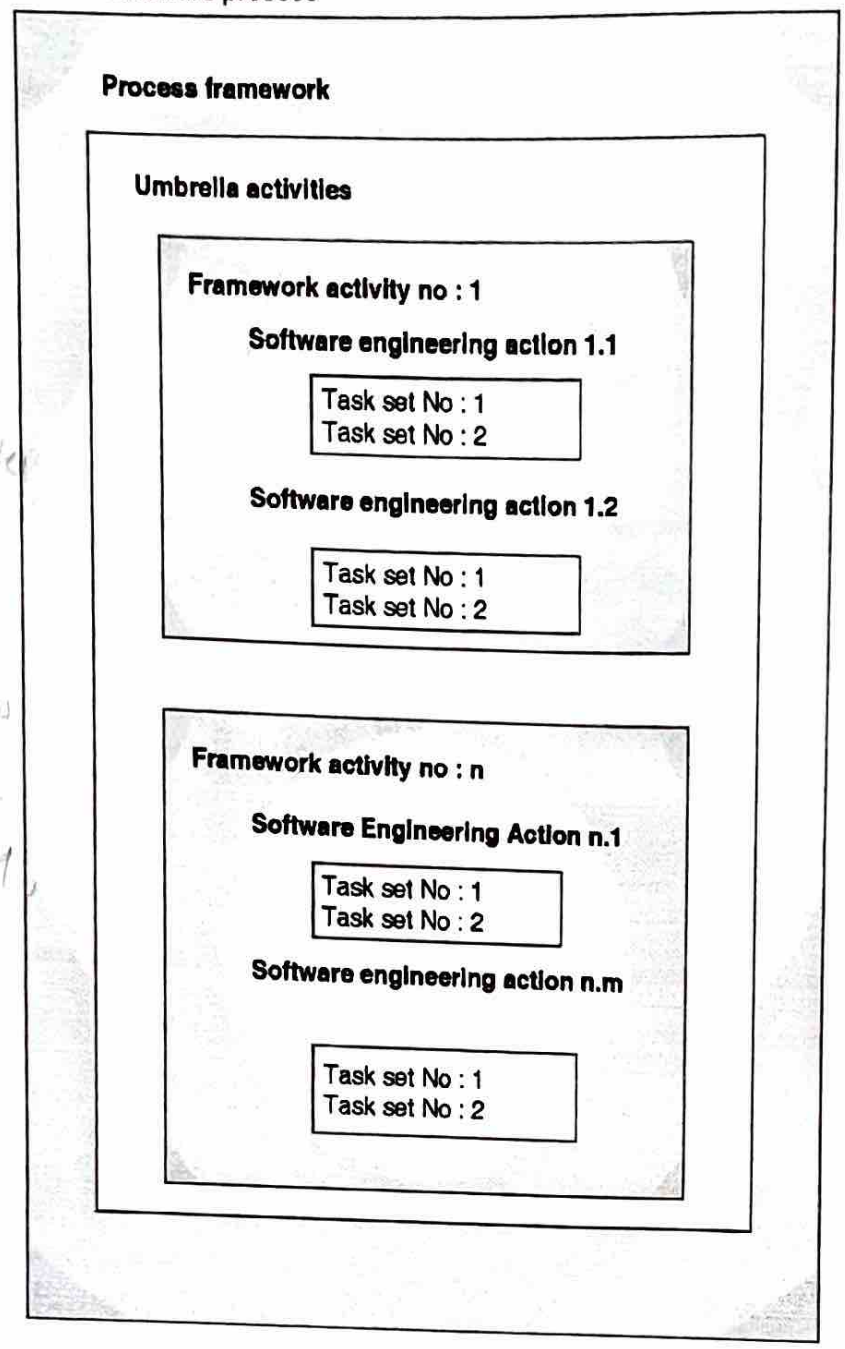
2.1.4 Construction

It includes coding and testing steps :

(i) Coding : Design details are implemented using appropriate programming language.

(ii) Testing : Testing is carried out for the following reasons :

- To check whether the flow of coding is correct or not.
- To check out the errors of program e.g. in C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables.
- To check whether program is giving expected output as per input specifications.



Umbrella activities
set of steps
or procedure
that the sw
eng teams follow
to maintain the
progress, quality,
change & risk
of the overall
development
tasks.

Fig. 2.1.1 : A software process framework

2.1.5 Deployment

- It includes software delivery, support and feedback from customer.
- If customer suggest some corrections, or demands additional capabilities, then changes are required for such corrections or enhancement.
- These five generic framework activities can be used for :
  - Development of small programs
  - Creation of large programs
  - Development of large system based programs
- In addition to these five generic framework activities, various umbrella activities are also associated throughout the development process.

2.2 Prescriptive Process Models

- All the software organizations describe a unique set of framework activities for their own development processes. In general, it should adopt all the generic framework activities and the set of tasks defined in Section 1.5 i.e. generic process model. Whatever process model is chosen by the organization, but it should encompass the following framework activities :

  (1) Communication cust - developer

  (2) Planning cost estimation & time line tracking

  (3) Modelling flow chart, Algo

  (4) Construction code, Testing

  (5) Deployment Delivery Support & feedback.

- The name 'prescriptive' is given since the model prescribes set of activities, actions, tasks, quality assurance and change control mechanism for every project. Each of the prescriptive models also prescribes a workflow.
- Workflow is defined as the flow of process elements and the manner in which they are interrelated. All the generic framework activities are defined earlier, but each of the prescriptive models put different emphasis to these generic framework activities and give different workflow.
- In the following section, we describe some of the prescriptive process models :

  (1) The waterfall model

  (2) Incremental process models

  (3) Evolutionary process models

  (4) The specialized process models

2.2.1 The Waterfall Model

Q. Differentiate in between waterfall and spiral model. **MU - May 19, 5 Marks**

- This model is also called as **'Linear sequential model'** or **'Classic life cycle model'**. Classic life cycle paradigm begin at system level and goes through analysis, design, coding, testing and maintenance.

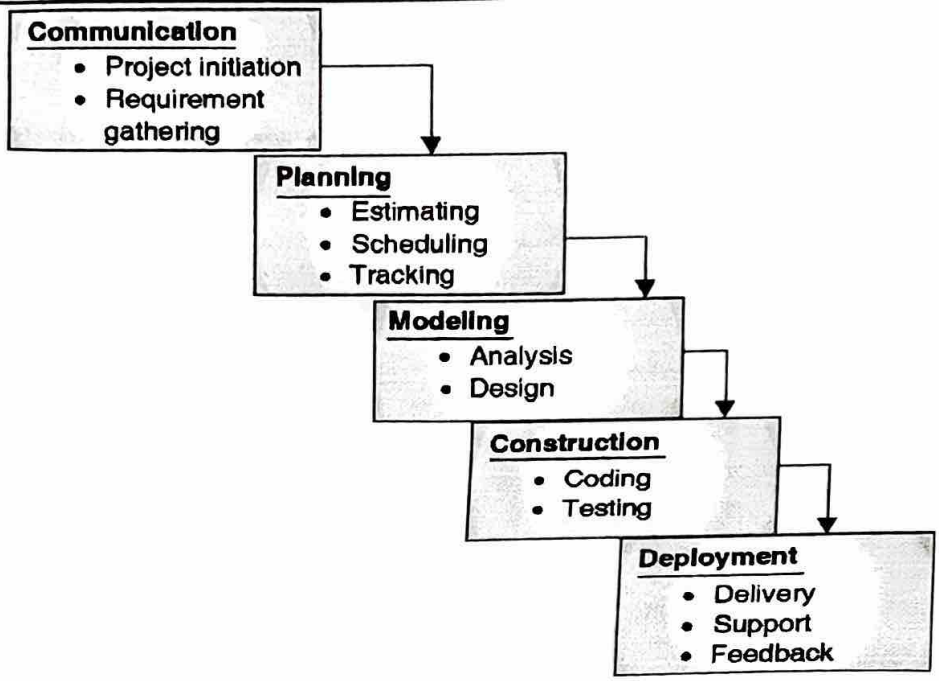


Fig. 2.2.1 : The Waterfall model

- An alternative design for 'Linear Sequential Model' is as shown in Fig. 2.2.2.

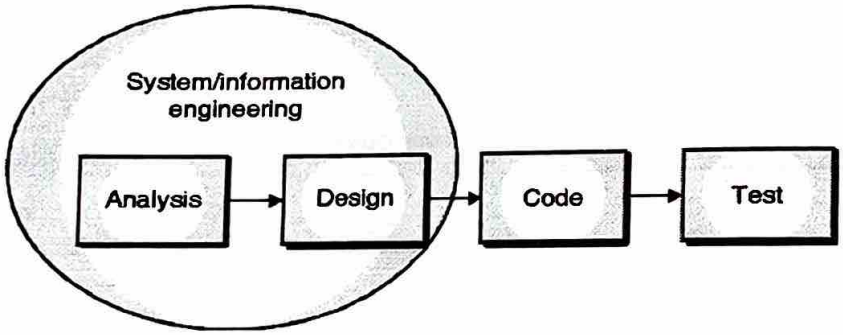


Fig. 2.2.2 : The linear sequential model

1. Communication

- The software development process starts with communication between customer and developer.
- According to waterfall model customer must state all requirements at the beginning of project.

2. Planning

It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and tracking.

3. Modelling

- It includes detail requirement analysis and project design (algorithm, flowchart etc).
- Flowchart shows complete pictorial flow of program whereas algorithm is step-by-step solution of problem

4. **Construction**

It includes coding and testing steps :

(i) **Coding :** Design details are implemented using appropriate programming language.

(ii) **Testing :** Testing is carried out to check whether flow of coding is correct, to check out the errors of program e.g. in C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables, to check whether program is giving expected output as per input specifications.

5. **Deployment**

- It includes software delivery, support and feedback from customer.
- If customer suggest some corrections, or demands additional capabilities then changes are required for such corrections or enhancement.

6. **Merits / Advantages**

(1) This model is very simple and easy to understand and use.

(2) Waterfall model is systematic sequential approach for software development. Hence it is most widely used paradigm for software development.

(3) In this approach, each phase is processed and completed at one time and thus avoids phases overlapping.

(4) It is very easy to manage since all the requirements are very well understood in the beginning itself.

(5) It establishes the milestones whenever the products are produced or reviews available.

7. **Demerits / Disadvantages**

(1) Problems in this model remain uncovered until software testing.

(2) One of the disadvantages of this approach is 'blocking states'. In blocking state, the members of development team waits for other members to complete the dependent tasks. Due to this, huge amount of time spent in waiting rather than spending time on some productive work. In today's world, the software work is fast paced and thus waterfall model is not useful.

(3) According to this model customer must state all his requirements at the beginning stage of development, which is difficult for the customer.

(4) This model is step-by-step systematic sequential approach. Ultimately, customer gets the working version of software too late. Hence customer requires patience.

(5) This approach is actually not realistic and does not match with real projects.

(6) Also it does not incorporate the risk assessment.

(7) Finally it is useful for smaller projects only where requirements are well understood in the beginning only.

2.2.1(A) V-Model (Software Development)

- In software engineering development process, the V-model represents a development process that may considered as an extension of the waterfall model.
- In more general V-model, instead of moving down in a linear way, the process steps are bent upwards after t coding phase as shown in Fig. 2.2.3.

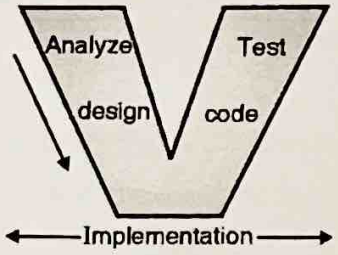


Fig. 2.2.3 : V-model

- The V-model demonstrates the relationships between each phase of the development life cycle.

2.2.2 Incremental Process Models

- Using these models a limited set of customer's requirements are implemented quickly and are delivered to customer.
- Then modified and expanded requirements are implemented step by step.
- Actually in this approach, the overall functionalities are split to smaller modules and these modules are quickly developed and delivered. Then refinements are possible in later increments or versions.
- In this approach, the initial requirements are very well defined. Each increment is passed through the overall development cycle i.e. phases of classic life cycle discussed earlier.
- The customer's feedback is very important after each of the increments (or releases) and next increment is developed. This process continues till the product is finished i.e. all the requirements are satisfied.
- Following are the two types of incremental process models :

(1) The incremental model (2) The RAD model

2.2.2(A) The Incremental Model

Q. Discuss incremental model for software development with merits and demerits. **MU - Dec. 16, 5 Marks**

- The incremental model combines the elements of waterfall model applied in an interactive fashion. The first increment is generally a core product.
- Each increment produces the product, which is submitted to the customer. The customer suggests some modifications.

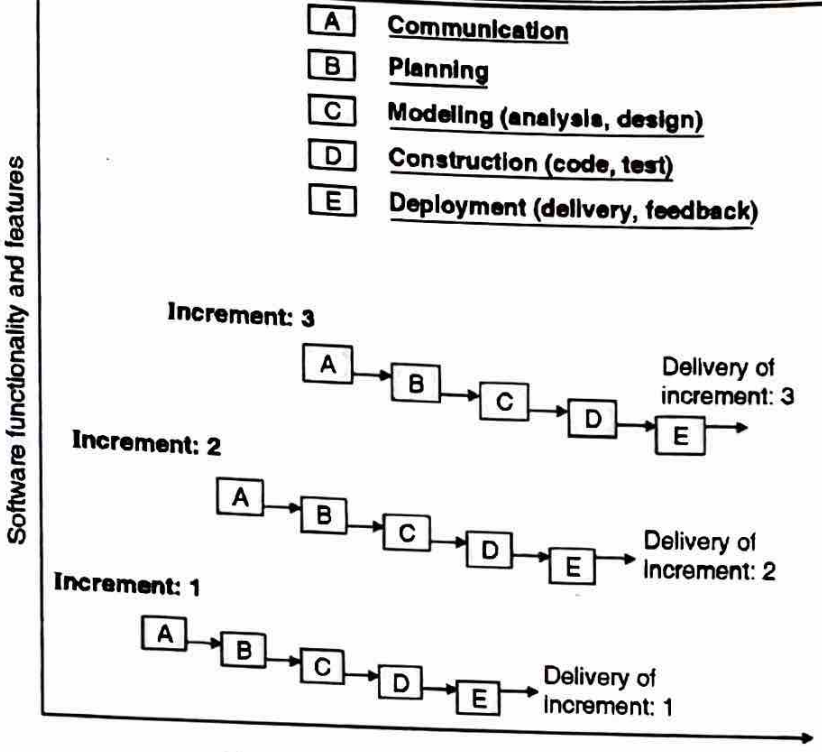


Fig. 2.2.4 : Project calendar time

- The next increment implements customer's suggestions and some additional requirements in previous increment. The process is repeated until the product is finished.
- Consider the development of word processing software (like MS Word or Word Star) using incremental model.

Increment – 1

Basic file management functions like : New, Open, Save, Save as etc. can be implemented in first increment.

Increment – 2

- More sophisticated editing and document production capabilities can be implemented in second increment.
- For example rulers, margins and multiple font selection can be implemented.

Increment – 3

Spelling, grammar checking and auto correction of words is implemented in third increment.

Increment – 4

The final advanced page layout capabilities are developed in fourth increment. And so on till the product is finished.

1. Merits / Advantages

(1) The incremental model is useful when the size of development team is small in the beginning or some team members are not available. Thus early increments can be implemented with small size of team.

(2) If the product satisfies the client, then the size of team can be increased.

(3) Also increments can be properly planned to handle all types of technical risks. For example some application may require a new hardware that is presently not available. In this situation the increments can be planned to avoid the use of that hardware.

(4) The initial product delivery is faster and cost of development is low.

(5) The customers can respond to the functionalities after each increment and come up with feedback.

2. **Demerits / Disadvantages**

(1) The cost of finished product may be increased in the end beyond the cost estimated.

(2) After each increment, the customer can demand for additional functionalities that may cause serious problems to the system architecture.

(3) It needs a very clear and complete planning and design before the whole system is broken into small increments.

2.2.2(B) The RAD Model

- RAD (Rapid Action Development) model is high-speed adaptation of waterfall model. Using RAD model, software product can be developed within a very short period of time i.e. almost within 60 to 90 days.
- The initial activity starts with communication between customer and developer. Depending upon initial requirements the project planning is done. Now the requirements are divided into different groups and each requirement group is assigned to different teams.
- Like other approaches, in RAD approach also all the generic framework activities are carried out. These generic framework activities are already discussed earlier.
- When all teams are ready with their final products, the product of each team is integrated i.e. combined to form a product as a whole.
- In RAD model each team carries out the following steps :

(1) **Communication :** It understands the business problem and information for software.

(2) **Planning :** Since various teams work together on different modules simultaneously, planning is important part of development process.

(3) **Modelling :** It includes :

(i) Business Modelling

It includes information flow among different functions in the project e.g., what information will be produced by each function, which functions handles that information etc.

(ii) Data Modelling

It includes different data objects used in software and relationship among different objects.

(iii) Process Modelling

During process modelling, process descriptions are created. e.g. add, modify, delete etc.

(4) **Construction :** It includes :

(i) Component reuse

Reusable components means the software components i.e. modules or procedures that are developed for specific application can be reused in development of other application.

(ii) Automatic code generation

The **software** producing the codes after providing proper inputs or selecting readymade options, e.g. Microsoft FrontPage used in Web development, Rational Rose used for Object Oriented analysis and designing.

(iii) Testing

Testing is carried out to check whether flow of coding is correct, to check out the errors of program e.g. In C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables, or check whether program is giving expected output as per input specifications.

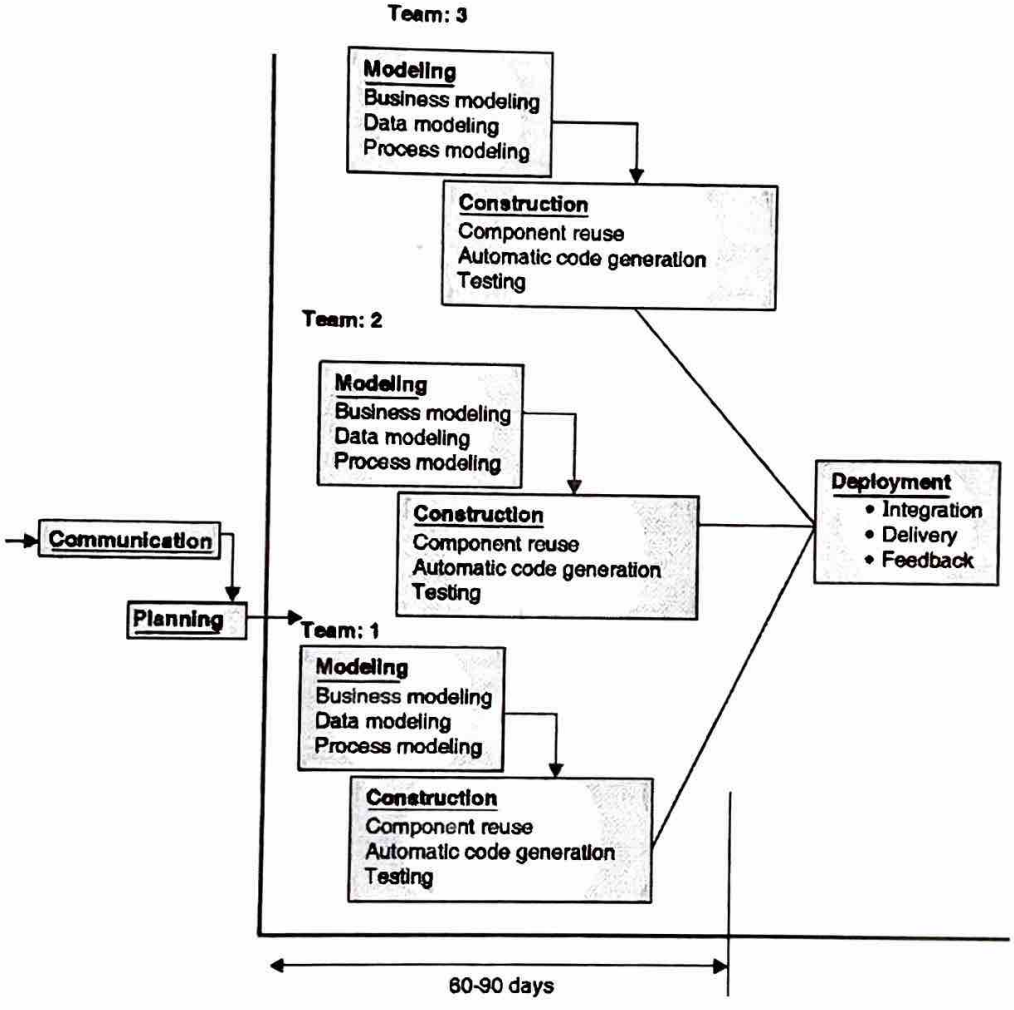


Fig. 2.2.5 : The RAD Model

1. Merits / Advantages

(1) Using the RAD approach, a product can be developed within very short period of time.

(2) It supports increased reusability of the components.

(3) It results in minimal code writing as it supports automatic code generation.

(4) It encourages the customer feedback.

(5) In this approach, quick initial reviews are possible.

(6) In this approach, modules integration is done from the beginning thus resolving number of integration issues.

2. Demerits / Disadvantages

(1) It can be used, if sufficient number of staff is available. Thus, it requires sufficient man power to create number of teams.

(2) In RAD approach, many teams work parallel to implement different functions of same project. Hence all teams must work with equal speed. If one team lags behind the schedule, overall project delivery will be late.

(3) If system can not be modularized properly, then building the components for RAD model will be problematic.

(4) The RAD model is not appropriate when technical risks are high. (e.g. a new application makes heavy use of new technology).

(5) This approach is useful for only larger projects.

(7) The RAD model had two primary disadvantages as follows :

- **Reduced scalability :** It occurs because a RAD application evolves from a prototype to a finished application. Thus there is less scope for scalability.
- **Reduced features :** It occurs due to time boxing. Time boxing is a time management technique in which the task is to be **completed** in a given frame of time called time boxes. Thus the features are pushed to be completed to the later releases due to short amount of time.

2.3 Evolutionary Process Models

- Evolutionary process models are iterative type models. Using these models the developer can develop increasingly more complete versions of the software.
- As the requirements change very often, the end product may be unrealistic and a complete version of the product is not possible. To overcome this drawback, the concept of limited version product is introduced and this version is gradually completed over the period of time.
- Each version is refined based on the feedbacks till it is completed.
- Following are the examples of evolutionary process models :

(1) The Prototyping paradigm
(2) The Spiral model
(3) The Concurrent development model

2.3.1 The Prototyping Paradigm

Q. Discuss prototyping model for software development with merits and demerits. **MU - Dec. 16, 5 Marks**

Q. What are the potential problems of Prototyping Model ? **MU - Dec. 17, 5 Marks**

- The **prototyping paradigm** offers best approach for human machine interaction. Ideally prototype serves as a mechanism for identifying software requirements.

- The prototyping approach is often called as **throw-away prototyping**. By using this approach, there is a rough demonstration of the customer's requirements i.e. the prototype used in demonstration is just a throwaway prototype.
- If working prototype is built, developer can use software tools if required (e.g. report generators).

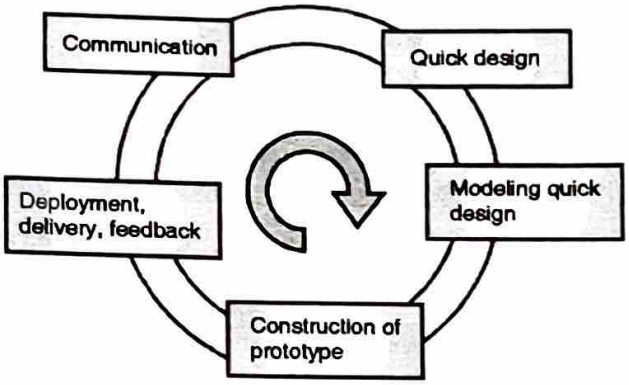


Fig. 2.3.1 : The prototyping model

- This produces working program quickly. Thus prototype can be served as 'first system'.

Different phases of prototyping model are :

1. **Communication**
   - The software prototyping paradigm starts with the communication between the developer and the customers.
   - The software engineer and the customer meet regularly to define the overall objectives of the desired software and to identify it's requirements.
2. **Quick design**
   - Quick design focuses on those aspects of software that are visible to the customer.
   - It includes clear input, output formats and human machine interfaces.
3. **Modelling quick design**

   The model of software is now built that gives clear idea of the software to be developed. This enables the developer to better understand the exact requirements.
4. **Construction of prototype**

   The concept of modelling the quick design leads to the development of prototype. This construction of prototype is deployed by the customer and it is evaluated by the customer itself.
5. **Deployment, Delivery, Feedback**
   - As picture of software to be built is very clear, customer can give his suggestions as a feedback.
   - If result is satisfactory, the model is implemented otherwise process is repeated to satisfy customers all requirements.

Merit / Advantage

Prototyping makes requirements more clear and system more transparent.

Demerits / Disadvantages

- The software developer generally compromises in the quality to get the working prototype quickly.
- Due to this reason, sometimes inappropriate operating system or programming language may be selected. He may use and implement inefficient algorithm.

2.3.2 The Spiral Model

Q.	With a neat diagram explain the Spiral model of software development.	MU - Dec. 17, 10 Marks
Q.	Differentiate in between waterfall and spiral model.	MU - May 19, 5 Marks

- The **Spiral model** is combination of well known waterfall model and iterative prototyping. It yields rapid development of more complete version of software.
- Using spiral model software is developed as series of evolutionary releases. During the initial releases, it may be just paperwork or prototype. But during later releases the version goes towards more completed stage.

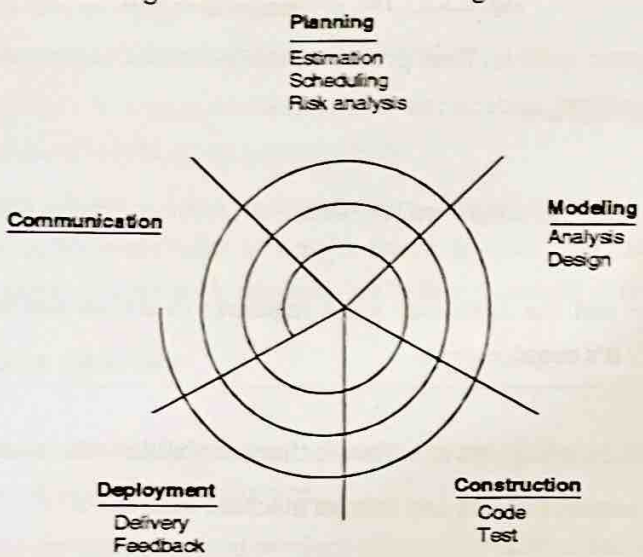


Fig. 2.3.2 : The Spiral Model

- The spiral model can be adopted to apply throughout the entire lifecycle of the application from concept development to maintenance. The spiral model is divided into set of framework activities defined by software engineer team. Each framework activity represents one segment of spiral as shown in Fig. 2.3.2.
- The initial activity is shown from centre of circle and developed in clockwise direction. Each spiral of the model includes following steps :

1. Communication

The software development process starts with communication between customer and developer.

2. Planning

It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and risk analysis.

3. Modelling

- It includes detail requirement analysis and project design (algorithm, flowchart etc).
- Flowchart shows complete pictorial flow of program whereas algorithm is step by step solution of problem.

4. Construction

It includes coding and testing steps :

(i) Coding : Design details are implemented using appropriate programming language.

(ii) Testing : Testing is carried out.

5. Deployment

- It includes software delivery, support and feedback from customer. If customer suggest some corrections, or demands additional capabilities then changes are required for such corrections or enhancement.
- Note that after customer evaluation, next spiral implements, 'customer's suggestions' plus 'enhancement plan'. Thus, each of iteration around the spiral leads to more completed version of software.

1. Merits / Advantages

(1) In this approach, project monitoring is very easy and more effective compared to other models.

(2) It reduces the number of risk in software development before they become serious problems.

(3) It is suitable for very high risk projects.

(4) Project estimates i.e. schedule and cost is more realistic.

(5) Risk management is in-built feature of spiral model.

(6) Changes can be accommodated in the later stages of development.

2. Demerits / Disadvantages

(1) If major risk is not discovered in early iteration of spiral, it may become a major risk in later stages.

(2) Each iteration around the spiral leads to more completed version of software. But it's difficult to convince (especially in contract situation) to the customer that the model is controllable.

(3) Cost of this approach is usually high.

(4) It is not suitable for low risk projects.

(5) Rules and protocols must be followed very strictly to implement the approach.

2.3.3 The Concurrent Development Model

- The **concurrent development** model is also called as **concurrent model**. This model is more appropriate for system engineering projects where different engineering teams are involved. In system engineering stage, the project requirements are considered at system level.
- Diagrammatically it can be represented as a series of framework activities, task, actions and their associated states.

- Fig. 2.3.3 exhibits one element of the concurrent process model :

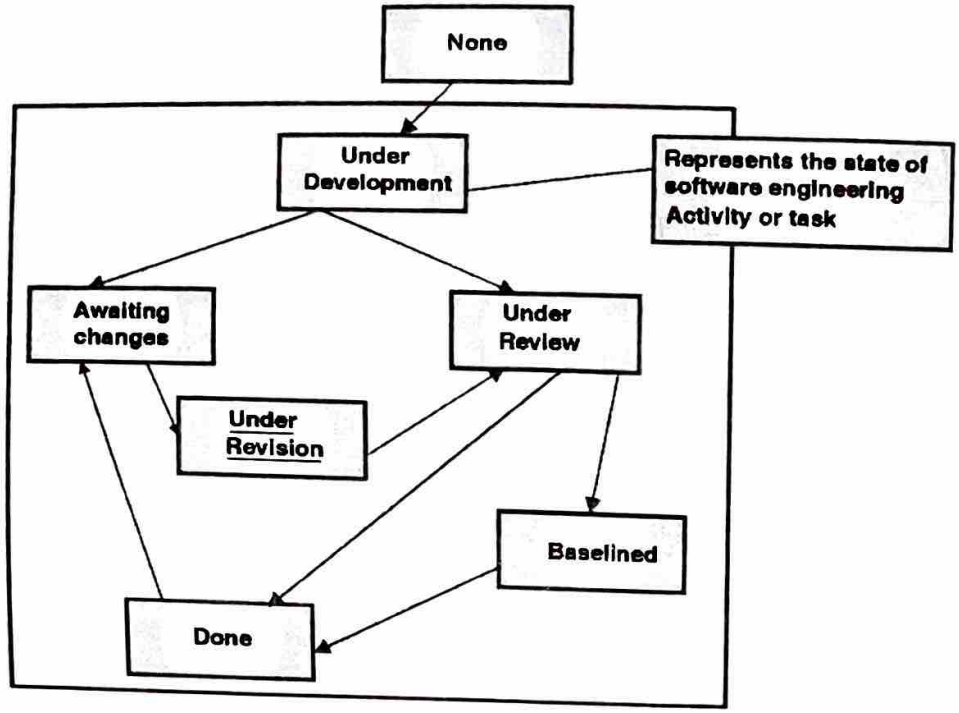


Fig. 2.3.3 : One element of concurrent process model (i.e. Modeling activity)

- The **modeling activity** is one of the states and other activities like **communication** or **construction** can also represented in an analogous manner.
- Let the **communication activity** has completed its first iteration and available in **awaiting changes state.** T **modelling activity** has completed its initial communication and ready to move to **under development** sta from none state.
- During these transitions, if customer indicates some modification in the requirement, then the **modelli activity** transits to **awaiting changes state** from **under development state.**
- Instead of considering activities, actions and tasks as sequence of events, it defines network of activities.
- The concurrent process model activities transits from one state to another state and this model is used in types of software development and it provides very clear idea of the current state of the project.

1. Merits / Advantages

(1) The concurrent development model is applicable to all types of software development processes.

(2) It gives very clear picture of current state of the project.

(3) It is easy to use and easy to understand.

(4) This approach is flexible and number of releases determined by the development team.

(5) New functionalities as elicited by the client can be accommodated late in the project.

(6) It has immediate feedback from testing.

2. **Demerits / Disadvantages**

(1) Since all the stages in the concurrent development model works concurrently, any change in the requirement from the client may halt the progress.

This may happen due to dependent components among different stages and it lead to more longer development cycles as compared the planned cycles.

(2) It requires excellent and updated communication between the team members. This may not be achieved all the time.

(3) The SRS must be updated at regular intervals to reflect the changes.

2.3.4 Differentiation between Prescriptive and Evolutionary Process Models

Sr. No.	Prescriptive process models	Evolutionary process models
1.	Developed to bring order and structure to the software development process.	Evolutionary software processes do not establish the maximum speed of the evolution. Due to this development process becomes slow.
2.	Defines a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software	Evolutionary process models lack flexibility, extensibility, and high quality.
3.	It is more popular.	It is less popular.
4.	It provides complete and full developed systems.	Time does not allow a full and complete system to be developed.
5.	**Example** : Water fall model, Incremental models.	**Example** : Prototyping, Spiral and Concurrent models.

Review Questions

Q. 1 State the generic process framework activities.

Q. 2 Explain waterfall model with its advantages and disadvantages.

Q. 3 Explain the waterfall model with V model.

Q. 4 Explain the waterfall model with work products of each activity.

Q. 5 Explain the incremental model with advantages and disadvantages ?

Q. 6 Explain the RAD model with advantages and disadvantages?

Q. 7 Explain evolutionary process models mentioning the types of projects for which they are suitable.

3 Agile Process Models

Syllabus

Agile process : Extreme Programming (XP), Scrum, Kanban Model

3.1 Agile Process Model

Q.	What is Agile Methodology ?	**MU - May 15, Dec. 19, 4 Marks**
Q.	What is agility in context of software Engineering ?	**MU - Dec. 15, May 17, 5 Marks**
Q.	What are Agile Processes ?	**MU - May 16, 5 Marks**
Q.	Write short note on Agile Methodology.	**MU- Dec. 16, 10 Marks**
Q.	Write short note on Agile Process Models.	**MU - May 18, 10 Marks**

- An agile process model includes the concept of development along with a set of guidelines necessary for the development process. The conceptual design is necessary for the satisfaction of the customer. The concept is also necessary for the incremental delivery of the product. The concept or the philosophy makes development task simple. The agility team must be highly motivated, updated with latest skill sets and should focus on development simplicity.
- We can say that agile development is an alternative approach to the conventional development in certain projects.
- The development guidelines emphasize on analysis and design activities and continuous communication between developers and customers. An agile team quickly responds to changes. The changes may be in development, changes in the team members, and changes due to new technology. The agility can be applied to any software process. It emphasizes rapid delivery of operational software.
- All the agile software processes should address three important assumptions :
  - Difficult to predict in advance software requirements
  - Design and construction are interleaved in most of the projects, it is difficult to predict design before construction.
  - Analysis, design, construction and testing are not much predictable.
- To address these assumptions of unpredictability, the agile development process must be adaptable. So an agile process must adapt incrementally.

3.1.1 Comparison between the Agile and Evolutionary Process Models

Sr. No.	Agile process model	Evolutionary process model
1.	These models satisfy customer through early and continuous delivery.	Using these models the developer can develop increasingly m complete versions of the software.
2.	It can accommodate changing requirements.	It yields rapid development of more complete version of softw rather accommodating continuous changing requirements.
3.	In this development process customer, business people and developers must work together daily.	Daily meetings are not required. After one evolution of desi customer, business people and developers can meet and discuss.
4.	The messages are conveyed orally i.e. face-to-face. Conversation is essential.	Oral messages are not so essential.
5.	Technical excellence and good design is achieved.	The developer often compromises in order to get working prototy quickly. Thus inappropriate operating system or programmi language may be used simply because it's available and known. Th inefficient algorithm may be implemented.

3.2 Agile Software Development

- In today's modern world, businesses spread across the globe in each sphere of life. It has become the glo phenomenon. Due to this changing environment, the client must respond to new opportunities and to the vola market conditions, and to the arrival of new products and services.
- The software application is actually the heart of all the business operations. So according to the rapid chang rules and business ideas, it should respond quickly.
- Thus the developer is supposed to develop the new software quickly to meet the desired requirements in stipulated time.
- In fact, in such rapidly changing market condition, rapid development of the software and its urgent deliver the need of the time and it is supposed to be the most critical requirement of any software system.
- Normally most of the clients are even ready to compromise the quality of software and the requirements at cost of faster development of the software and its quick delivery.
- Since all the businesses usually operate in the rapidly changing environment, it becomes highly impossible gather complete set of requirements. The requirements elicited by the customers in the beginning are alw changed since the customers find it difficult to predict the actual working of the software and the challenges come across.
- Thus understanding the requirements quickly and accommodate the requirements change narrated by customer quickly is the need of the time. Otherwise, delayed delivery of the software may make it unusable out of date.

- Thus waiting for the complete requirements gathering from the customer will not work for the rapid development. Since the requirement of the customer changes with the progress of the software development process.
- Due to this reason, the conventional approaches like waterfall model or specification based processes will delay the final delivery of the software system.
- But in some cases like critical control system, the complete analysis of the requirement is mandatory else it may cost life. For such type of safety systems, plan-driven approach is always the best choice.
- Rapid software development processes are the ultimate choice for the development of useful products in quick time span. In this approach the software is developed as a series of increments.
- Agile methods are incremental development methods in which the increments are usually small the new versions of the system are created rapidly and handed over to the customers within the span of 15 to 20 days and their feedback is received for the next versions.

3.2.1 Agile methods

- In the early days of software development, developer used to plan the project carefully, focus on quality assurance and employee the analysis and design methods supported by various CASE tools.
- This was the general practice of the software developer community that was responsible for the development of larger projects such as government projects etc.
- This larger projects use larger team and the team members may spread geographically across the world and they worked on the software for the longer period of time.
- Therefore, the development period may extend up to the 10 years from the initial specification to the final delivery of the product.
- Such a planed-driven approach will involve significant overheads in all the stages of the development processes like requirement specification, planning, designing, coding and documentation.
- In order to overcome this heavy weight plan-driven approach, the concept of agile methods was proposed. This methodology mainly focused on the product instead of focusing on the design and documentation part.
- All the agile methods trust on the incremental approach instead of the conventional waterfall approach. The incremental approach is the best approach where the customer requirements and the software requirements change rapidly or frequently.
- The philosophy behind agile methods is also observed in **Agile Manifesto** that is accepted by most of the leading software developers. The Agile Manifesto is discussed in the following section.

3.2.2 Agile Manifesto

- The Agile Manifesto states that :

  "We are uncovering better ways of developing software by doing it and helping others do it.

  Through this work we have come to value,

- **Individuals and interactions** work over processes and tools
- **Working software** takes responsibility of comprehensive documentation
- **Customer collaboration** look after contract negotiation
- **Responding to change** after having a good plan."

- Even though these agile methods are based on the concept of incremental development, quicker delivery a faster deployment, the agile manifesto gives different processes to achieve this.
- But most of the set of principles used by various experts that are based on agile manifesto are very muc common.

3.2.3 Agility Principles

Q. Explain principles of Agile Methodology. **MU - May 15, 4 Marks**

Q. Describe and discuss characteristics of Agile Process Model. **MU - Dec. 17, 10 Marks**

- Agile software development describes a set of principles for software development under which requirement and solutions evolve through the collaborative effort of self-organizing cross-functional teams.
- It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and encourages rapid and flexible response to change.
- The agile alliance has given twelve agility principles as follows :
  (1) Satisfy customer through early and continuous delivery.
  (2) Accommodate changing requirements.
  (3) Deliver the working software frequently in shorter time span.
  (4) The customer, business people and developers must work together on daily basis during entir development.
  (5) Complete the task with motivated developers and facilitate healthy and friendly environment.
  (6) Convey the message orally, face-to-face conversation.
  (7) Working software is the primary measure of progress.
  (8) Agile process promotes sustainable development.
  (9) Achieve technical excellence and good design.
  (10) There must be simplicity in development.
  (11) The best architecture, requirements and design emerge from self-organizing teams.
  (12) Every team should think how to become more effective. It should review regularly and adjust its behaviou accordingly.
- Thus the principles of agility may be applied to any software process. To achieve agile development, the tea must obey all the principles mentioned above and conduct proper planning.
- All the agile software processes should address three important assumptions :
  - Difficult to predict in advance software requirements

- Design and construction are interleaved in most of the projects, it is difficult to predict design before construction.
- Analysis, design, construction and testing are not much predictable.

- To address these assumptions of unpredictability, the agile development process must be adaptable. So an agile process must adapt incrementally.

3.3 Extreme Programming Practices

- The Extreme Programming is one of the most commonly used agile process models. All the agile process models obey the principles of agility and the manifesto of agile software development.
- The XP uses the concept of object oriented programming. This approach is preferred development paradigm.
- As in conventional approach, a developer focuses on the framework activities like planning, design, coding and testing, the XP also has set of rules and practices.
- Following Fig. 3.3.1 shows the Extreme Programming process and all the key XP activities are summarized :

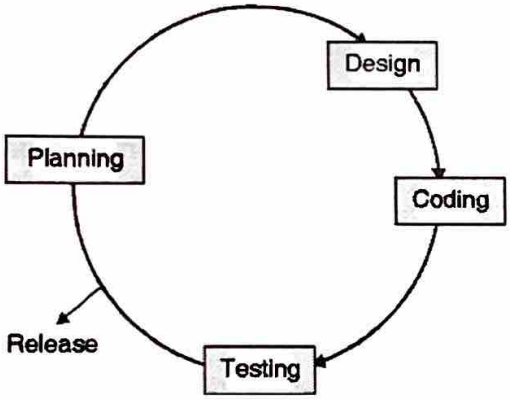


Fig. 3.3.1 : Extreme programming process

3.3.1 XP Values

- Following are a set of five values that establish a foundation for all work performed in context with XP.
  (1) Communication
  (2) Simplicity
  (3) Feedback
  (4) Courage
  (5) Respect
- For any development process, there must be a regular meeting of developer and the customer. There should be a proper communication for requirement gathering and discussion of the concepts.
- The simple design can always be easily implemented in code.
- The feedback is an important activity which leads to the discipline in the development process.
- In every development projects, there is always a pressure situation. The courage or the discipline will definitely make the task easy.

- In addition to all the XP values, the agile should inculcate respect among all the team members, between oth stake holder and with customers.

3.3.2 The XP Process

Q. Explain Extreme Programming (XP) with suitable diagram. **MU - Dec. 15, May 17, 5 Marks**

- The XP process encompasses four framework activities :

(i) Planning	(ii) Design
(iii) Coding	(iv) Testing

- **Planning** starts with requirement gathering activity that enables XP team to understand the business rules fo the software. Customers and developers work together to decide the final requirement.
- **Design** of the product follows the KIS (Keep It Simple) in the XP environment. A simple design is alwa welcome over the complicated ones.
- **Coding** starts after the design work is over. Once the code is completed, it can be unit-tested immediately. Th important concept during the coding activity in XP recommends that two people work together at one compute to create the code.
- **Testing** - All the individual unit tests are organized into a 'universal testing suite'. Integration and validatio testing of the system can occur on daily basis. XP acceptances test, also called customer tests are specified b customer and focus on overall system features and functionality.

3.4 Scrum

Q. Write short note on SCRUM. **MU - Dec. 17, 10 Marks**

- It is one of the agile software development methods. It is an iterative and incremental software developme framework. It gives holistic and flexible development strategies.
- It enables teams to self organize by online collaborations of team members. Its key principle is to recognize tha during project development customers can change their requirement and requirements of customers ar unpredictable.
- It adopts empirical approach. It assumes problem cannot be fully understood or defined but focusing o maximizing team ability to deliver quickly.
- Different terminologies used in SCRUM process is as follows :
  - **SCRUM team :** It contains product owner, development team and scrum master.
  - **Product owner :** Person responsible for product backlog.
  - **SCRUM master :** Person responsible for scrum process.
  - **Development team :** Team of people responsible for delivering the product.
  - Sprint burn down chart : Daily sprint progress.
  - **Release burn down chart :** Chart of completed product backlog.

- **Product backlog :** List of priority wise requirement.
- **Sprint backlog :** List of task to be completed which are prioritized.
- **Sprint :** Period in which development occurs. Duration of time
- **Spike :** Period used to research concept.
- **Tracer bullet :** It is spike with current architecture, technology, best practices, etc.
- **Tasks :** Items added to sprint backlog at the beginning of sprint which are broken into hours.
- **Definition of done :** Exit criteria which decides product backlog items are completed or not.
- **Velocity :** Total efforts a team is capable of in a sprint.
- **Scrum-but :** It is exception to scrum methodology.

3.4.1 Process Flow

SCRUM process flow contains different stages. Different stages are as follows :

1. **Setup**
   - Setup environment contains the following :
   - Remove any customization from existing application.
   - Activate scrum process pack plug-in.
   - Assign scrum roles to users.
2. **Create a product**
   - Product represents the functionality which is identified by owner.
   - Product contains different features which are needed for enhancement which are useful for customer satisfaction.
   - It can have few features or many features.
3. **Create user stories**
   - These are product requirement created by product owner.
   - User stories are written in plain language. Less technical jargons are used.
   - It contains specific requirement that product should have.
   - Stories cannot be created without associating with product.
4. **Create release**

   It has start and end date in which number of development iterations are completed.
5. **Create sprint**
   - It is basic unit of time in development process.
   - It can have any length. Typically it takes one to four weeks to complete.
   - Scrum master creates one or more sprints.

- Sprints should release in given start and end dates.
- Scrum team need to complete all the stories which are committed.
- Scrum master expects all the stories are fully implemented and ready to release.

6. Plan sprint

- Team and scrum master need to decide on which stories they can commit to complete within a sprint.
- Scrum master make sure that capability of sprint team matches the requirement of stories.
- If capacity of the stories exceeds then scrum master add team members, remove stories or add sprint as when needed.
- Velocity chart is used in estimation process.
- Velocity chart shows the historical record of number of completed points.
- With the help of velocity chart scrum master get the idea of capacity of team.
- Velocity chart is important tool in planning sprint.

7. Track sprint progress

- Scrum master is responsible for checking the progress.
- He provides the progress report of the team.
- Team members frequently update task and records.

8. Generate charts

- Progress of a sprint can be tracked with the help of different charts.
- Chart is one of the important tools of scrum team.
- Burn down chart compares ideal progress in sprint against the actual progress.
- It is done on daily basis.

3.4.2 Scrum Roles

Different roles are used in scrum process. Three different roles are as follows :

(1) Product owner
(2) Development team
(3) Scrum master

1. Product owner

- They are stakeholders of the customers.
- They ensure teams delivers values to business.
- They write customer centric items, rank customer centric item and add to product backlog.
- The scrum process needs to be one product owner. Sometimes they are part of development team.

Role of the product owners in product requirement are as follows :

- Important role of product owner is communication with development team.
- Identifying important requirement and communicating is important task of product owner.
- They generally reduce the communication gap between team and stakeholders.
- They demonstrate the solution to stakeholders.
- They announce the different release of the product.
- They communicate the team status.
- Organizes the milestone review.
- They educate the stakeholders in development process.
- They negotiate priorities, scope, funding and schedule.

2. **Development team**

- They are responsible for delivering the product.
- Team generally consists of people of different skills.
- Team size can be from between 3 to 9.
- They includes member like analyst, designer, developer, tester, technical communicator, document writing, etc.
- Development team is self organizing.

3. **Scrum master**

- Scrum is facilitated by scrum master.
- He is accountable for product goals and deliverables.
- He is not a team lead or manager. He acts as a buffer between development team and distracting influences.
- He is the enforcer of the scrum process.
- He generally involves in key meeting and challenges the team to improve.
- Project manager is responsible for people management but scrum master is not related to people management.

3.4.3 Scrum Cycle Description

- Scrum cycle is divided into different activities.
- Scrum cycle activities are as follows :

(1) Define
(2) Plan
(3) Build
(4) Review
(5) Retrospect

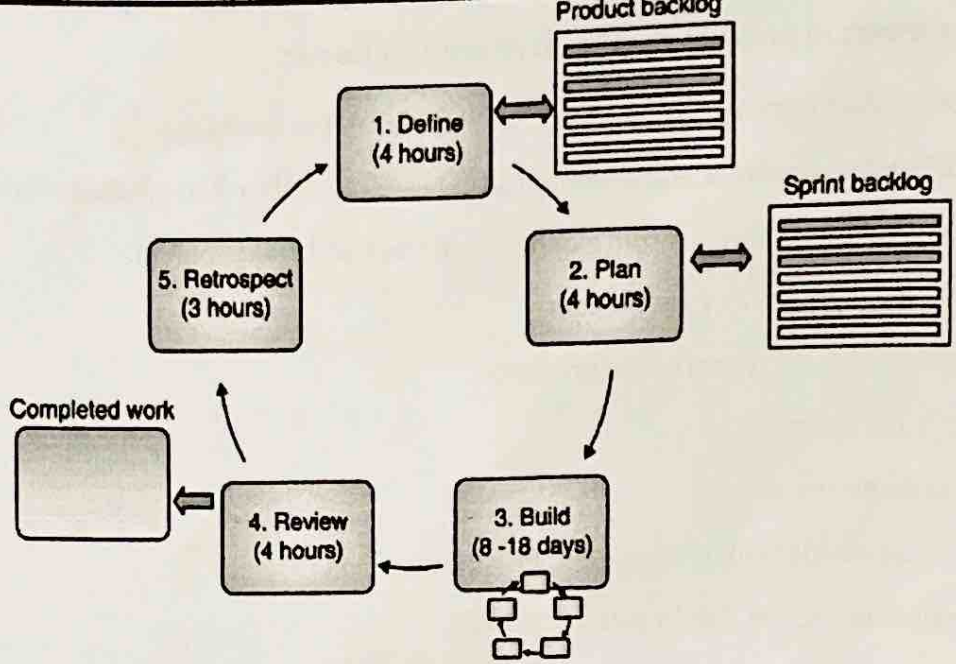


Fig. 3.4.1 : SCRUM cycle

(1) Define : During defining the requirement generally scrum team, scrum master and product owner s together. They decide the priority of team for duration of sprint. It is useful in high level direction for t team.

(2) Plan : It is about selecting items from product backlog. It also evaluates value of different items and estima the efforts needed by scrum team. Generally items are selected from product backlog which can b considered as sprint backlog.

(3) Build : Task from the sprint backlog is selected for development. They go on selecting the items till t items from the sprint backlog get completed.

(4) Review : Review is presented by scrum team to product owner about the implementation of the items.

(5) Retrospect : It is final steps of the retrospection. It has few objectives. It focuses on identifying t improvement area by the scrum team. They also discuss on the success of earlier iterations.

3.4.4 Product Backlog

- It is ordered list of requirements which are maintained for product.
- It contains different details like features, bug fixes and non functional requirements.
- It contains whatever is needed for the success of the product.
- Items from the product backlog are ordered by product owner using different factors which includes da needed to complete the task, business values, risk, dependencies, etc.
- Product backlogs are written in story format.
- It mainly contains what exactly need to be delivered.
- It also contains the order in which sequence the requirement should be delivered.
- Requirement listing and ordering that requirement is solely done by product owner.
- It contains assessment of business values by the product owner.

- Development team's assessment is also part of product backlog.
- Different methods are used by product owner to decide the priority of requirement which includes Fibonacci sequence and other methods.
- If some requirements are urgent then product owner can request to scrum team about prior requirement.
- Sizes of backlog items are determined by the development team.
- It does not contain only user stories, but it contains other information also.
- Product owner is responsible for maximizing the value of the product. He is also responsible for maximizing the work of development team.
- Product backlog is used for :
    - Taking request for product modification.
    - It includes adding new facility, replace old facility, remove some unwanted features.
    - Ensure delivery team is given work which maximizes the business benefits to the owner of the product.

3.4.5 Sprint Planning Meeting

- Product owner, development team and scrum master is involved in the sprint planning meeting.
- If required outside beneficiary can be invited for sprint planning meeting.
- In this meeting product owner describes the highest priority work and communicate to development team.
- Development team ask the question if any doubts, to the product owner.
- This meet is held at the beginning of sprint cycle.
- They generally select what work need to be done.
- They prepare sprint backlog which gives details of time taken to complete the work.
- They discuss how much work should be done during the current sprint cycle.
- Maximum time limit is 8 hours.
- Within 8 hours, in the first four hour entire team discusses about prioritizing the product backlog.
- In second 4 hours, development team plan for sprint which comes with sprint backlog.
- Sample agenda for sprint planning meeting is as follows :
    - Product roadmap
    - Name of the iterations
    - Capacity of team
    - Review
    - Stories
    - Commit
    - Status of development
    - Velocity of previous iterations
    - Concerns
    - Updates
    - Tasking out
- Sprint planning meeting template is shown in Fig. 3.4.2.

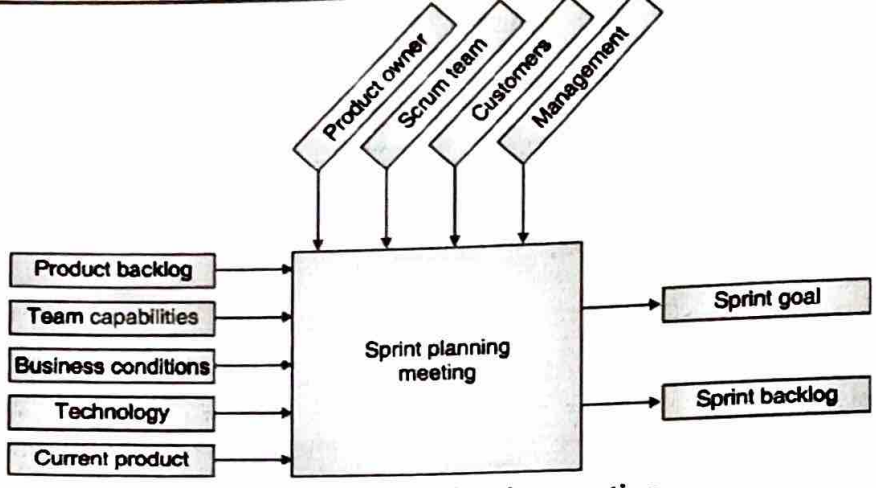


Fig. 3.4.2 : Sprint planning meeting

3.4.6 Sprint Backlog

- It is nothing but list of work development team need to complete in the sprint.
- This list is derived from product backlog.
- It contains list of user stories in sprint in the order of priority.
- It also contains the relative effort estimate.
- The task needed to develop every story is also maintained in the sprint backlog.
- In sprint planning meeting team selects some number of product backlog items.
- Product backlog items are selected in user story format.
- They also identifies user stories need to complete.
- Size of sprint backlog is selected by team.
- Sprint backlog can be maintained using spreadsheet.
- Example of sprint backlog is shown in Table 3.4.1.

Table 3.4.1

User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	....
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information	Update security tests	6	6	4	0		
	Design a solution to	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests...	12	12	10	6		
	Code the ...	8	8	8	4		

3.4.7 Sprint Execution

- Sprint is basic unit of development in scrum development.
- Scrum make progress is series of sprint.
- During the sprint every day project status meeting occurs. This meeting is called as daily scrum meeting.
- Sprint execution is nothing but the work which is performed by scrum team to meet sprint goals.
- All the work which is necessary to deliver is performed.
- It accounts for majority of time during a sprint.
- It begins after sprint planning.
- It ends when sprint review starts.

3.4.8 Daily Scrum Meeting

- Every day during sprint project meeting occurs.
- This meeting is called as daily scrum meeting.
- It is helpful for schedule coming day's work.
- Scrum meeting has following guidelines :
  - All members presents for meeting with updates.
  - Usually meeting starts on time even if some members are absent.
  - Usual meeting time is at the morning.
  - Meeting time and venue are same every day.
  - Approximate meeting time is 15 minutes.
  - Generally core team member speak in the meeting.

Different questions are answered by team. Some of the questions are as follows :

- What have been done since yesterday?
- What is today's planning?
- Any impediments or stumbling blocks?

Sample scrum meeting is shown in Table 3.4.2.

Table 3.4.2

Monday	Tuesday	Wednesday	Thursday	Friday
		Sprint 1 planning 1 pm. - 5 pm.	Daily scrum 9:15 - 9:30	Day scrum 9:15 - 0:30
aily scrum 15 - 9:30	Daily scrum 9:15 - 9:30 Backlog grooming 1 pm - 2 pm	Daily scrum 9:15 - 9:30	Daily scrum 9:15 - 9:30 Backlog grooming 1 pm - 2 pm	Daily scrum 9:15 - 9:30

Monday	Tuesday	Wednesday	Thursday	Fri
Daily scrum 9:15 - 9:30	Daily scrum 9:15 - 9:30 Dry run and prep for sprint review (optional) 2 pm - 3 pm	Daily scrum 9:15 - 9:30 Sprint 1 review 10 am - 11 am Sprint retrospective 1 11 am - 12 pm Celebration lunch 12 pm - 1 pm Sprint 2 planning 1 pm - 5 pm		

3.4.9 Maintaining Sprint Backlog and Burn-Down Chart

- Burn down is graphical representation of work left versus time.
- Pending work is on vertical axis and time is at horizontal axis.
- It is nothing but the run chart of outstanding work.
- It is useful in predicting when the work will be completed.
- It is used in agile software development particularly in scrum process.
- It can be applicable to any project.
- Sample burn down chart is shown in Fig. 3.4.3.

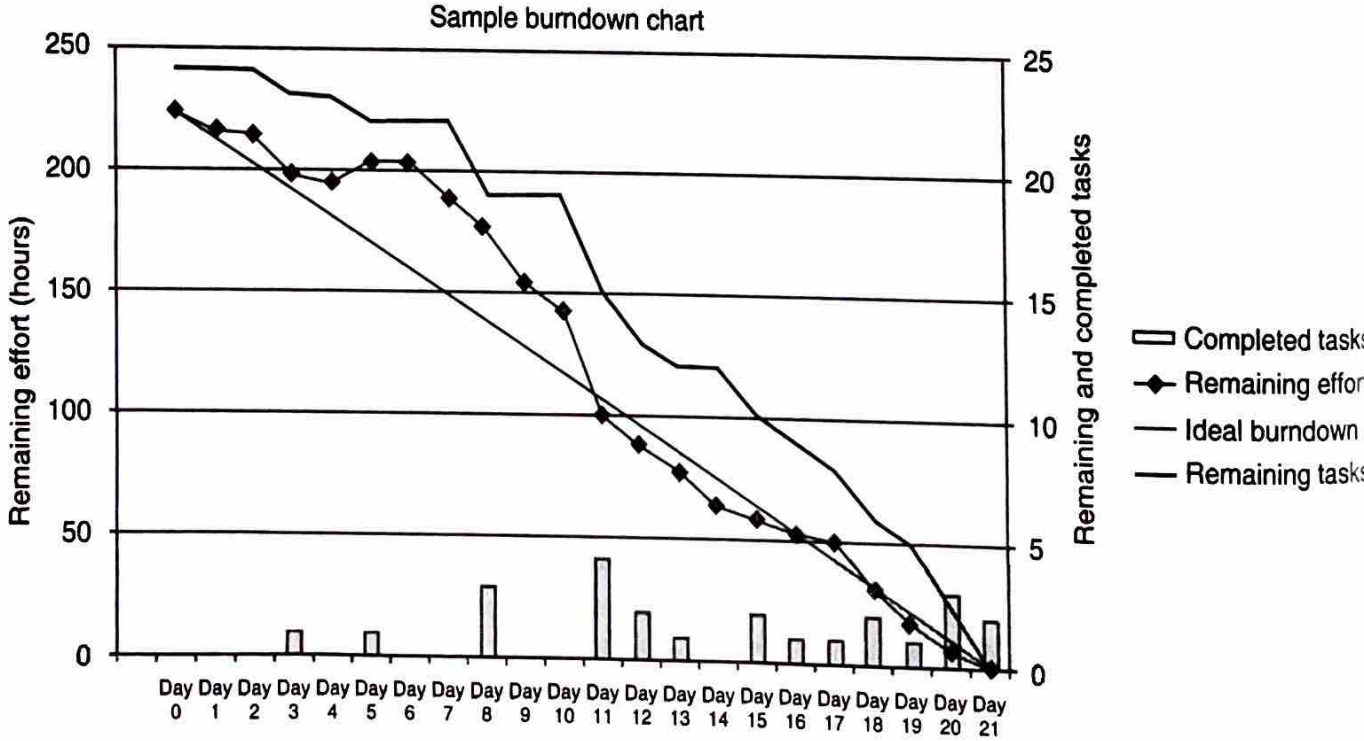


Fig. 3.4.3 : burn down chart

3.4.10 Sprint Review and Retrospective

- These meetings are held at the end of sprint cycle.
- These meetings are sprint review and retrospective meeting.
- In the sprint review meeting following points are discussed :
  - Review the completed work.
  - Review incomplete work.
  - Show the completed work to stakeholders.
  - Incomplete work not demonstrated.
  - Meeting is limited to four hours.
- In the sprint retrospective following points are considered :
  - Every team member reflects on past sprint.
  - Make continuous process improvement.
  - Two questions are asked in the sprint. One question is : What is well in the current sprint? What improvements can be done in next sprint?
  - The time limit for the sprint retrospective meeting is 3 hours.
- This meeting is head by scrum master.

3.5 Introduction to Agile Tools : Kanban

- Kanban is a popular framework used by software teams practicing agile software development. It is enormously prominent among today's agile software teams, but the kanban methodology of work dates back more than 50 years.
- In the late 1940s Toyota began optimizing its engineering processes based on the same model that supermarkets were using to stock their shelves.

Supermarkets stock just enough products to meet consumer demand, a practice that optimizes the flow between the supermarket and the consumer.

Because inventory levels match consumption patterns, the supermarket gains significant efficiency in inventory management by decreasing the amount of excess stock it must hold at any given team.

Meanwhile, the supermarket can still ensure that the given product a consumer needs is always in stock.

Agile software development teams today are able to leverage these same JIT principles by matching the amount of work in progress (WIP) to the team's capacity.

This gives teams more flexible planning options, faster output, clearer focus, and transparency throughout the development cycle.

While the core principles of the framework are timeless and applicable to almost any industry, software development teams have found particular success with the agile practice.

- In part, this is because software teams can begin practicing with little to no overhead once they understa basic principles.
- Unlike implementing kanban on a factory floor, which would involve changes to physical processes an addition of substantial materials, the only physical things a software teams need are a board and cards, and those can be virtual.

3.5.1 Kanban Boards

- The work of all kanban teams revolves around a kanban board, a tool used to visualize work and optimiz flow of the work among the team.
- While physical boards are popular among some teams, virtual boards are a crucial feature in any agile soft development tool for their traceability, easier collaboration, and accessibility from multiple locations.
- Regardless of whether a team's board is physical or digital, their function is to ensure the team's wo visualized, their workflow is standardized, and all blockers and dependencies are immediately identified resolved.
- A basic kanban board has a three-step workflow: To Do, In Progress, and Done. However, depending on a te size, structure, and objectives, the workflow can be mapped to meet the unique process of any particular tea
- The kanban methodology relies upon full transparency of work and real-time communication of capa therefore the kanban board should be seen as the single source of truth for the team's work.

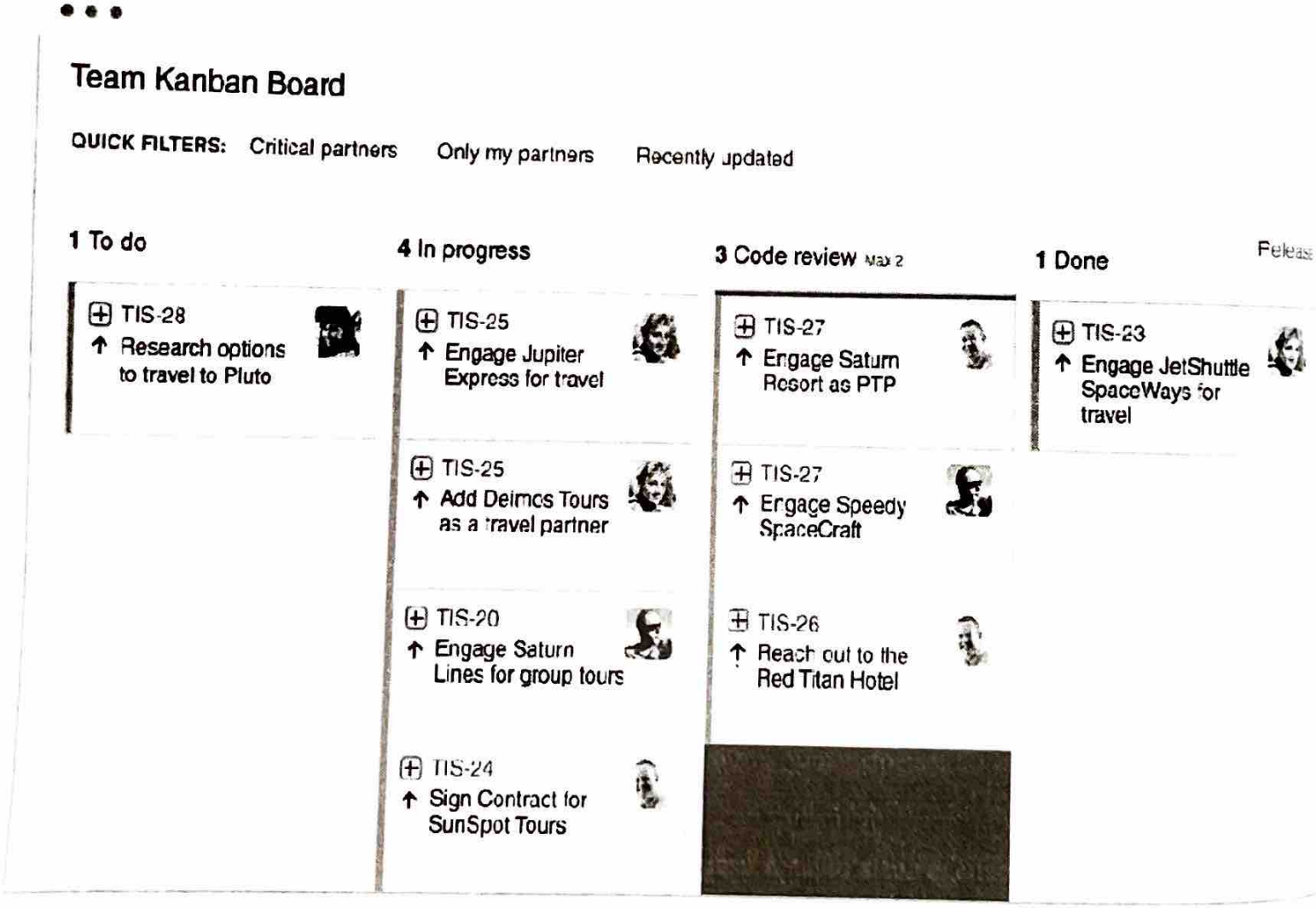


Fig.3.5.1 : Kanban boards

3.5.2 Kanban Cards

- In Japanese, kanban literally translates to "visual signal." For kanban teams, every work item is represented as a separate card on the board.
- The main purpose of representing work as a card on the kanban board is to allow team members to track the progress of work through its workflow in a highly visual manner.
- Kanban cards feature critical information about that particular work item, giving the entire team full visibility into who is responsible for that item of work, a brief description of the job being done, how long that piece of work is estimated to take, and so on.
- Cards on virtual kanban boards will often also feature screenshots and other technical details that is valuable to the assignee.
- Allowing team members to see the state of every work item at any given point in time, as well as all of the associated details, ensures increased focus, full traceability, and fast identification of blockers and dependencies.

3.5.3 The Benefits of Kanban

- Kanban is one of the most popular software development methodologies adopted by agile teams today. Kanban offers several additional advantages to task planning and throughput for teams of all sizes.
- Planning flexibility : A kanban team is only focused on the work that's actively in progress. Once the team completes a work item, they pluck the next work item off the top of the backlog.
- The product owner is free to reprioritize work in the backlog without disrupting the team, because any changes outside the current work items don't impact the team.
- As long as the product owner keeps the most important work items on top of the backlog, the development team is assured they are delivering maximum value back to the business. So there's no need for the fixed-length iterations you find in scrum.
- Shortened cycle times : Cycle time is a key metric for kanban teams. Cycle time is the amount of time it takes for a unit of work to travel through the team's workflow–from the moment work starts to the moment it ships.
- By optimizing cycle time, the team can confidently forecast the delivery of future work.
- Overlapping skill sets lead to smaller cycle times. When only one person holds a skill set, that person becomes a bottleneck in the workflow. So teams employ basic best practices like code review and mentoring help to spread knowledge.

Shared skills mean that team members can take on heterogeneous work, which further optimizes cycle time. It also means that if there is a backup of work, the entire team can swarm on it to get the process flowing smoothly again.

For instance, testing isn't only done by QA engineers. Developers pitch in, too.

In a kanban framework, it's the entire team's responsibility to ensure work is moving smoothly through the process.

- Fewer bottlenecks : Multitasking kills efficiency. The more work items in flight at any given time, the m context switching, which hinders their path to completion.
- That's why a key tenant of kanban is to limit the amount of work in progress (WIP). Work-in-progress lim highlight bottlenecks and backups in the team's process due to lack of focus, people, or skill sets.
- For example, a typical software team might have four workflow states: To Do, In Progress, Code Review, a Done. They could choose to set a WIP limit of 2 for the code review state.
- That might seem like a low limit, but there's good reason for it: developers often prefer to write new code, rath than spend time reviewing someone else's work.
- A low limit encourages the team to pay special attention to issues in the review state, and to review others wo before raising their own code reviews. This ultimately reduces the overall cycle time.
- Visual metrics : One of the core values is a strong focus on continually improving team efficiency an effectiveness with every iteration of work.
- Charts provide a visual mechanism for teams to ensure they're continuing to improve. When the team can se data, it's easier to spot bottlenecks in the process (and remove them).
- Two common reports kanban teams use are control charts and cumulative flow diagrams.
- A control chart shows the cycle time for each issue as well as a rolling average for the team.

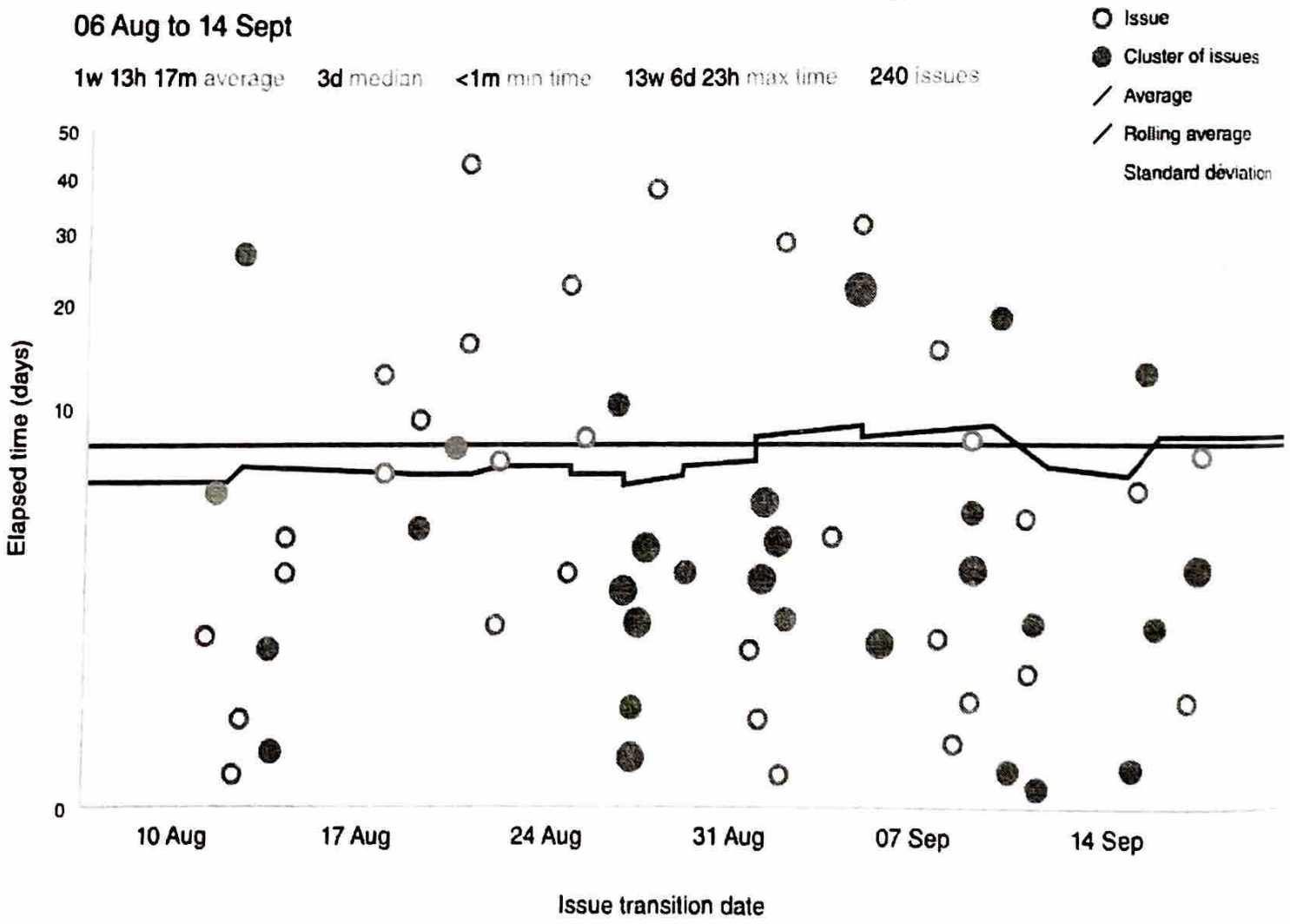


Fig. 3.5.2 : Control chart

- A cumulative flow diagram shows the number of issues in each state. The team can easily spot blockages by seeing the number of issues increase in any given state. Issues in intermediate states such as "In Progress" or "In Review" are not yet shipped to customers, and a blockage in these states can increase the likelihood of massive integration conflicts when the work does get merged upstream.

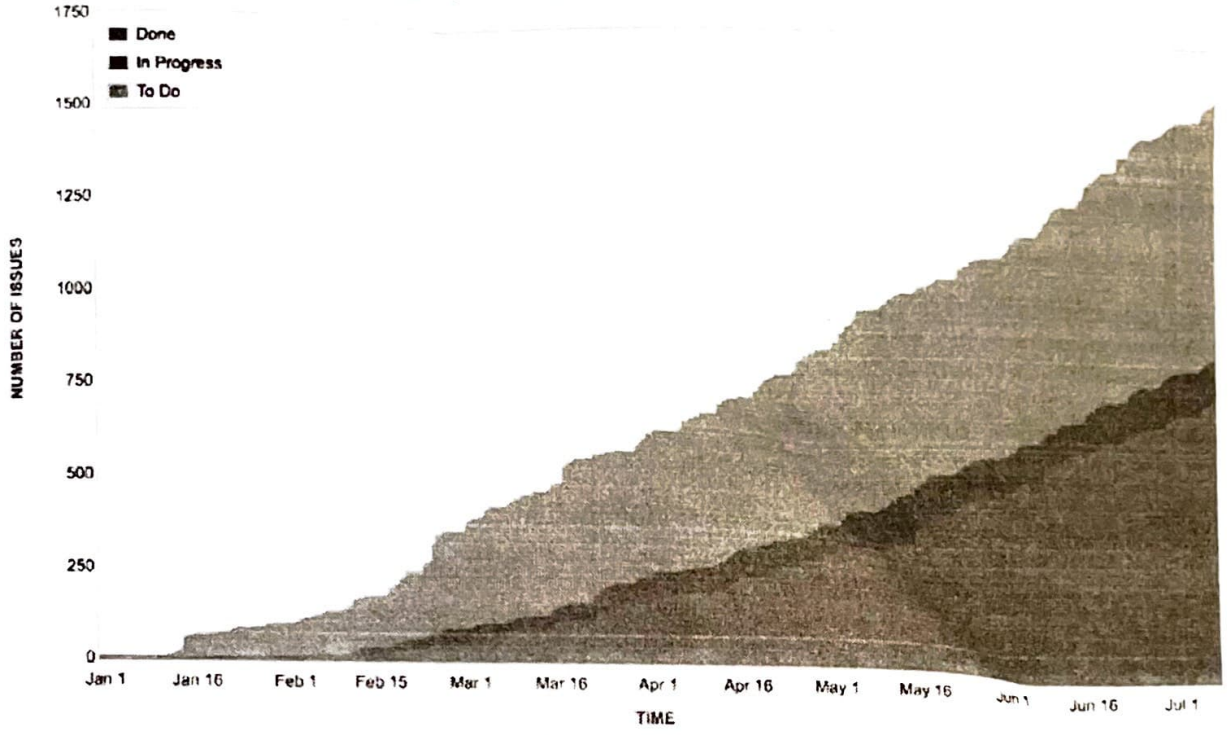


Fig. 3.5.3 : Cumulative flow diagram

- **Continuous delivery :** We know that continuous integration–the practice of automatically building and testing code incrementally throughout the day–is essential for maintaining quality. Now it's time to meet continuous delivery (CD). CD is the practice of releasing work to customers frequently–even daily or hourly. Kanban and CD beautifully complement each other because both techniques focus on the just-in-time (and one-at-a-time) delivery of value.
- The faster a team can deliver innovation to market, the more competitive their product will be in the marketplace. And kanban teams focus on exactly that: optimizing the flow of work out to customers.

3.5.4 Comparison between Kanban and Scrum

Kanban	Scrum
In Kanban, there is continuous flow.	In Scrum approach, we have regular fixed length sprints (i.e. 2 weeks)
Continuous delivery or at the team's discretion.	Delivery at the end of each sprint if approved by the product owner.
Cycle time is the key metrics in Kanban approach.	Velocity is the key metrics.
Changes can happen at any time.	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learning around estimation.