UNIT I

CHAPTER 3

Advanced Process Models and Tools

Syllabus

Agile software development: Agile methods, Plan-driven and agile development, Extreme programming Practices, Testing in XP, Pair programming. Introduction to agile tools: JIRA, Kanban, Case Studies: An information system (mental health-care system), wilderness weather system.

3.1 Agile Process Model

Review Question

Q. Explain Agile process model. What are the different agility principles ?

- 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

SPPU - May 13, Feb. 16

University Questions

Q. Compare the agile and evolutionary process models with a specific process model for each. **(May 2013, 8 Marks)**

Q. Compare the process between the Agile and evolutionary model. **(Feb. 2016, 4 Marks)**

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 more complete versions of the software.
2.	It can accommodate changing requirements.	It yields rapid development of more complete version of software 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 design, 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 prototype quickly. Thus inappropriate operating system or programming language may be used simply because it's available and known. Thus inefficient algorithm may be implemented.

Syllabus Topic : Agile Software Development

3.2 Agile Software Development

Review Question

Q. Explain agile development. What are the key factors that are considered for agile development ?

- In today's modern world, businesses spread across the globe in each sphere of life. It has become the global phenomenon. Due to this changing environment, the client must respond to new opportunities and to the volatile 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 changing 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 the stipulated time.
- In fact, in such rapidly changing market condition, rapid development of the software and its urgent delivery is 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 the 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 to gather complete set of requirements. The requirements elicited by the customers in the beginning are always changed since the customers find it difficult to predict the actual working of the software and the challenges to come across.
- Thus understanding the requirements quickly and accommodate the requirements change narrated by the customer quickly is the need of the time. Otherwise, delayed delivery of the software may make it unusable and 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.

Syllabus Topic : Agile Methods

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 and 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 much common.

3.2.3 Agility Principles

SPPU – Apr. 17

University Question

Q. List the principles of Agility.

(April 2017, 3 Marks)

- Agile software development describes a set of principles for software development under which requirements 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 it 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 entire 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 behaviour accordingly.
- Thus the principles of agility may be applied to any software process. To achieve agile development, the team 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.

Syllabus Topic : Plan-Driven and Agile Development

3.3 Plan-Driven and Agile Development

- In agile development approaches, the design and implementation are considered as the main activity in the software development process.
- In addition to that, they also use activities like requirement specification, testing, debugging and maintenance.
- But in a plan-driven approach, output associated with stage is also identified. The output obtained in one stage is used as the basis of planning for the later stages of the software process.

- Fig. 3.3.1 exhibits comparison between plan-driven and agile approaches to the requirement specification :

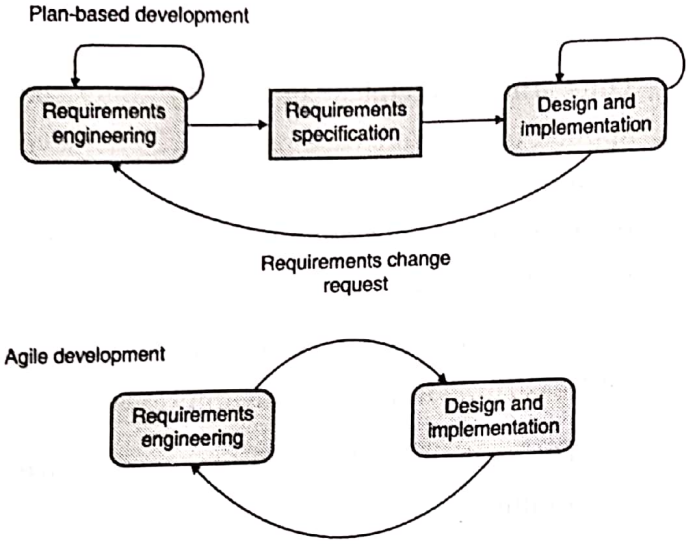

Fig. 3.3.1 : Distinction between plan-driven and agile approaches to the requirement specification

- In **plan-driven approach**, the iteration may occur within each of the activities and they are communicated with the help of formal documentation. After this formal communication, final **"Requirement Specification"** will be produced.
- In contrast, for an **agile approach** the iteration occurs across different activities. Observe the Fig. 3.3.1, the requirement specification and design are developed together to produce final product.
- As discussed earlier, the agile approach employes incremental development and delivery. Thus there is a series of incremental deliveries or we can say that series of versions for the software application under consideration.
- In general practice, so many software organizations have used agile methods and also use agile practices and they have integrated these with their plan-driven processes.

3.3.1 Comparison between Plan-driven and Agile Development

Sr. No.	Plan-driven development	Agile development
1.	In a plan-driven approach, output associated with stage is also identified. The output obtained in one stage is used as the basis of planning for the later stages of the software process.	In agile development approaches, the design and implementation are considered as the main activity in the software development process.
2.	In plan driven development, Iteration occurs within each of the activities.	In agile development, iteration occurs across different activities.
3.	Communication between different activities done with the help of formal documentation and after this communication, final "Requirement Specification" will be produced	The requirement specification and design are developed together to produce final product.
4.	Plan-driven software development uses structure to control risk.	Agile software development uses flexibility to control risk.
5.	Plan-driven emphasizes formal communications and control.	Agile emphasizes continual informal communications and an ability to react to changes and uncertainty.
6.	It is less adaptive in nature.	It is more adaptive in nature.
7.	Examples are : Traditional project management, Unified Process.	Examples are : SCRUM, Extreme Programming

Syllabus Topic : Extreme Programming Practices

3.4 Extreme Programming Practices

Q22

SPPU –Dec. 15, Feb. 16

University Questions

Q. What is Extreme Programming ? **(Dec 2015, 7 Marks)**

Q. What are the different key strategies used for Xtreme Programming approach to software development ? **(Feb. 2016, 6 Marks)**

- 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. 3.4.1 shows the Extreme Programming process and all the key XP activities are summarized :

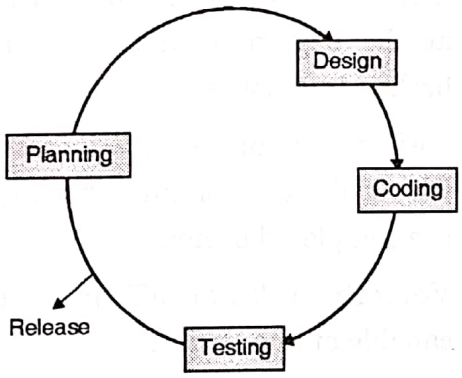

Fig. 3.4.1 : Extreme programming process

3.4.1 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 other stake holder and with customers.

3.4.2 The XP Process

SPPU - Dec. 14, Dec. 15

University Question

Q. Explain the Extreme Programming Process. **(Dec 2014, Dec. 2015, 6 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 for 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 always welcome over the complicated ones.
- **Coding** starts after the design work is over. Once the code is completed, it can be unit-tested immediately. The important concept during the coding activity in XP recommends that two people work together at one computer to create the code.
- **Testing** - All the individual unit tests are organized into a 'universal testing suite'. Integration and validation testing of the system can occur on daily basis. XP acceptances test, also called customer tests are specified by customer and focus on overall system features and functionality.

3.4.3 Scrum

Review Questions

Q. What is concept of SCRUM? Explain ?
Q. Explain scrum process flow.?
Q. What are different roles in scrum ?
Q. Explain scrum cycle.
Q. What is product backlog ?
Q. Explain sprint planning meeting with diagram.
Q. What do you mean by sprint backlog ?
Q. Explain daily scrum meeting.
Q. What do you mean by sprint review and retrospective?

- It is one of the agile software development methods. It is an iterative and incremental software development 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 that during project development customers can change their requirement and requirements of customers are unpredictable.
- It adopts empirical approach. It assumes problem cannot be fully understood or defined but focusing on 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.
  - **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.

3.4.3.1 Process Flow

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

➢ **Setup**
  - Setup environment contains the following :
  - Remove any customization from existing application.
  - Activate scrum process pack plug-in.
  - Assign scrum roles to users.

➢ **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.

➢ **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.

➢ **Create release**

It has start and end date in which number of development iterations are completed.

➢ **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.

➢ **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 and 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.

➢ **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.

➢ **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.

3.4.4 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.

3.4.5 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	

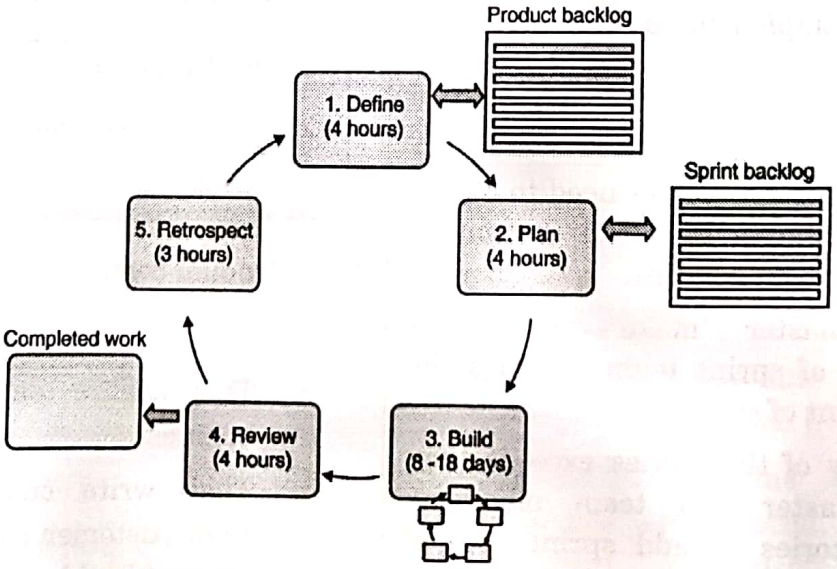

Fig. 3.4.2 : SCRUM cycle

1. **Define** : During defining the requirement generally scrum team, scrum master and product owner sits together. They decide the priority of team for duration of sprint. It is useful in high level direction for the team.
2. **Plan :** It is about selecting items from product backlog. It also evaluates value of different items and estimate the efforts needed by scrum team. Generally items are selected from product backlog which can be considered as sprint backlog.
3. **Build :** Task from the sprint backlog is selected for development. They go on selecting the items till the 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 the improvement area by the scrum team. They also discuss on the success of earlier iterations.

3.4.6 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 date 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.

3.4.7 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. 3.4.3.

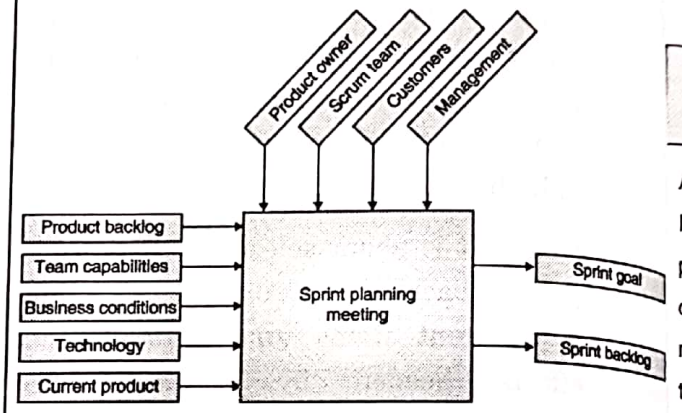

Fig. 3.4.3 : Sprint planning meeting

3.4.8 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		

3.4.9 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.

3.4.10 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
Daily scrum 9:15 - 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
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 1 retrospective 11 am - 12 pm Celebration lunch 12 pm - 1 pm		

Monday	Tuesday	Wednesday	Thursday	Friday
		Sprint 2 planning 1 pm - 5 pm		

3.4.11 Maintaining Sprint Backlog and Burn Down Chart

- Burn down is graphical representation o work left versus time.
- Pending work is on vertical axis and time i at horizontal axis.
- It is nothing but the run chart o outstanding work.
- It is useful in predicting when the work wil 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. 3.4.4.

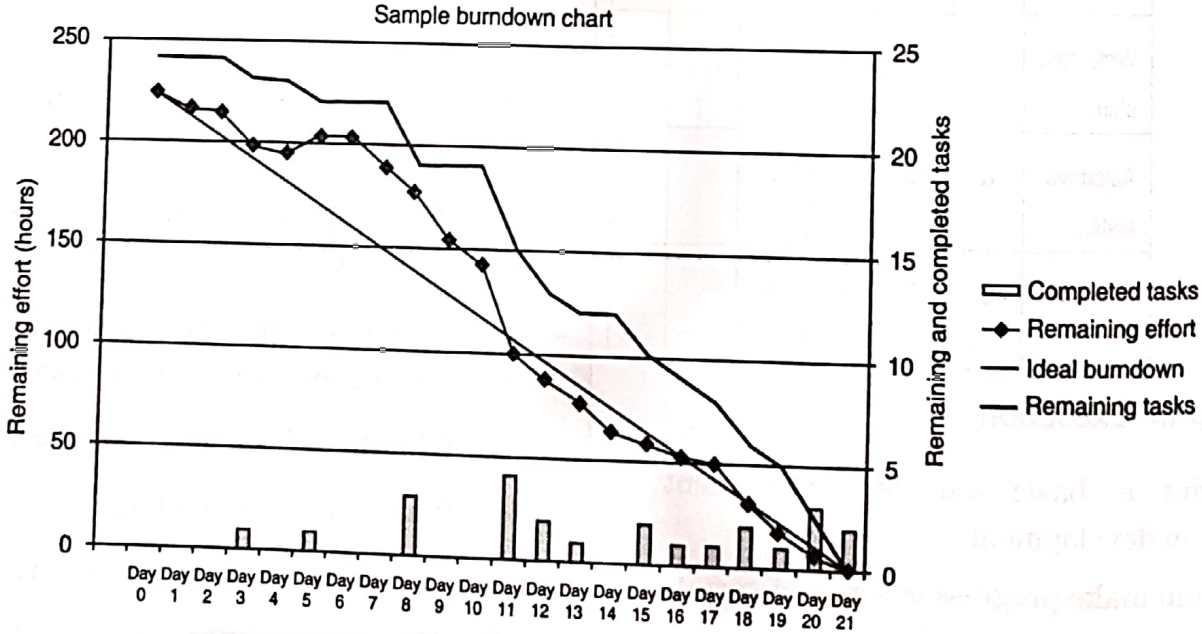

Fig. 3.4.4 : burn down chart

3.4.12 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.

Syllabus Topic : Testing in XP

3.5 Testing in XP

- Before the start of coding, the unit test is created in XP approach. This creation of unit test is the key element of XP approach.
- This unit test must be implemented by using some framework. This framework enables them to become automated.
- All the individual unit tests are organized into a 'universal testing suite'. Integration and validation testing of the system can occur on daily basis.
- XP acceptances test, also called customer tests are specified by customer and focus on overall system features and functionality.

3.5.1 Exploratory Testing Versus Scripted Testing

Review Question

Q. Compare Exploratory testing with scripted testing in the context of agile software development.

- Scripted testing considers two roles namely test designer and tester. Test designer is usually designs the test cases. He is high skilled specialist. Tester executes the test case which is designed by test designer.
- In exploratory testing different roles are not considered. Test designing and actual testing is done by same person. Tester designs the test cases and executes it. According to previous results he may create more test case and execute it.
- Learning, test design and execution all are done by tester in exploratory testing.
- In script testing team can be formed of different employees where one of the members would be very high skilled one and other members can beginners. Cost of testing can be saved with script testing.
- In exploratory testing depends upon skill of the employees which results in high project cost.
- With scripted testing high planning is possible. We can predict for desired outcome with scripted testing. As scenarios are ready we execute the test cases precisely.
- In exploratory testing planning is impossible or very difficult. We cannot guarantee all the test cases will be executed or not.

- With scripted testing well documentation can be created. Test designer can document the test scenarios whereas tester can records the status of executed test. With exploratory very less documentation is used.
- Exploratory testing is flexible whereas script testing is not.
- If some changes happen in the requirement then test designer need to change the scenarios. When scenarios are changed then only tester can executes the test cases.
- Exploratory testing is interesting than that of scripted testing.

3.6 Agile Practices

- Agile development is supported by number of practices. It includes the different area like requirement analysis, design, coding, testing, project management, process, quality, etc. Some of the important agile practices are as follows :
  - Pair Programming
  - Refactoring
  - Test Driven Development
  - Continuous Integration

Syllabus Topic : Pair Programming

3.6.1 Pair Programming

Review Questions

Q. Explain Pair programming in the context of agile software development.

Q. What are various benefits of pair programming ?

Q. What do you mean by Remote pair programming ?

- It is one of agile practice is agile software development.
- In this method two programmer work together on one machine.
- One programmer types the code. Anot programmer observes point and sugge the changes.
- Sometimes programmer can switch th role to each other.
- Observer programmer can sugg improvement in code.
- With this method programmer who wri the code can concentrate on efficient codi Observer programming can think a suggest improvements.

Benefits of pair programming

There are numerous advantages of pa programming.

- **Economy :** It spends around 15% mo time on programming. But design code is efficient than that of individu coding. It also improves developme time and support cost.
- **Design quality :** With pair desi more ideas can be generated. As t programmers are working together are able to check all possibilities duri design. This can be accomplished different programmer bring their pri experience together. They thi differently and able to generate m ideas.
- **Satisfaction :** With pair programmi programmer can enjoy work than th of individual programming. They more confident in their solutions. Th confidence results in success software.
- **Learning :** Ideas are shared among programmer. They are able to le from each other. They are able to sh their knowledge with each other. allows programmer to obse programming code and give feedback accordingly.

- **Team building and communication:** It allows to share the problem with each other and able to find the solution on it. With this practice communication among team member increase with benefits in greater team building.

Remote pair programming

- It is part of pair programming.
- It allows programmer to be geographically apart from each other but connected through the network.
- It is also called as virtual pair programming or distributed pair programming.
- They generally work using shared editors, shared desktop, remote pair programming tools.
- Sometimes this method creates the problems which are not present in face to face programming.
- Some of the problem is remote pair programming are delay in communication, communication link failure problem, lack of coordination, etc.
- Different tools are provided for remote pair programming. Some of the tools are as follows :
  - Microsoft Lync
  - VNC
  - Screenhero
  - Skype
  - Terminal multiplexers (tmux a, screen -x)
  - Gobby
  - Saros
  - Xpairtise
  - Floobits
  - Visual studio anywhere
  - Cloud9

3.6.2 Refactoring

Review Questions

Q. Explain Refactoring in the context of agile software development. What are different refactoring techniques?

Q. What are different editors and IDE that supports automated refactoring ?

- It is one of the agile practices in agile software development.
- It is process of restructuring the existing code.
- The changes are made without affecting external behaviour.
- It does not change the functional requirement of the code.
- It changes the non-functional requirement of the code. Non functional requirements are useful in efficiency, performance, throughput, etc.
- It has numerous advantages like improved readability, reducing complexity, reusability, etc.
- It results in creating more expressive software architecture which results in extensibility.
- Generally series of standards are applied for code refactoring.
- Some standards are used which are designed by organization and some other standards also used in code refactoring.
- They are generally identified by code smell.
- E.g. we have code which is very long or it is almost duplicate of another code. In this situation code refactoring can be applied where we can remove the code duplication. After refactoring the basic functionality of the code remain same.
- Code duplication can be removed by designing shared function. Shared by will be used whenever required instead of creating same copy of code again and again.
- Two main objectives of refactoring are maintainability and extensibility.

- With code maintainability, we can easily identify the bugs in the software as the code is properly documented.
- With code extensibility, new features can be added very easily.
- We need automatic unit test before applying code refactoring.

List of refactoring techniques

- Some of famous refactoring techniques we will discuss in this topic.
- Some techniques are applied to only certain languages or situations.
- Many development environments provide code refactoring.

Some of the techniques are as follows.

- **Techniques allowing more abstraction**
  - Code encapsulation where different fields can be accessed with getter and setter methods.
  - Code sharing can be achieved by generalizing types. If we make generalize code then chances of sharing that code increases.
  - Replace type checking code with states.
  - Use polymorphism.
- **Techniques for breaking code apart into more logical pieces**
  - Break the code into different modules. Modules increase the code reusability.
  - Extracting class results in moving code to new code.
  - Extracting methods results into moving methods to new methods.
- **Techniques for improving name and location of code**
  - Move methods or field to appropriate class.
  - Change the name of methods or fields to user friendly name. Name should signify the purpose of method or variable.
  - Use concept of superclass from object oriented programming.
  - Use concept of subclass from object oriented programming.

Automated code refactoring

Some editors and IDE supports automated refactoring. Some of the software are listed below.

- IntelliJ IDEA
- WebStorm
- Eclipse
- Netbeans
- JDeveloper
- Visual studio
- ReSharper
- Code rush
- Visual assist
- Photran
- Xcode
- AppCode

3.6.3 Test Driven Development

Review Question

Q. Explain test driven development (TDD) with block diagram. What are different phases of TDD ?

- It is one of the agile practices in agile development.
- This software development process relies on repetition of short development cycles.
- Changes in software are quick and easy.
- Short development cycle includes writing automated test case which defines desired improvement or new function. Minimum coding is done to pass the test. Afterwards code are redesigned as per the standard.
- It is related to extreme programming.

Test driven development is shown in Fig. 3.6.1.

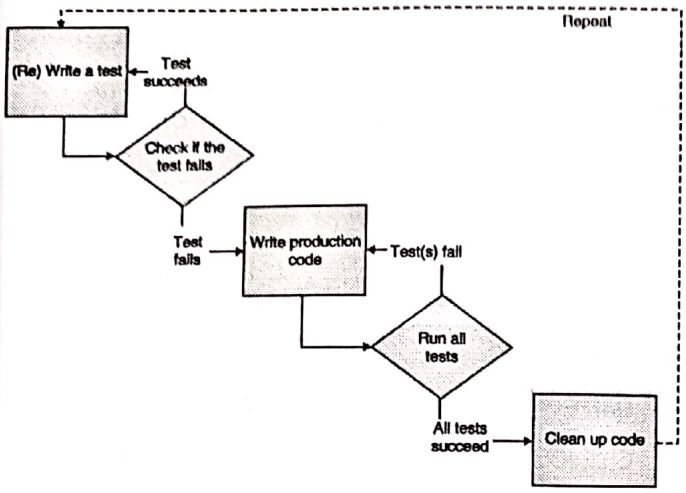

Fig. 3.6.1 : Test driven development

- Different phases of test driven development are as follows.

(a) Create test

- In test driven development new feature can be added by creating test.
- This test should get fail because this feature was not added previously.
- Before writing test, before should understand the test properly then only test should be added.
- Requirements can be understood using different techniques like use cases and user stories.

(b) Run test and see it fails

- In this phase we run the created test and check if it running properly or not.
- Test should get fail so that we can write code for it.

(c) Writing some code

- In this phase we write code for test to get pass.
- The code written is not perfect but it is useful to pass the test for time being.
- Afterwards this code will be improved.
- Purpose of this code is just for passing the test and not for permanent purpose.

(d) Run test

We run the test on created code.

(e) Refactor code

- Earlier we have written code which was not final.
- In this phase, code will improved and made efficient to fulfill all the coding standard.

(f) Repeat

- The process is repeated for another test also.
- Size of steps should be small.
- If the test fails then programmer need to undo the changes.

3.6.4 Continuous Integration

Review Questions

Q. Explain Continuous integration in the context of agile software development. What are various principles of Continuous integration.

Q. List different advantages of Continuous integration ?

Q. List Software tools which supports continuous integration ?

- It is method of merging all developer copies of work together frequently.
- It is part of extreme programming.
- Main aim is to prevent integration problems.
- It is useful in iterative software development.
- Earlier it was used in coordination with test driven development.

Principles of continuous integration

- **Maintain code repository :** All the project source code should be properly stored in code repository. It is useful in revision control system.

- **Automate the build :** There should be facility for building the integration automatically. Single command should able to do all the activities. Different build tools like make can be used. Automated build includes integration.
- **Make build self testing :** After making the integration software should work as per specification. Testing facility after integration should be automated one.
- **Everyone commits to baseline every day :** With this facility every committer can reduce conflicting changes.
- **Every commit should be built :** System should build commits to current working version.
- **Keep the build fast :** Building process should be rapid.
- Other principles are as follows.

  Test in clone of the production environment

  Make it easy to get the latest deliverables

  Everyone can see the results of the latest build

  Automate deployment

Advantages

It has many advantages which are as follows.

- If unit test fails, developer can revert the code to bug free state of the code.
- Developers can detect and fix integration problem continuously.
- Early warning of broken or incomplete code.
- They can give early warning of conflicting changes.
- Code is available constantly.
- Immediate feedback can be given to developers
- Creating modules can be possible wit continuous integration.

Software tools which supports continuou integration

- Apache Continuum
- Apache Gump
- AutomatedQA Automated Build Studio
- Atlassian Software Systems Bamboo
- Buildbot
- BuildHive
- CABIE
- Cascade
- CruiseControl
- CruiseControl.rb
- CruiseControl.NET
- Electric Cloud
- FinalBuilder Server
- Hudson
- Jenkins
- IBM Rational Team Concert
- IBM Rational Software SCLM
- TeamCity
- TinderboxTravis CI
- Xcode 5

3.7 Agile Modeling (AM)

Review Question

Q. Discuss the significance of Agile Modelin (AM). What are different modeling principle that make agile modeling unique ?

- There are many situations where th software developer build very larg software systems used for some busines applications and other critical softwar system. The software developer shoul model the complexity of such system so tha he should be able to define the scope of th system. The software developer do so fo the following reasons :

  All the component of the system should b understood properly

  - The actual problem can be divided int various partition to solve and merge i into a single unit
  - The quality of the software i evaluated at each step during it construction.

- In the last three decades, various modelling technique have been put forwarded for analysis and design at both architectural level and component level.
- All these methods have profound impact and are difficult to apply. This method once applied is difficult to sustain over the life of software project.
- In order to handle this deficiency, some rigid techniques are required so that the projects can be handled intellectually.
- The best alternative to this approach is agile modelling. The agile modelling is defined as the practice based methodology that is very effective in modelling documentation of various software systems.
- The agile modelling is actually a collection of design principles and practices for modelling and it is applied to software system.
- In actual practice, the agile models are more effective and lightweight as compared to traditional models.
- In addition to agile manifesto the agile modelling suggest following techniques :
  - **Model with a purpose :** A software developer using agile modelling must have a proper and specific goal in his mind before he start developing a project .
  - **Use multiple model :** The software developer is free to use to various model and notation to explain a software in a better way. The agile modelling suggests different aspect of system and provides the value to the intended end-users.
  - **Travel light :** Agile modelling also suggest to use the models that provide long terms values and must be able to accommodate to the changes occurring at the later stages i.e. the developer must used light weight model.
  - **Content is more important than presentation :** In agile modelling the contents are very important for the intended users. Based on presentation only, the work can not be accomplished
  - **Know the model and the tools you are using to create the software :** While using agile methodology a software developer must know the strength and weaknesses of each of the models and tools.
  - **Adapt locally :** The agile Modelling approach must be adaptable to the needs of the team.

Syllabus Topic : Introduction to Agile Tools : JIRA

3.8 Introduction to Agile Tools : JIRA

- JIRA is an agile tool used for issue tracking and project management by over 25,000 customers in 122 countries in the world.
- JIRA lets you prioritize, assign, track, report and audit your issues from software bugs and help-desk tickets to project tasks and change requests.
- JIRA is offered in three packages :
  - JIRA Core includes the base software.
  - JIRA Software is intended for use by software development teams and includes JIRA Core and JIRA Agile.
  - JIRA Service Desk is intended for use by IT or business service desks.
- The main features of JIRA for agile software development are the functionality to plan development iterations, the iteration reports and the bug tracking functionality.
- JIRA is a commercial software product that can be licensed for running on-premises or available as a hosted application. Pricing depends on the maximum number of users.

3.8.1 JIRA platform

- Every JIRA application (JIRA Software, JIRA Service Desk, and JIRA Core) is built on the JIRA platform. The JIRA platform provides a set of base functionality that is shared across all JIRA applications, like issues, workflows, search, email, and more.
- You can extend and modify this functionality via the integration points provided by the JIRA platform, including the JIRA REST APIs, webhooks, plugin modules, etc.
- The pages in this section will help you learn about the JIRA platform and how to develop for it. You'll find information on JIRA's architecture and JIRA add-ons, guides to developing for different parts of JIRA, JIRA Data Centre, and more.

3.8.2 JIRA Architecture

- JIRA is a web application written in Java. It is deployed as a standard Java WAR file into a java Servlet Container such as Tomcat.
- JIRA uses Open Symphony's WebWork 1 to process web requests submitted by users. Please note that WebWork 1, not 2, is used. WebWork 1 is a MVC framework similar to Struts. Each request is handled by a WebWork action which usually uses other objects, such as utility and Manager classes to accomplish a task.

JIRA uses JSP for the View layer. So most of HTML that is served to the user as the response to their web request is generated by a JSP. Therefore, to generate a response the WebWork action uses a JSP.

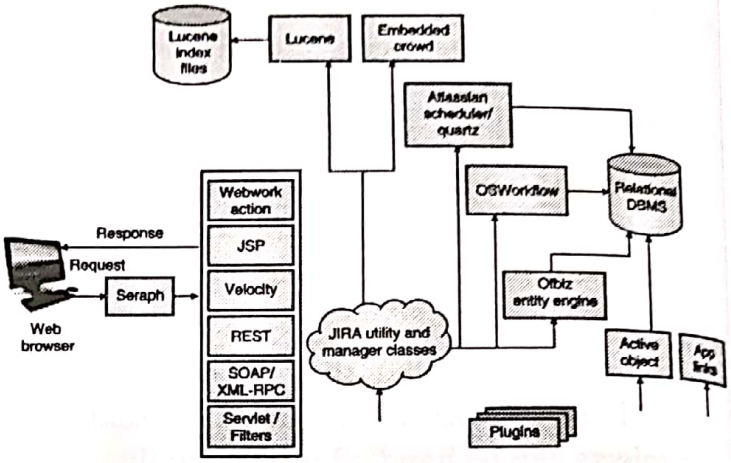

Fig. 3.8.1 : JIRA architecture

3.8.3 JIRA Database Schema

- To generate schema information for the JIRA database, e.g. PDF file, follow the instructions below. You can generate schema information in pdf, txt and dot formats. Note, if you want to generate the schema in PDF format, you need to have Graph viz installed.

(1) Download the attached plugin:

- For JIRA 5: JIRA-schema-diagram-generator-plugin-1.0.jar
- For JIRA 6: JIRA-schema-diagram-generator-plugin-1.0.1.jar
- For JIRA 7: JIRA-schema-diagram-generator-plugin-1.1.0.jar

(2) Install the plugin in your JIRA instance by following the instructions on Managing JIRA's Plugins.

(3) **Database Schema**

- Go to the JIRA administration console and navigate to System > Troubleshooting and Support > Generate Schema Diagram
- (tick)Keyboard shortcut: g + g + start typing generate
- Enter the tables/columns to omit from the generated schema information, if desired

- If you want to generate a pdf, enter the path to the Graph viz executable.
- Click Generate Schema.
- The 'Database Schema' page will be displayed with links to the schema file in txt, dot and pdf format.

3.8.4 JIRA Mobile Connect

- JIRA Mobile Connect (JMC) is an iOS library that can be embedded into any iOS app to provide following extra functionality:
  1. **Real-time crash reporting**, have users or testers submit crash reports directly to your JIRA instance.
  2. **User or tester feedback** views that allow users or testers to create bug reports within your app.
  3. **Rich data input**, users can attach and annotate screenshots, leave a voice message, and have their location sent.
  4. **Two-way communication with users**, thank your users or testers for providing feedback on your app.

3.8.5 JIRA Applications

- The JIRA family of applications currently consists of JIRA Software, JIRA Service Desk, and JIRA Core. A JIRA application is built on the JIRA platform, and extends and modifies the base functionality provided by the JIRA platform.
- For example, the JIRA Software application provides software development features that are not part of the JIRA platform, such as agile boards, linked development tool information (e.g. commits, builds, etc), and software project templates.

3.8.6 JIRA APIs

- The JIRA platform provides both Java APIs and REST APIs that you can use to interact with JIRA programmatically. These APIs are common to all JIRA applications.
- In addition, JIRA Software and JIRA Service Desk provide APIs for application-specific functionality. For example, the JIRA Software REST API has methods for creating sprints, creating boards, retrieving epics, etc.

Syllabus Topic : Introduction to Agile Tools - Kanban

3.9 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 understand the basic principles.
- Unlike implementing kanban on a factory floor, which would involve changes to physical processes and the addition of substantial materials, the only physical things a software teams need are a board and cards, and even those can be virtual.

3.9.1 Kanban Boards

- The work of all kanban teams revolves around a kanban board, a tool used to visualize work and optimize the flow of the work among the team.
- While physical boards are popular among some teams, virtual boards are a crucial feature in any agile software 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 work is visualized, their workflow is standardized, and all blockers and dependencies are immediately identified and resolved.
- A basic kanban board has a three-step workflow: To Do, In Progress, and Done. However, depending on a team's size, structure, and objectives, the workflow can be mapped to meet the unique process of any particular team.
- The kanban methodology relies upon full transparency of work and real-time communication of capacity, therefore the kanban board should be seen as the single source of truth for the team's work.

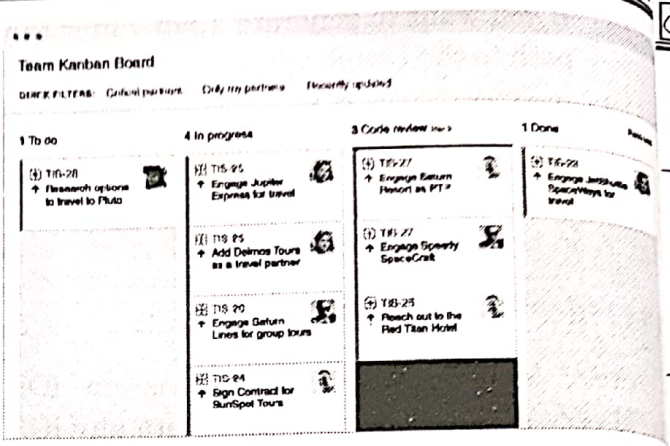

Fig.3.9.1 : Kanban boards

3.9.2 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.

3.9.3 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 more 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 limits 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, and 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, rather 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 work 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 and effectiveness with every iteration of work.
- Charts provide a visual mechanism for teams to ensure they're continuing to improve. When the team can see 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.

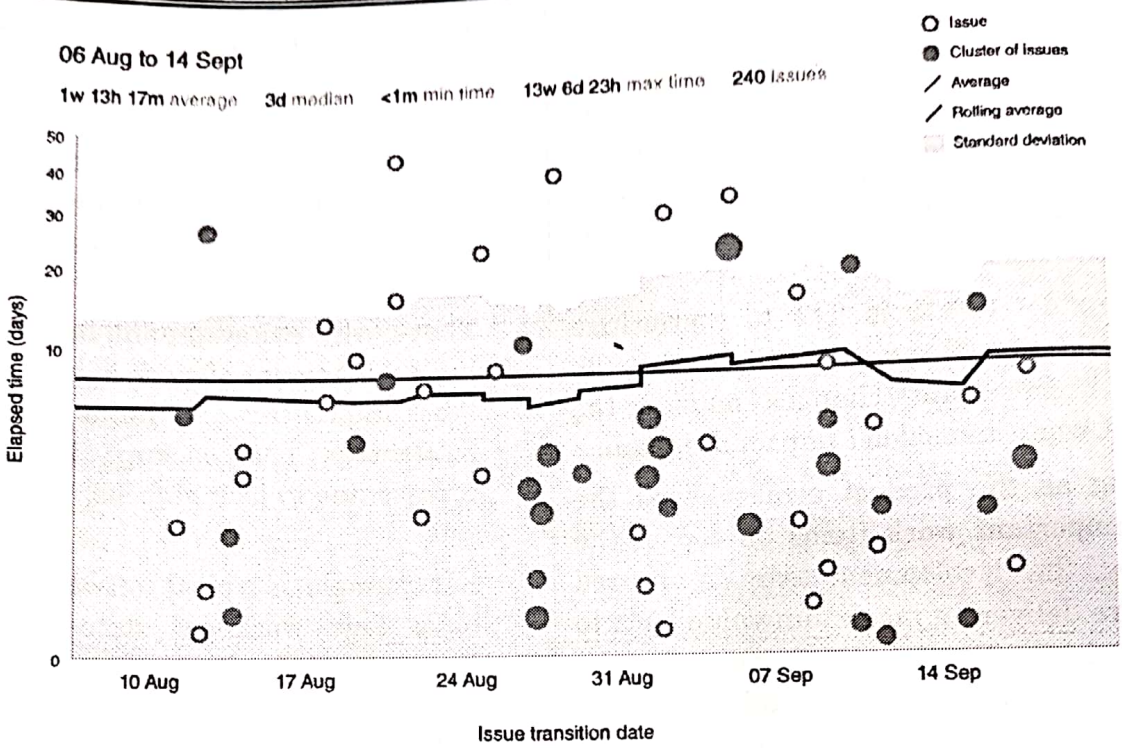

Fig. 3.9.2 : Control chart

- A cumulative flow diagram shows the number of issues in each state. The team can easily sp blockages by seeing the number of issues increase in any given state. Issues in intermedia 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 do get merged upstream.

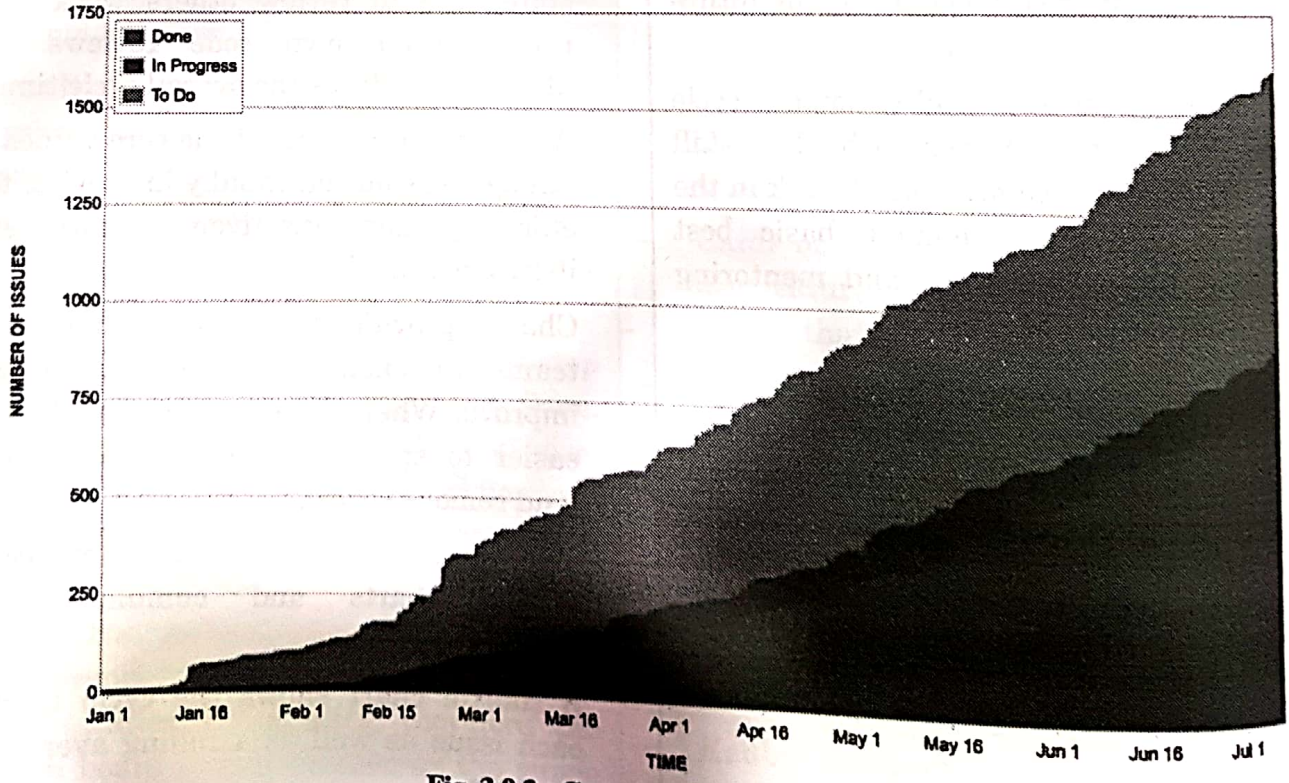

Fig. 3.9.3 : Cumulative flow diagram

- **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.

3.9.4 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.
Roles : No existing roles. Some teams enlist the help of an agile coach.	**Roles :** Product owner, scrum master, development team etc.

Syllabus Topic : Case Study : An Information System(Mental Health-care System)

3.10 Mental Health-Care System

- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received. Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems. To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centers.
- The MHC-PMS (Mental Health Care-Patient Management System) is an information system that is intended for use in clinics. It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity. When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected. The system is not a complete medical records system so does not maintain information about other medical conditions. However, it may interact and exchange data with other clinical information systems. Figure 1.6 illustrates the organization of the MHC-PMS.
- The MHC-PMS has two overall goals:
  1. To generate management information that allows health service managers to assess performance against local and government targets.
  2. To provide medical staff with timely information to support the treatment of patients.
- The nature of mental health problems is such that patients are often disorganized so may miss appointments, deliberately or accidentally lose prescriptions and

medication, forget instructions, and make unreasonable demands on medical staff. They may drop in on clinics unexpectedly. In a minority of cases, they may be a danger to themselves or to other people. They may regularly change address or may be homeless on a long-term or short-term basis. Where patients are dangerous, they may need to be 'sectioned'—confined to a secure hospital for treatment and observation.

Users of the system include clinical staff such as doctors, nurses, and health visitors (nurses who visit people at home to check on their treatment). Nonmedical users include receptionists who make appointments, medical records staff who maintain the records system, and administrative staff who generate reports.

The system is used to record information about patients (name, address, age, next of kin, etc.), consultations (date, doctor seen, subjective impressions of the patient, etc.), conditions, and treatments. Reports are generated at regular intervals for medical staff and health authority managers. Typically, reports for medical staff focus on information about individual patients whereas management reports are anonymized and are concerned with conditions, costs of treatment, etc.

The key features of the system are :

1. **Individual care management :** Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors who have not previously met a patient can quickly learn about the key problems and treatments that have been prescribed.
2. **Patient monitoring :** The system regularly monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected. Therefore, if a patient has not seen a doctor for some time, a warning may be issued. One of the most important elements of the monitoring system is to keep track of patients who have been sectioned and to ensure that the legally required checks are carried out at the right time.
3. **Administrative reporting :** The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

- Two different laws affect the system. These are laws on data protection that govern the confidentiality of personal information and mental health laws that govern the compulsory detention of patients deemed to be a danger to themselves or others. Mental health is unique in this respect as it is the only medical speciality that can recommend the detention of patients against their will. This is subject to very strict legislative safeguards. One of the aims of the MHC-PMS is to ensure that staff always acts in accordance with the law and that their decisions are recorded for judicial review if necessary.
- As in all medical systems, privacy is a critical system requirement. It is essential that patient information is confidential and is never disclosed to anyone apart from authorized medical staff and the patient themselves. The MHC-PMS is also a safety-critical system. Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.

- The overall design of the system has to take into account privacy and safety requirements. The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients. There is a potential conflict here, privacy is easiest to maintain when there is only a single copy of the system data. However, to ensure availability in the event of server failure or when disconnected from a network, multiple copies of the data should be maintained.

Syllabus Topic : Case Study : Wilderness Weather System

3.11 Wilderness Weather System

- To help monitor climate change and to improve the accuracy of weather forecasts in remote areas, the government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.

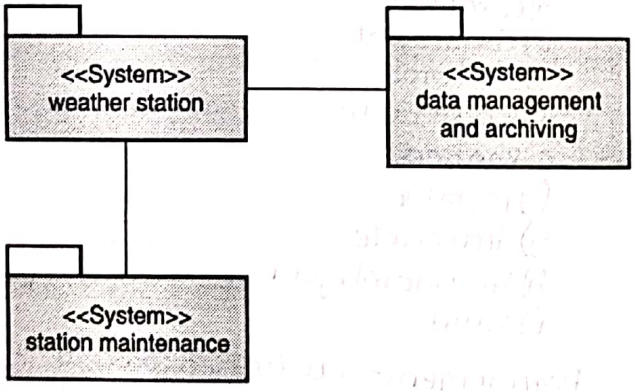

Fig. 3.11.1 : The weather station's environment

- These weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed, and wind direction.
- Wilderness weather stations are part of a larger system (Fig. 3.11.1), which is a weather information system that collects data from weather stations and makes it available to other systems for processing. The systems in Fig. 3.11.1 are:

1. **The weather station system :** This is responsible for collecting weather data, carrying out some initial data processing, and transmitting it to the data management system.
2. **The data management and archiving system :** This system collects the data from all of the wilderness weather stations, carries out data processing and analysis, and archives the data in a form that can be retrieved by other systems, such as weather forecasting systems.
3. **The station maintenance system :** This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems. It can update the embedded software in these systems. In the event of system problems, this system can also be used to remotely control a wilderness weather system.

- In Fig. 3.11.1, I have used the UML package symbol to indicate that each system is a collection of components and have identified the separate systems, using the UML stereotype «system». The associations between the packages indicate there is an exchange of information but, at this stage, there is no need to define them in any more detail.
- Each weather station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure, and the rainfall over a 24-hour period. Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.

- The weather station system operates by collecting weather observations at frequent intervals—for example, temperatures are measured every minute. However, because the bandwidth to the satellite is relatively narrow, the weather station carries out some local processing and aggregation of the data. It then transmits this aggregated data when requested by the data collection system. If, for whatever reason, it is impossible to make a connection, then the weather station maintains the data locally until communication can be resumed.
- Each weather station is battery-powered and must be entirely self-contained—there are no external power or network cables available. All communications are through a relatively slow-speed satellite link and the weather station must include some mechanism (solar or wind power) to charge its batteries. As they are deployed in wilderness areas, they are exposed to severe environmental conditions and may be damaged by animals. The station software is therefore not just concerne with data collection. It must also :
  1. Monitor the instruments, power, an communication hardware and repo faults to the management system.
  2. Manage the system power, ensurin that batteries are charged whenev the environmental conditions perm but also that generators are shut dow in potentially damaging weath conditions, such as high wind.
  3. Allow for dynamic reconfiguratio where parts of the software a replaced with new versions and whe backup instruments are switched in the system in the event of syste failure.
- Because weather stations have to be sel contained and unattended, this means th the software installed is complex, eve though the data collection functionality fairly simple.

Requirement Engineering.
functions:
a) Inception - where to start from?
b) Elicitation - feedback basically
c) Elaboration - detailed work.
d) Negotiation - discussion on issues.
e) Specification - final work product
f) Validation - check with expectations
g) Requirement management -

Functional Requirements.
a) How the system must react to aiaput
b) System reaction in some situations
c) Do nothing in some situations

Non-Functional Requirements.
a) Reliability
b) response time
c) System performance
d) System security
e) System availability
f) Portability.
g) Robustness
h) Ease otuse.

Characteristics of SRS
a) Complete
b) Consistent
c) Accurate
d) Modifiable
e) Ranked
f) Testable
g) Traceable
h) ~~AA~~ Unambiguous (1 meaning)
i) Valid.

Requirements elicitation: process
a) collaborative requirements gath
b) Quality function deployment
c) User scenarios
d) Work product