1) Three prespectives

Ans :

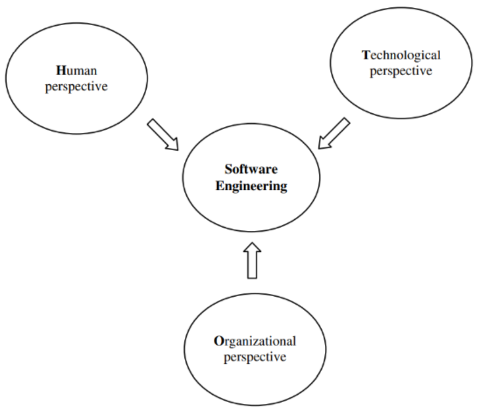


Figure 1.1 *The HOT analysis framework for software engineering.*

A software development method should address not only the technological aspects, but also the work environment and the professional framework. Accordingly, agile software engineering is reviewed in this book from the following three perspectives:

- The **H**uman perspective, which includes cognitive and social aspects, and refers to learning and interpersonal (teammates, customers, management) processes. **H**OT
- The **O**rganizational perspective, which includes managerial and cultural aspects, and refers to the workspace and issues that extend beyond the team. H**O**T
- The **T**echnological perspective, which includes practical and technical aspects, and refers to how-to and code-related issues. HO**T**

Specifically, we explain how the attention that agile software development gives these aspects helps cope with the challenges of software projects. To highlight this multifaceted perspective of the agile approach, we introduce the Human, Organizational, and Technical (HOT) analysis scale for software development. See Figure 1.1 for a schematic view of the HOT analysis framework.

2) AGILE MANIFESTO AND FOUR PRINCIPLES (OR) WHY AGILE IS IMPORTANT ?

ANS :

The Agile Manifesto is a document that identifies four key values and 12 principles that its authors believe software developers should use to guide their work

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 over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

igure 1.2 *Manifesto for agile software development.*

1.5.1 Individuals and Interactions over Processes and Tools

This principle encourages us to focus on the individuals involved in the development process, rather than on the process and/or the tools. Specifically, this principle encourages software developers to give high priority to the people who participate in the development process, as well as to their interaction and communication when they develop, interact, think, discuss, and make decisions with respect to different issues related to the software development process and environment. In other words, according to this principle, one of the first considerations that should be taken into account when a decision related to the development process is made, is the influence of the decision's outcome on the people who are part of the development environment as well as on their relationships and communications.

1.5.2 Working Software over Comprehensive Documentation

This principle affirms that the main goal of software projects is to produce quality software products. This idea has three main implications.

First, agile software development focuses on the development itself and the creation of only those documents that are needed for the development process. Some of these essential documents, depending on their characteristics and usefulness, may be posted on the wall of the agile collaborative workspace so that they will be accessible to *all* the project stakeholders *all the time*.

Second, agile software development processes start the product's actual development (that is, coding) as soon as possible, in order to get some sense of the developed product. This early development enables both the teammates and the customers to improve their understanding of the developed product and to proceed with the development process on a safer ground.

Third, from the customers' perspective, this principle helps ensure that customers get a bugless high quality system that meets their requirements. This, of course, has direct implication for the quality-related activities that agile teams perform.

As can be seen, this principle supports the first principle of the Agile Manifesto by binding the people who participate in the development process with the actual development process. Such a connection inspires a culture in which software quality is one of the main values.

1.5.3 Customer Collaboration over Contract Negotiation

This principle changes the perception of the customer's role in software development. It encourages agile software developers to base their work on ongoing and daily contact with the customer. Such a close contact enables customers to cope successfully with the changes that characterize software projects.

Human interrelationships, mainly between customer and management, are emphasized by this principle of the manifesto. These interrelations in turn have direct implications on the development team, which should employ specific practices to ensure this kind of relationship and communication. Such practices, when employed on a daily basis, directly influence the culture of agile organizations.

This principle also points to a conceptual change with respect to the nature and formulation of software product contracts.

Thus, by dealing with contract- and communication-related issues aimed at ensuring that the customer gets the desired product, this principle of the Agile Manifesto further supports the second principle of the Agile Manifesto.

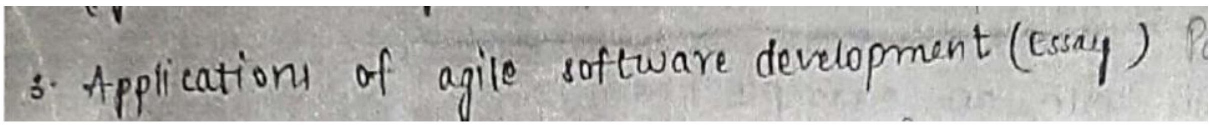
1.5.4 Responding to Change over Following a Plan

This principle encourages agile software developers to establish a process that copes successfully with changes introduced during development, without compromising the high quality of the developed product. The rationale for this principle is the recognition that customers cannot predict a priori all their requirements; therefore, a process should be established in which the requirements as gradually understood by the customer can be shared with the teammates. Accordingly, agile software development allows the introduction of changes in the developed product which have emerged from an improved understanding of the software requirements, without necessarily increasing the cost of development. This process is explained mainly in Chapter 3, Customers and

Agile Principles

1. Satisfy Customers Through Early & Continuous Delivery
2. Welcome Changing Requirements Even Late in the Project
3. Deliver Value Frequently
4. Business people and developers must work together daily throughout the project
5. Build Projects Around Motivated Individuals
6. Communicate Face-to-face
7. Working Software is the Primary Measure of Progress
8. Maintain a Sustainable Working Pace
9. Continuous attention to technical excellence
10. Simplicity
11. Self-organizing Teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

---

3) 
3. Applications of agile software development (Essay)

ANS :
1)Whole team.
2) Short releases.
3) Time estimations
4) Measures.
5) Customer collaboration
6) Test-driven development.
7) Pair programming
8)Refactoring.

Whole team. The practice of Whole Team means that the development team (including all developers and the customer) should communicate in a face-to-face fashion as much as possible. The practice is applied in several ways.

First, the development team is located in a collaborative workspace—a space which supports and facilitates communication. Second, all team members participate in all the product presentations to the customer, hear the customer's requirements, and are active in the actual planning process (see Chapter 3, Customers and Users). Third, participants that traditionally belong to separate teams (e.g., testers and designers), are integrated into the development team and process.

In the Whole Team concept each team member has an additional role besides that of developer (see Chapter 2, Teamwork). The rationale for this role distribution is that one person, usually the team leader, no matter how skilled he or she is, cannot handle in a professional manner all of the essential and complex responsibilities involved in software development. The distribution of responsibilities in the form of roles helps to control and manage these responsibilities. In addition,

Short releases. Agile software development processes are based on short releases of about two months, divided into short iterations of one or two weeks, during which the scope of what is to be developed in that iteration is not changed. At the end of each iteration the software is presented to the customer and the customer provides feedback to the team and sets the development requirements for the next iteration.

The detailed plan of each short iteration is carried out during a full day—a Business Day—which is specifically allocated for this purpose at the beginning of each iteration (see Chapter 3, Customers and Users). In the Business Day all the project stakeholders participate—customer, team members, users, management representatives, representatives of related projects, and so on. The Business Day includes three main parts: a presentation of what was developed in the previous iteration, along with any relevant measures taken; a short reflective session in which the development process performed so far is analyzed and lessons are learnt; and the actual planning of the next iteration. At the end of the Business Day, a balanced workload is ensured among all team members.

Time estimations. In agile software development two important practices are performed with respect to time estimation. First, the teammate who is in charge of the development of a specific task also estimates the time needed for its

development; this practice increases the team member's responsibility and commitment to the project. Second, development tasks are formulated in a way that allows their time estimation to be set in hour resolution. This fact is important because the greater a development task is, the harder it is to estimate its development time, and vise versa: the smaller the time segment estimated, the more accurate its estimation is. Consequently, the development pace can be planned more precisely. This encourages a culture in which plans can be set and followed in such a way that deadlines should not be postponed.

Measures. In order to navigate the development process so that a high quality product is produced, agile software development processes are accompanied by various measures about which all the project stakeholders decide according to their needs. The importance attributed to these measures is expressed, among other means, by a special role—that of tracker—assigned to one team member, who is in charge of the measures.

Measures enable the team to improve the development process, and consequently, the developed software. Measures also convey the message that the development process should be monitored and that this monitoring should be transparent to all participating developers. Chapter 5, Measures, further elaborates on this practice.

Customer collaboration. Agile software development methods encourage the customer to become part of the development process. The goal is to get ongoing feedback from the customers and to proceed according to their needs. This avoids the need for speculation, which may lead to incorrect working assumptions.

Test-driven development. Test-driven software development encourages developers to build automatic unit and acceptance tests.

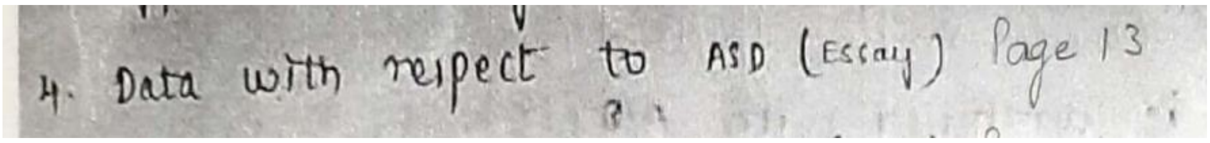
Unit tests are written prior to code writing in a gradual process: each step starts with writing a specific test case followed by adding a small functionality that lets the test be successful. This implies that thought must be given to the development task before actual coding starts. It also helps control the development process by clarifying what has been developed and tested so far.

Pair programming. This practice means that each code is developed by two teammates who sit together in front of a computer and in an interactive, communication-based process work on a specific development task (Williams et al. 2000). It is important to note that even though the development is carried out in pairs, there is personal responsibility for each development task.

During pair programming it is harder to be distracted, and hence pairs tend to remain focused on the development task. In addition, each task is characterized by two levels of abstraction: that of the driver—the one who works with the keyboard and thus thinks on a lower level of abstraction, and that of the navigator—the mate who can think about the development task on a higher level of abstraction.

The application of this practice implies that all team members will become familiar with all parts of the developed software and improve their comprehension of the entire project. This fact encourages a culture that is characterized by knowledge sharing.

Refactoring. The practice of refactoring encourages teammates to improve code readability without adding functionality. The mere inclusion of refactoring in agile software development acknowledges that one cannot, a priori, predict all development details; it therefore legitimates time dedicated to improving software readability without pressure to move on to the next developmental task. The


4. Data with respect to ASD (Essay) Page 13

Ans :

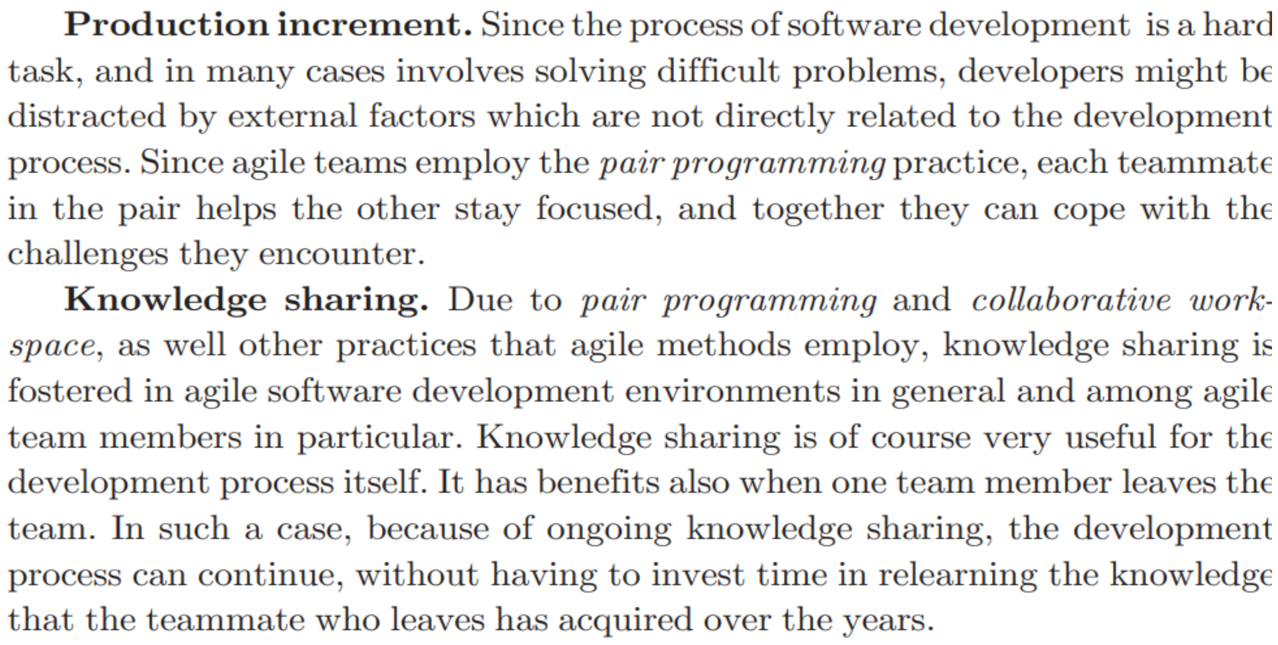
1) **Code is easier to work with.**
2) **Development is manageable and controlled.**
3) **The customer's needs are met.**
4) **Production increment.**
5) **Knowledge sharing.**

   The data about the reduction in time to market are now explained based on the different agile practices discussed in the previous section.

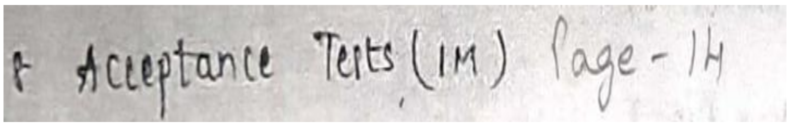
Code is easier to work with. It is easier to work with code developed through an agile process mainly due to the test-driven development, pair programming, and refactoring practices. These agile practices demonstrate that since software development is a complex process, early investment in the code quality results in code that is easier to work with in later stages.

Development is manageable and controlled. Due to the short iterations/releases practice, in each iteration only a small portion of the development tasks are dealt with, estimated, and developed. As has been mentioned before, this fact eases the time estimation and development processes. Since the estimation of small software chunks is more accurate and is made in hour resolution, it is easier to observe whether the development pace is kept or whether an adjustment should be made in the project plan.

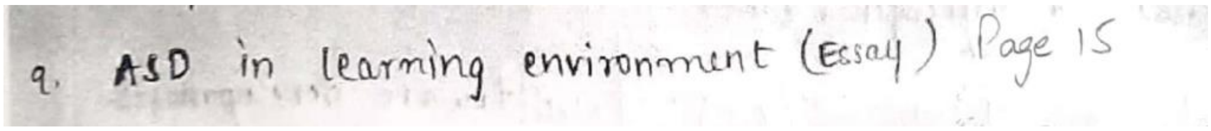
The customer's needs are met. Since the customer is involved in the project development for the entire process and shares with the development team his or her vision during the entire process, only what he or she articulates and needs is developed. Thus, the extra effort and time teammates sometimes invest in developing what they just suppose to be the customer's needs, without having a way to verify that this is indeed what is requested by the customer, are saved and put entirely into fulfilling and developing the customer's actual requirements.


Production increment. Since the process of software development is a hard task, and in many cases involves solving difficult problems, developers might be distracted by external factors which are not directly related to the development process. Since agile teams employ the *pair programming* practice, each teammate in the pair helps the other stay focused, and together they can cope with the challenges they encounter.

Knowledge sharing. Due to *pair programming* and *collaborative workspace*, as well other practices that agile methods employ, knowledge sharing is fostered in agile software development environments in general and among agile team members in particular. Knowledge sharing is of course very useful for the development process itself. It has benefits also when one team member leaves the team. In such a case, because of ongoing knowledge sharing, the development process can continue, without having to invest time in relearning the knowledge that the teammate who leaves has acquired over the years.

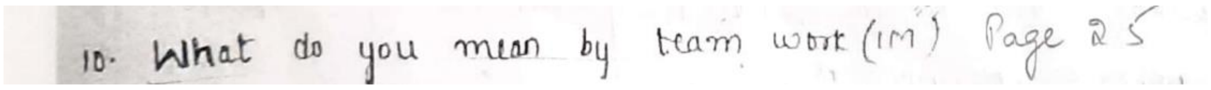

8 Acceptance Tests (1M) Page - 14

Ans:
acceptance testing, is **the final stage in the software testing process**. It is typically performed by the end-users or client to determine whether an application or feature fulfills its purpose. AT must be completed before the software can be released to the market.

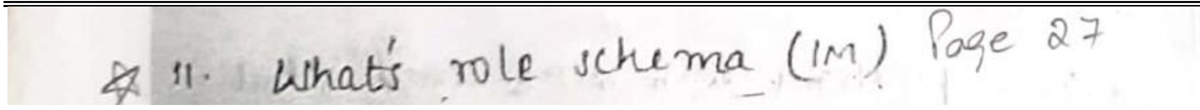

9. ASD in learning environment (Essay) Page 15

Ans : TEXT BOOK PAGE NO : 15 (matter)

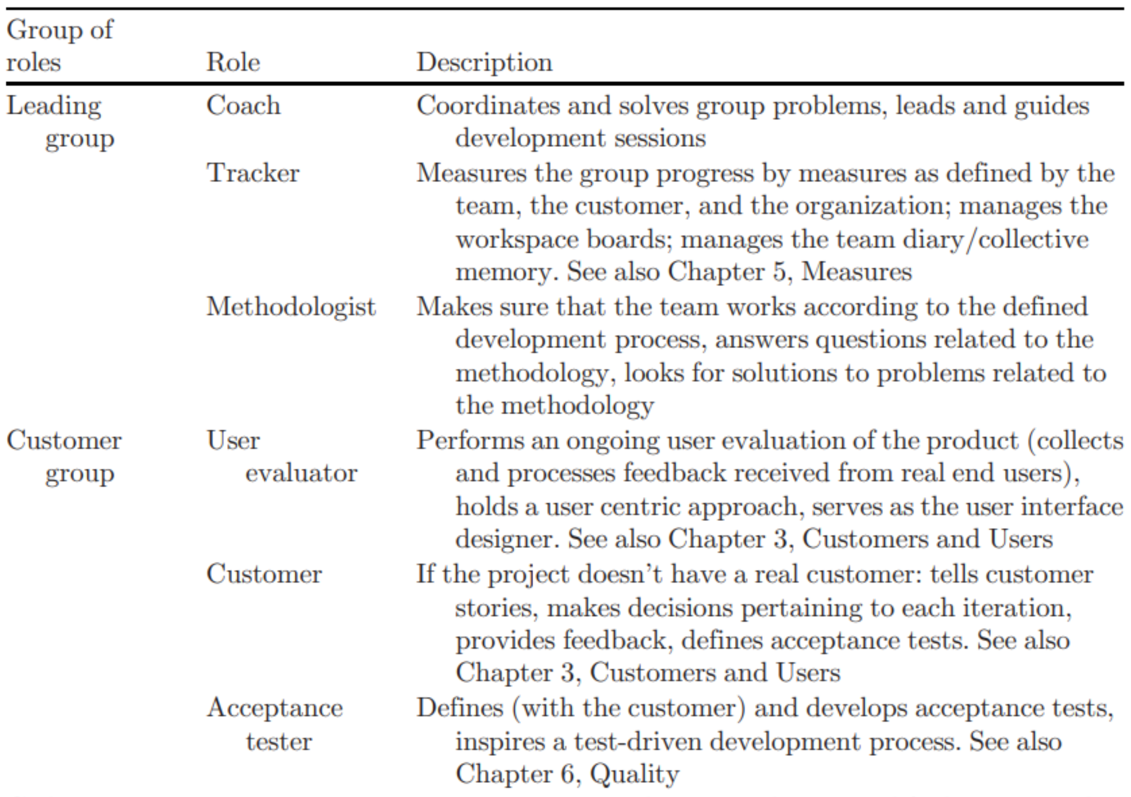
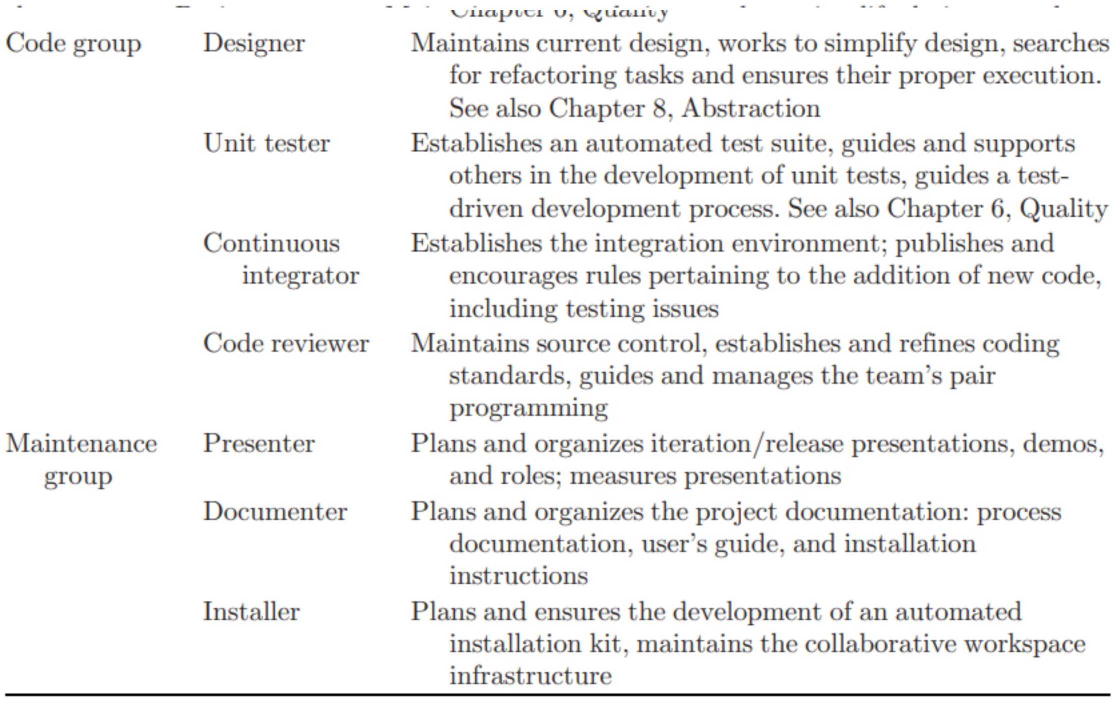
Elicite Communication
Teaching and Learning Principles :
1)Teaching and Learning Principle1: Inspire the Software Development Approach
2) Teaching and Learning Principle 2: Let the Learners Experience the Software Development Approach
The Studio Environment
The Academic Coach Role
Overview of the Studio Meetings
Launching the Project Development in the Studio
Team Forming


10. What do you mean by team work (1M) Page 25

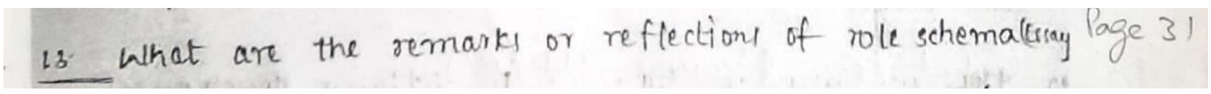
Ans :
TEAMWORK: It addresses how to build teams in a way that promotes team members' accountability and responsibility, and that fosters communication between teammates. One of the basic ways to start team building is by assigning roles to the team members. For this purpose a role scheme is presented in this chapter, according to which each team member is in charge of a specific managerial aspect of the development process, such as design and continuous integration, in addition to his or her development tasks.


☆ 11. What's role schema (1M) Page 27

Ans : (important)

Group of roles	Role	Description
Leading group	Coach	Coordinates and solves group problems, leads and guides development sessions
	Tracker	Measures the group progress by measures as defined by the team, the customer, and the organization; manages the workspace boards; manages the team diary/collective memory. See also Chapter 5, Measures
	Methodologist	Makes sure that the team works according to the defined development process, answers questions related to the methodology, looks for solutions to problems related to the methodology
Customer group	User evaluator	Performs an ongoing user evaluation of the product (collects and processes feedback received from real end users), holds a user centric approach, serves as the user interface designer. See also Chapter 3, Customers and Users
	Customer	If the project doesn't have a real customer: tells customer stories, makes decisions pertaining to each iteration, provides feedback, defines acceptance tests. See also Chapter 3, Customers and Users
	Acceptance tester	Defines (with the customer) and develops acceptance tests, inspires a test-driven development process. See also Chapter 6, Quality
Code group	Designer	Maintains current design, works to simplify design, searches for refactoring tasks and ensures their proper execution. See also Chapter 8, Abstraction
	Unit tester	Establishes an automated test suite, guides and supports others in the development of unit tests, guides a test-driven development process. See also Chapter 6, Quality
	Continuous integrator	Establishes the integration environment; publishes and encourages rules pertaining to the addition of new code, including testing issues
	Code reviewer	Maintains source control, establishes and refines coding standards, guides and manages the team's pair programming
Maintenance group	Presenter	Plans and organizes iteration/release presentations, demos, and roles; measures presentations
	Documenter	Plans and organizes the project documentation: process documentation, user's guide, and installation instructions
	Installer	Plans and ensures the development of an automated installation kit, maintains the collaborative workspace infrastructure

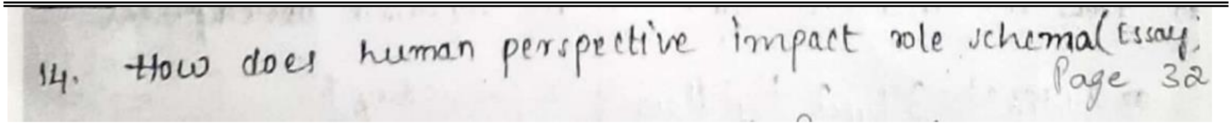


13 What are the remarks or reflections of role schema(essay) Page 31

ANS :

- The set of roles that a software development method includes in its role scheme reflects the values that a software development method attempts to inspire. Therefore, the role scheme that a software development method defines is one of the key elements of the method. Indeed, different agile methods suggest different role schemes that support their values and conception of software development processes.
- In addition to the role definitions presented in Table 2.1, several roles also support communication between the four groups. For example, the installer is also in charge of communication with the code group.
- At the first stages of the development process, or when the team is established, the role holders should learn their roles and establish a procedure that will enable them to perform their role properly. In the next stages of the agile project, role holders should maintain the spirit and the actual performance of the aspect that their role focuses on.
- When teams consist of fewer than twelve developers, several roles can be unified and assigned to one team member. There are different ways to unify roles, and each has its own advantages. In each case, however, the entire list of roles should be assigned and performed by all team members.

- The team can choose whether each of its members will specialize in one role for a long period of time or, alternatively, whether the roles will rotate among the team members. The exact way by which it is done in practice should be set by each team according to the team members' preferences. For example, it can be decided that roles are reassigned at the beginning of each release.
- Such a team organization eases project management, since it is clear who is in charge of what aspect of the development project, what aspect should be treated by whom, and who should be approached when a specific problem, which belongs to a specific aspect of the development process, arises. Even in cases when there are role overlaps, they will not interrupt the process. Sometimes they can even foster project development. One example is when the unit tester and the acceptance tester work together to introduce test-driven development.


14. How does human perspective impact role schema (Essay.
Page 32

ANS :

Social Aspect

- A personal role increases teammates' involvement, communication, accountability, responsibility, and commitment to the software development process and to their team.
- Team members wish to have a specific role in addition to their development tasks in order to increase their influence and involvement in the project management.

Cognitive Aspect

- Since each team member approaches the product from one specific perspective, each can focus on this one specific aspect without being distracted by the multifaceted nature of software product development. In other words, on the global level, the role definition encourages each team member to treat the software product from one perspective. Consequently, each gradually improves his or her understanding about that aspect.
- The role scheme supports the thinking of the development process on multiple levels of abstraction (see Chapter 8, Abstraction). Since abstraction is a key component of software development, every mechanism that supports team members' thinking in terms of different levels of abstraction should be enhanced. On the one hand, each team member sees his or her development task on a relatively low level of abstraction; and on the other hand, the personal role of each team member enables each of them to gain a global overview of the developed system on a higher level of abstraction. Agile methods support thinking at different levels of abstraction in additional ways, such as short releases (see Chapter 3, Customers and Users) and refactoring (see Chapter 8, Abstraction).

- The role scheme enhances knowledge distribution, since each team member specializes in one domain and shares his or her knowledge with the other team members. In addition, since the role scheme leads to knowledge distribution, no harm happens when one team member leaves the team. Indeed, he or she has gained expertise in his or her role; at the same time, however, parts of this knowledge have already been spread. Thus, if a team member leaves the team, the other team members have a reasonable amount of knowledge to continue with respect to that role.
- The role scheme supports the individual's professional development. Team members perform their roles and improve their role performance while learning the practice that their role represents. In turn, they become experts in the specific aspect of software development on which their personal role focuses. In addition, when a team member feels that he or she has exhausted one role's contribution to his or her professional development and wishes to hold another role in the team, as has already been mentioned, role rotation can take place.

---

15 Dilemmas in team work (Essay) Page 34

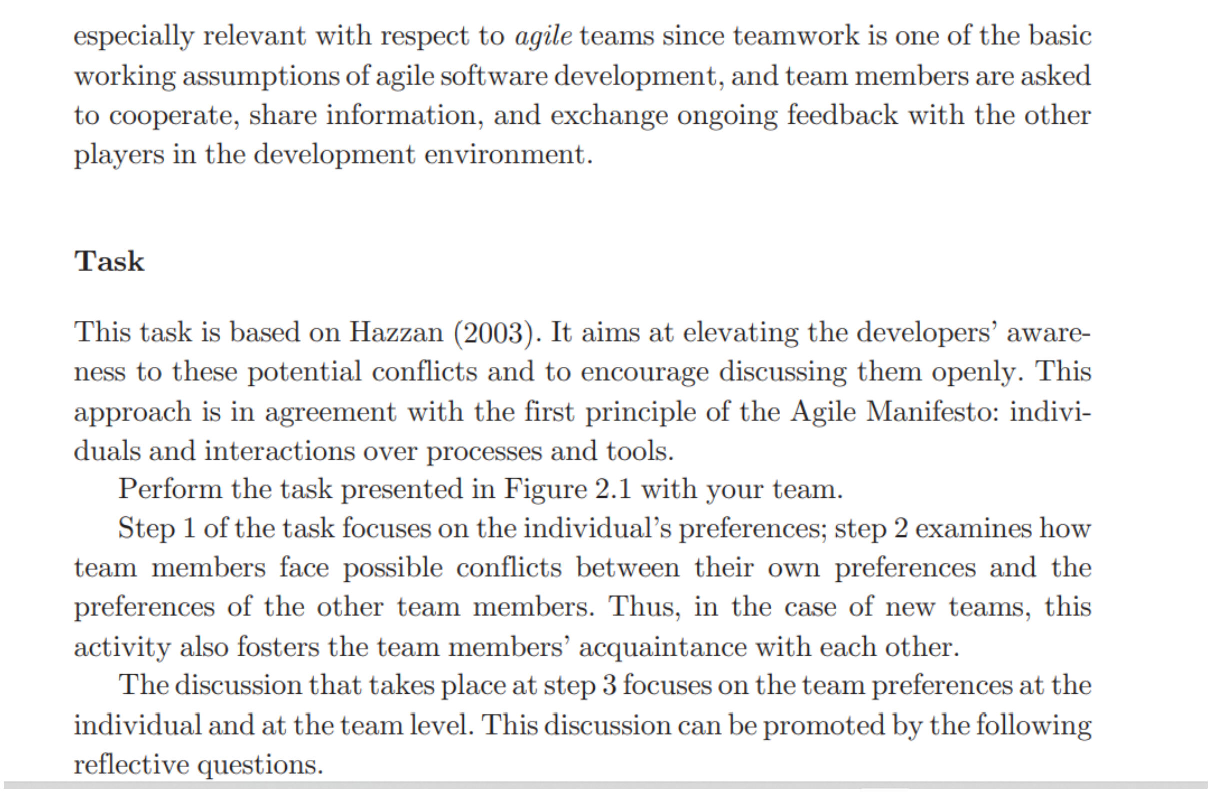
ANS :

2.5 Dilemmas in Teamwork

One of the problems that can arise with respect to teamwork is the question of how to allocate incentives, rewards, and bonuses among team members.

This question is relevant with respect to many professionals and kinds of institutions. However, reward allocation in software engineering is important mainly, but not only, because teamwork is essential in software development. As a result, conflicts between the required cooperation on the one hand, and one's desire to excel as an individual on the other, may intensify. The discussion is

\


especially relevant with respect to *agile* teams since teamwork is one of the basic working assumptions of agile software development, and team members are asked to cooperate, share information, and exchange ongoing feedback with the other players in the development environment.

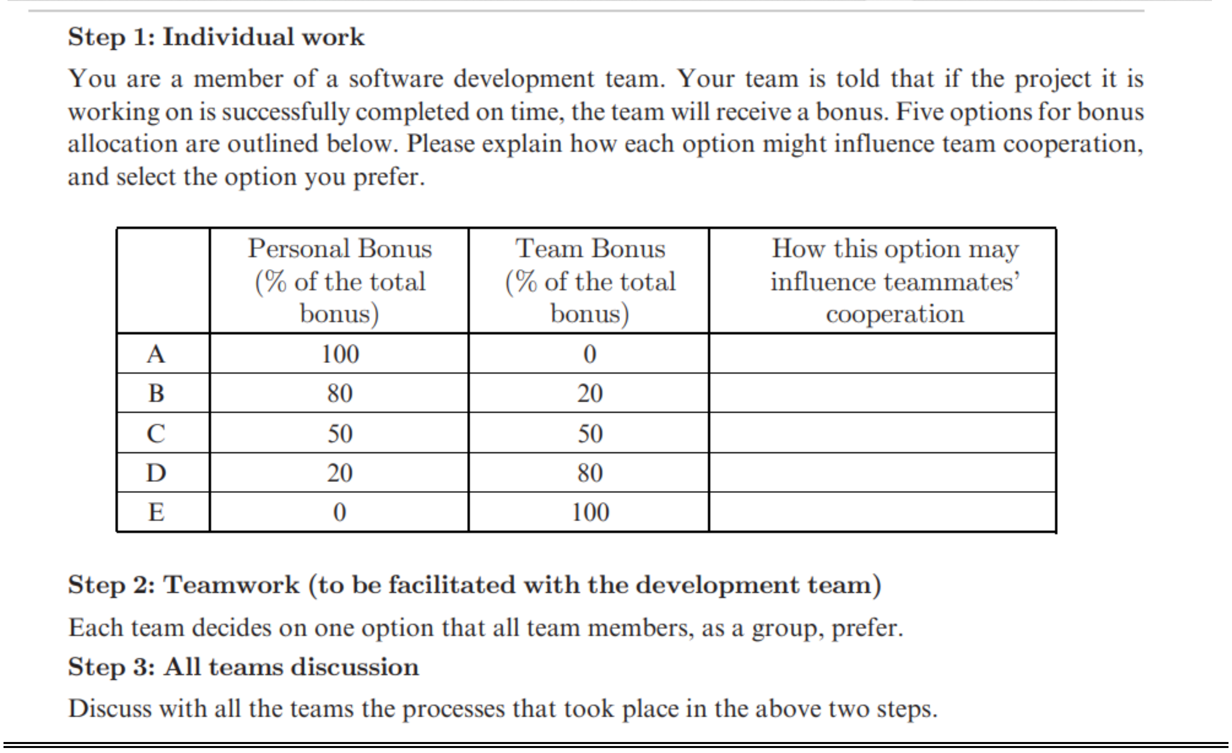
Task

This task is based on Hazzan (2003). It aims at elevating the developers' awareness to these potential conflicts and to encourage discussing them openly. This approach is in agreement with the first principle of the Agile Manifesto: individuals and interactions over processes and tools.

Perform the task presented in Figure 2.1 with your team.

Step 1 of the task focuses on the individual's preferences; step 2 examines how team members face possible conflicts between their own preferences and the preferences of the other team members. Thus, in the case of new teams, this activity also fosters the team members' acquaintance with each other.

The discussion that takes place at step 3 focuses on the team preferences at the individual and at the team level. This discussion can be promoted by the following reflective questions.


Step 1: Individual work

You are a member of a software development team. Your team is told that if the project it is working on is successfully completed on time, the team will receive a bonus. Five options for bonus allocation are outlined below. Please explain how each option might influence team cooperation, and select the option you prefer.

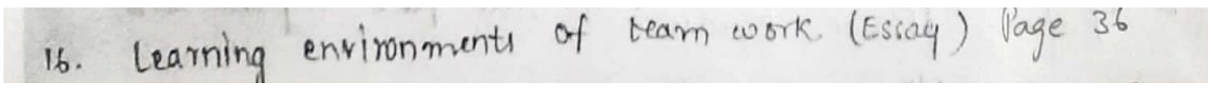
	Personal Bonus (% of the total bonus)	Team Bonus (% of the total bonus)	How this option may influence teammates' cooperation
A	100	0	
B	80	20	
C	50	50	
D	20	80	
E	0	100	

Step 2: Teamwork (to be facilitated with the development team)

Each team decides on one option that all team members, as a group, prefer.

Step 3: All teams discussion

Discuss with all the teams the processes that took place in the above two steps.


16. Learning environments of team work. (Essay) Page 36

ANS : TEXT BOOK PAGE NO : 36 (MATTER)

Teaching and Learning Principles :

1) Assign Roles to Team Members.

Role Activities:

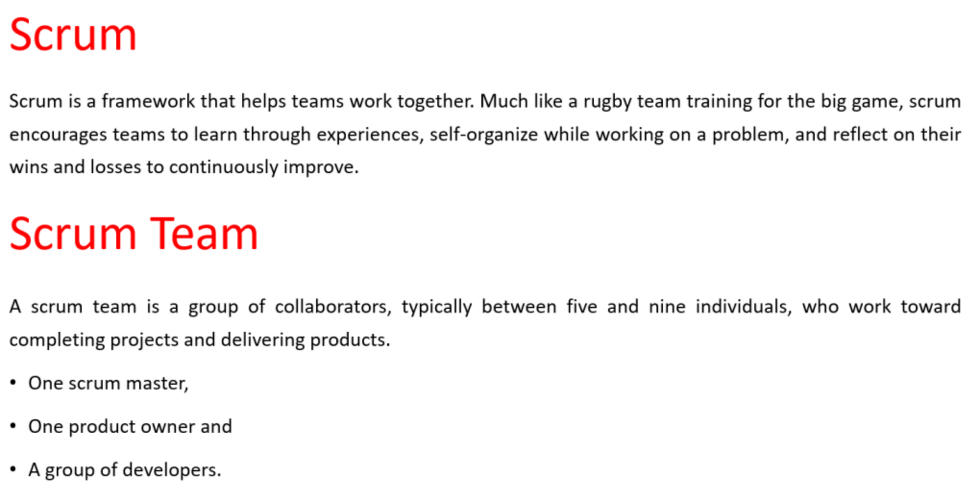
Role Assignment Activities

Role Maintenance Activities :

1) Stand-up meeting:
2) Presentations to customers:
3) Feedback after presentations:

Role Improvement Activity

What is Scrum ? and Scrum Team ?



Scrum

Scrum is a framework that helps teams work together. Much like a rugby team training for the big game, scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.

Scrum Team

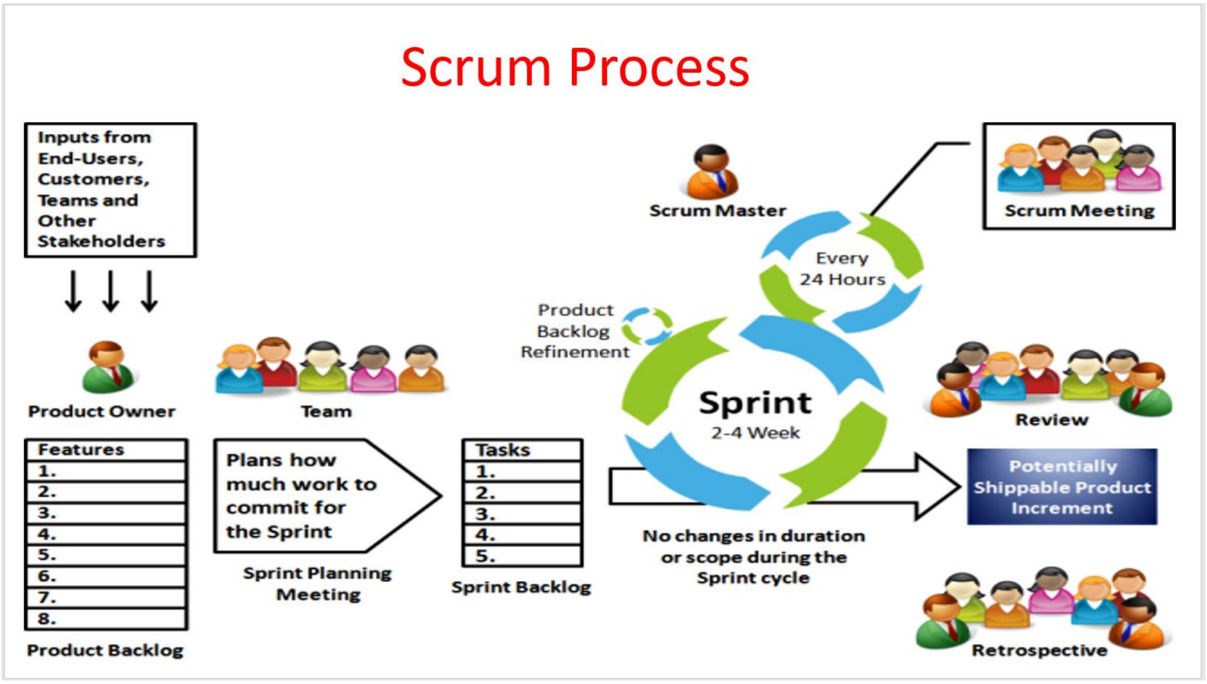
A scrum team is a group of collaborators, typically between five and nine individuals, who work toward completing projects and delivering products.

- One scrum master,
- One product owner and
- A group of developers.

Explain the process of scrum and with a case study ?

Ans :

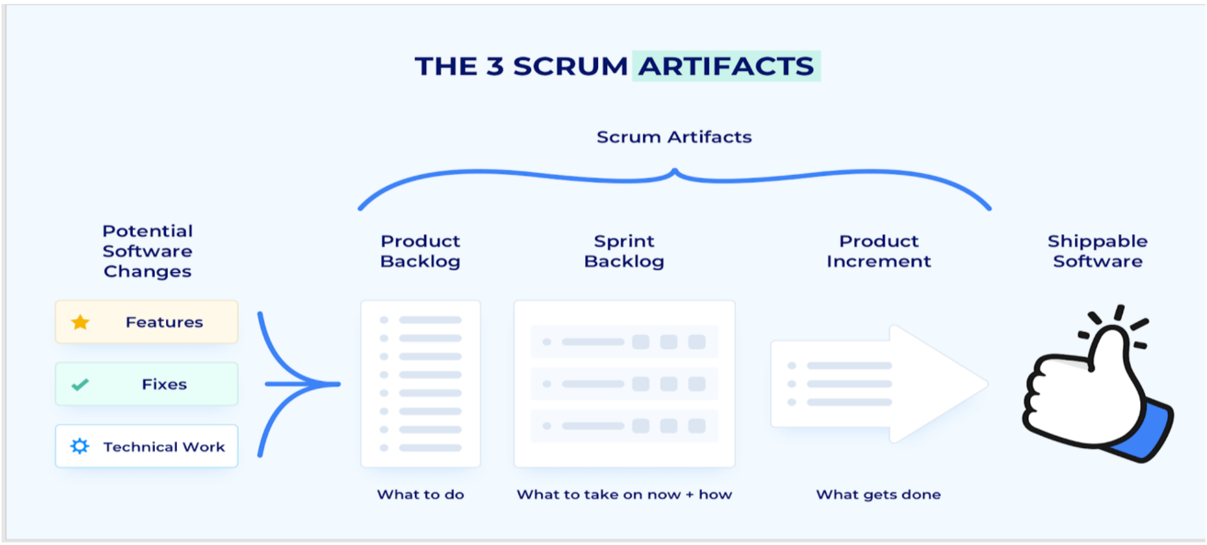
- A Sprint is a short, time-boxed period when a scrum team works to complete a set amount of work.
- A Product backlog is a prioritized list of work for the development team that is derived from the roadmap and its requirements. The most important items are shown at the top of the product backlog so the team knows what to deliver first
- A Sprint backlog is a subset of the product backlog and lists the work items to complete in one specific sprint. The purpose of the sprint backlog is to identify items from the product backlog that the team will work on during the sprint.


Scrum Process
Inputs from End-Users, Customers, Teams and Other Stakeholders
Product Owner
Team
Scrum Master
Scrum Meeting
Every 24 Hours
Product Backlog Refinement
Sprint
2-4 Week
Review
Features
1.
2.
3.
4.
5.
6.
7.
8.
Product Backlog
Plans how much work to commit for the Sprint
Sprint Planning Meeting
Tasks
1.
2.
3.
4.
5.
Sprint Backlog
Potentially Shippable Product Increment
No changes in duration or scope during the Sprint cycle
Retrospective

Scrum Events

1. Sprint Planning
2. Daily Scrum
3. Sprint Review
4. Sprint Retrospective
5. The Sprint

explain this in own words !



Case study examples :

Non functional requirments :

Security –

The system uses SSL (secured socket layer) in all transactions that include any confidential customer information.

The system must automatically log out all customers after a period of inactivity.

Reliability –

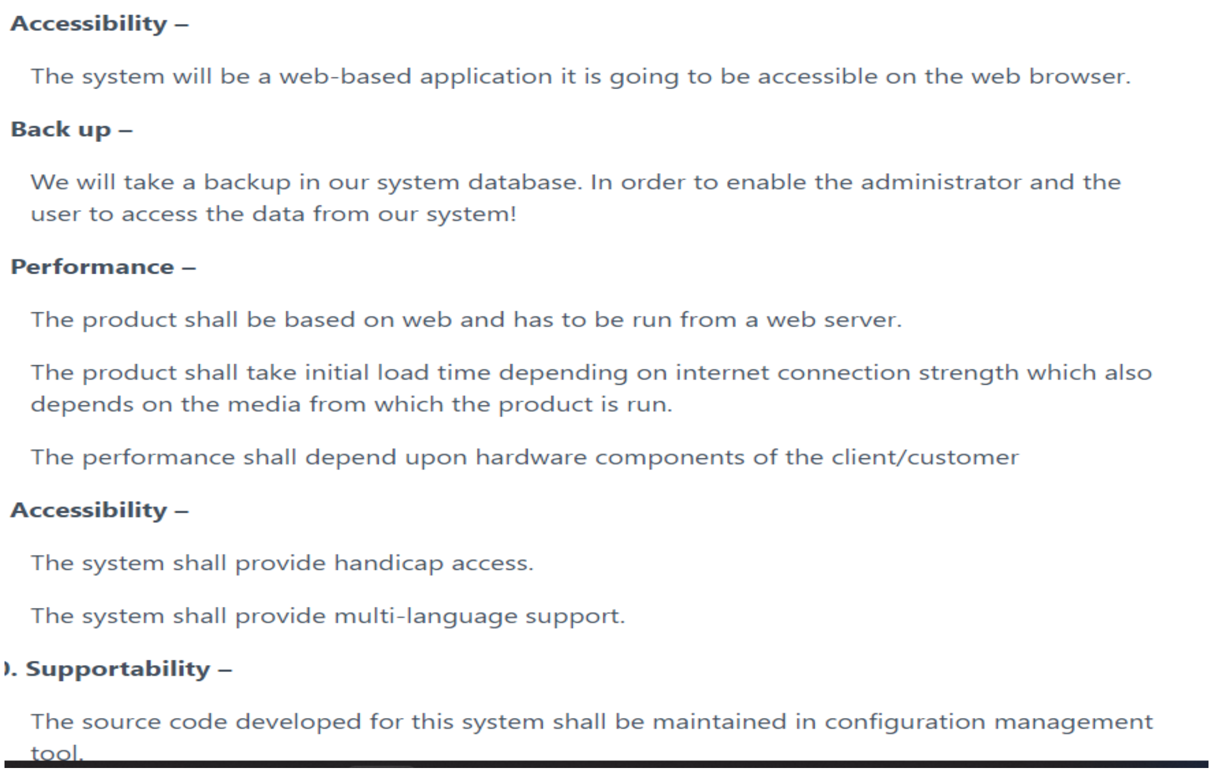
The system provides storage of all databases on redundant computers with automatic switchover.

Availability –

The system should be available at all times, meaning the user can access it using a web browser, only restricted by the downtime of the server on which the system runs. In case of an of a hardware failure or database corruption, a replacement page will be shown

Maintainability –

A commercial database is used for maintaining the database and the application server takes care of the site



Accessibility –

The system will be a web-based application it is going to be accessible on the web browser.

Back up –

We will take a backup in our system database. In order to enable the administrator and the user to access the data from our system!

Performance –

The product shall be based on web and has to be run from a web server.

The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run.

The performance shall depend upon hardware components of the client/customer

Accessibility –

The system shall provide handicap access.

The system shall provide multi-language support.

). Supportability –

The source code developed for this system shall be maintained in configuration management tool.

Functional requirement:

1. Registration –

If a customer wants to book the ticket then he/she must be registered, an unregistered user can't book the ticket.

2. Login –

Customer logins to the system by entering valid user id and password for booking the ticket.

3. Search Movie –

The system shall have a search function. Customer or visitor can search movies based on movie name, date, time and venue

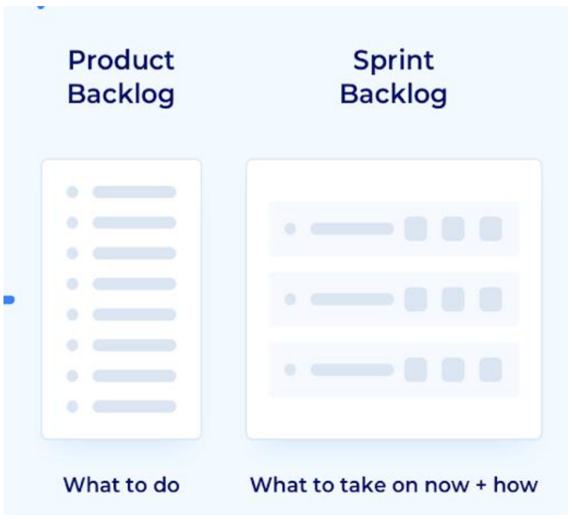
4. Seat Viewing –

The customer shall be shown a 2D image of the seats from which the desired seats are selected.

5.Ticket canceling –

The customer shall be given an option to cancel ticket one hour before the movie with some fine.

Backlogs :



We have to write product backlogs and sprint backlogs on our own

Product backlogs lo meru oka sari u r scrum master anukondii anukunii mi team members ki em works estahroo releated to ticket booking ki avi oka 4 rayandii enough

Sprint backlog lo emoo inka koni ideas rayandii same oka 4 emo product backlog lo and inka 4 emo sprint lo ala diagram geseee

Uses of scrum :

Quicker release of useable product to users and customers.

Higher quality.

Higher productivity.

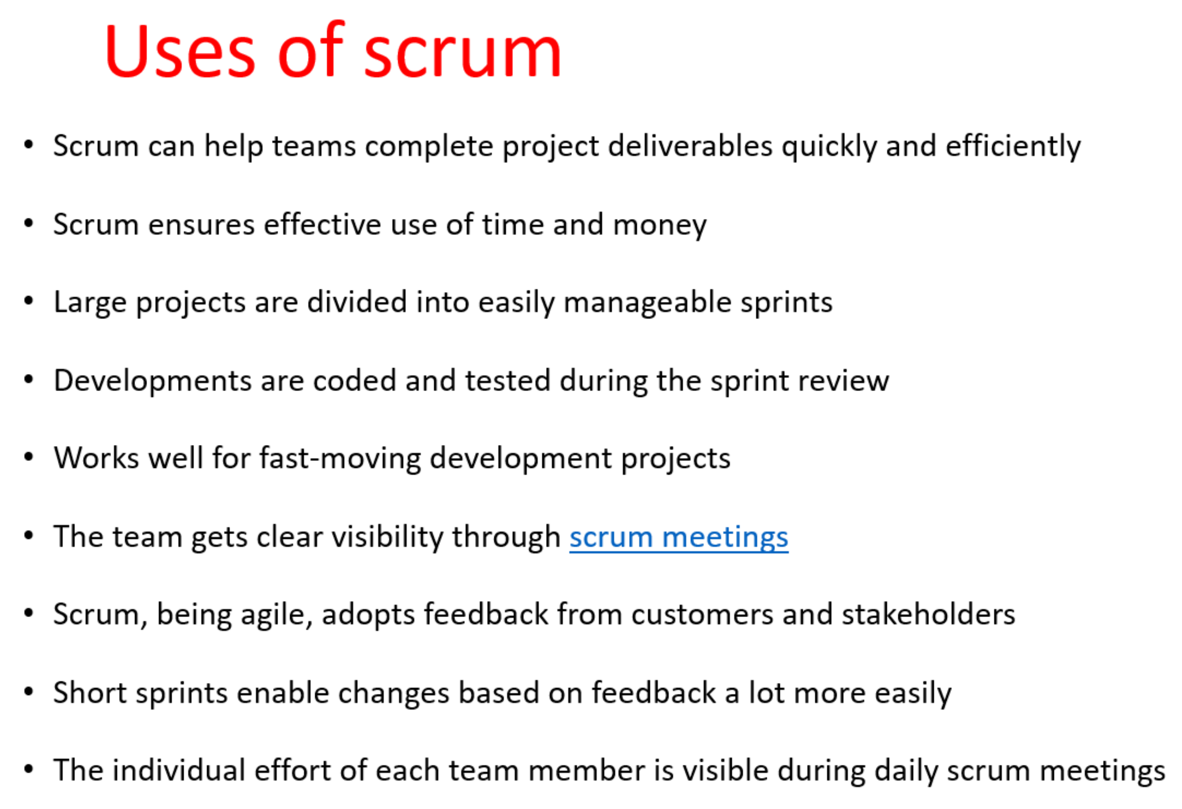
Lower costs.

Greater ability to incorporate changes as they occur.

Better employee morale.

Better user satisfactio.

Being able to complete complex projects that previously could not be done.



Uses of scrum

- Scrum can help teams complete project deliverables quickly and efficiently
- Scrum ensures effective use of time and money
- Large projects are divided into easily manageable sprints
- Developments are coded and tested during the sprint review
- Works well for fast-moving development projects
- The team gets clear visibility through scrum meetings
- Scrum, being agile, adopts feedback from customers and stakeholders
- Short sprints enable changes based on feedback a lot more easily
- The individual effort of each team member is visible during daily scrum meetings

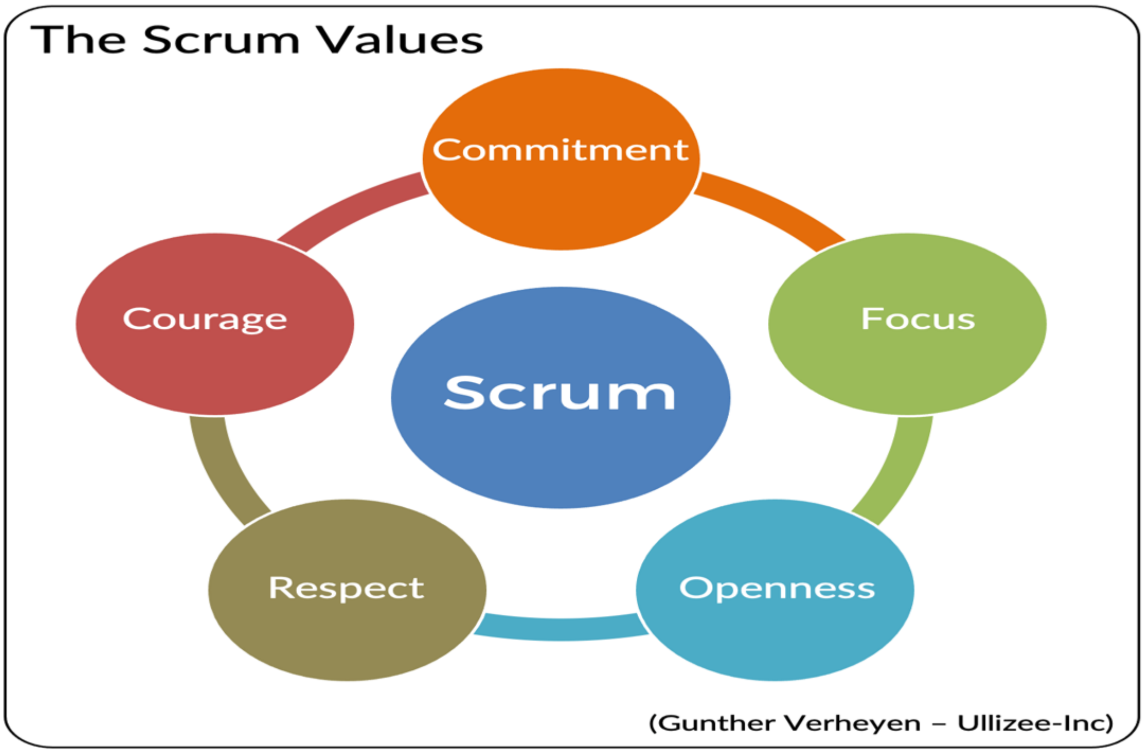
Scrum Vs Kanban

	Scrum	Kanban
Origin	Software development	Lean manufacturing
Ideology	Learn through experiences, self-organize and prioritize, and reflect on wins and losses to continuously improve.	Use visuals to improve work-in-progress
Cadence	Regular, fixed-length sprints (i.e. two weeks)	Continuous flow
Practices	Sprint planning, sprint, daily scrum, sprint review, sprint retrospective	Visualize the flow of work, limit work-in-progress, manage flow, incorporate feedback loops
Roles	Product owner, scrum master, development team	No required roles

Scrum by Example is written as an episodic story, with a small cast of characters and a simple fictional product. In each episode, there will be a breakdown of a specific problem or issue, as well as information and advice for how to handle it, whether you are a ScrumMaster or in any of the Scrum roles

Kanban example :

Giving a Project Manager Visibility into Status


The Scrum Values
Commitment
Courage
Focus
Scrum
Respect
Openness
(Gunther Verheyen – Ullizee-Inc)