Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT I

Chapter 5 – Agile Software Development

Concepts:

Agile Methods

Plan-Driven and Agile development

Extreme Programming

Agile Project Management (Scrum Process)

Scaling Agile Methods

Introduction

Businesses now operate in a global, rapidly changing environment. They have to respond to new opportunities and markets; changing economic conditions and the emergence of competing products and services. Software is a part of almost all business operations so new software is developed quickly to take advantage of new opportunities and to respond to competitive pressure. Rapid development and delivery are therefore now often the most critical requirement for software systems.

RAD Fundamental Properties

Rapid software development processes are designed to produce useful software quickly. The software is not developed as a single unit but as a series of increments with each increment including new system functionality.

Incremental development, Scrum, Agile Development and Extreme Programming these are the few approaches used to develop the Rapid Development systems. There are some fundamental characteristics:

1) The processes of specification, design and implementation are interleaved (mixture). There is no detailed system specification and design documentation is minimized or generated automatically by the programming environment used to implement the system. The user requirement document only defines the most important characteristics of the system.

2) The system is developed in a series of versions. End -users and other system stakeholders are involved in specifying and evaluating each version. They may propose changes to the software and new requirements that should be implemented in a later version of the system.

3) System user interfaces are often developed using an interactive development system that allows the interface design to be quickly created by drawing and placing icons on the interface. The system may then generate a web-based interface for a browser or an interface for a specific platform such as Microsoft Windows.

Agile Methods

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams' cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders.

The agile method is an iterative and incremental tactic for software design that utilizes constant planning, understanding, upgrading, team partnership, development, and delivery. The agile process is fragmented into separate models that teams work on, thereby encouraging flexibility to changes.
the agile methodology originates with customers defining the end uses of the final product and the kind of problems the final product attempts to address. This exercise helps in resolving and clarifying the customer's anticipations and requirements for the project development team.

As the project commences, the designated teams start to plan and work on a complete process through planning, implementing, and appraising. As the development process is iterative, errors are resolved in the intermediate stage of the project. This process enables the final deliverable product to match the customer's wants better, who can then propose new and changed

requirements to be included in later iterations of the system. The best-known agile method is extreme programming.

Agile methods have been very successful for some types of software development:

1) Product development where a software company is developing a small or medium sized product for sale

2) Custom system development within an organization where there is a clear commitment from the customer to become involved in the development process and where there are not lot of external rules and regulations that affect the software.

The philosophy behind agile methods is reflected in the agile manifesto that was agreed on by many of the leading developers of these methods. The manifesto focuses on the following:

1) Individuals and interactions over processes and tools

2) Working software over comprehensive documentation

3) Customer collaboration over contract negotiation

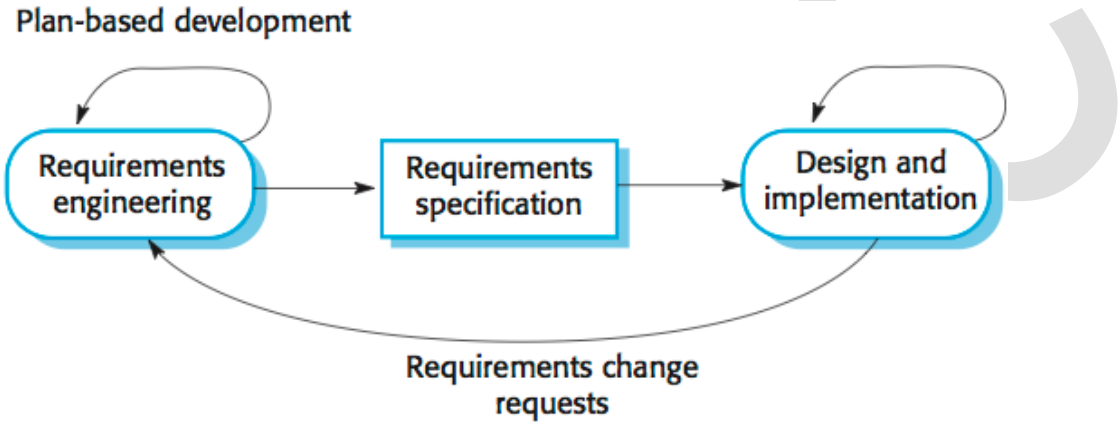
4) Responding to change over following a plan

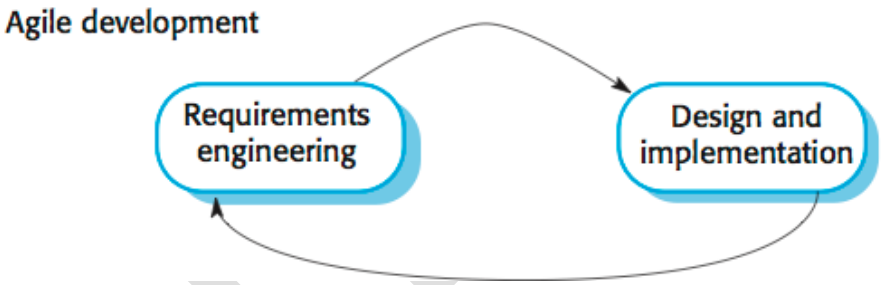
Agile methods are all based on around the notion of incremental development and delivery they propose different processes to achieve this. They share a set of principles based on agile manifesto which are shown in following table:

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is to provide and prioritize new system requirements and to evaluate the iteration of the system
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment
People not Process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without perspective process

Embrace change	Except the system requirements to change and so design the change to accommodate these changes
Maintain Simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible actively work to eliminate complexity from the system

Plan – driven and Agile development





Plan-driven development

A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance. Not necessarily waterfall model: plan-driven, incremental development is possible. Iteration occurs within activities.

Agile development

Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on many technical, human, and organizational issues.

Technical, human, organizational issues

Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:

♣ Is it important to have a very detailed specification and design before moving to implementation?

➔ If so, you probably need to use a plan-driven approach.

♣ Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic?

➔ If so, consider using agile methods.

♣ How large is the system that is being developed?

➔ Agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.

♣ What type of system is being developed?

➔ Plan-driven approaches may be required for systems that require a lot of analysis before implementation (e.g., real-time system with complex timing requirements).

♣ What is the expected system lifetime?

➔ Long-lifetime systems may require more design documentation to communicate the original intentions of the system developers to the support team.

♣ What technologies are available to support system development?

➔ Agile methods rely on good tools to keep track of an evolving design

♣ How is the development team organized?

➔ If the development team is distributed or if part of the development is being outsourced, then you may need to develop design documents to communicate across the development teams.

♣ Are there cultural or organizational issues that may affect the system development?

➔ Traditional engineering organizations have a culture of plan-based development, as this is the norm in engineering.

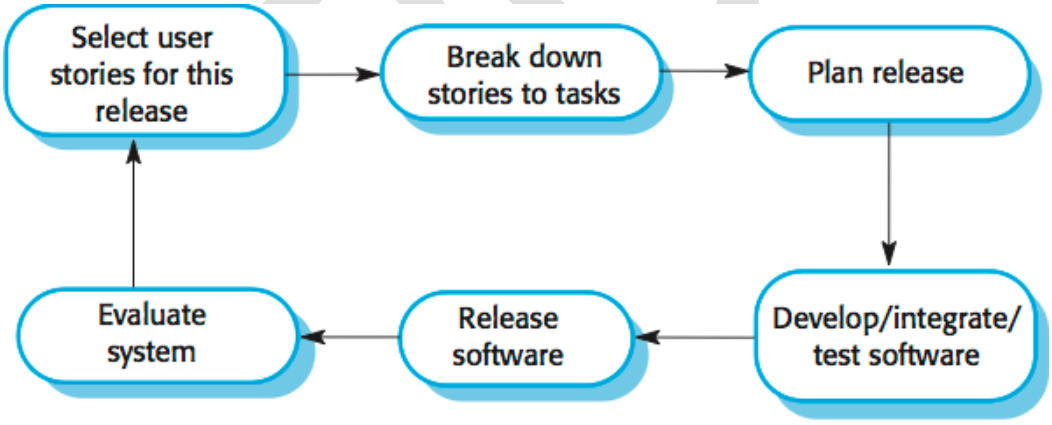
♣ How good are the designers and programmers in the development team?

➔ It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code

♣ Is the system subject to external regulation?

➔ If a system has to be approved by an external regulator (e.g., the FAA approve software that is critical to the operation of an aircraft) then you will probably be required to produce detailed documentation as part of the system safety case.

Extreme Programming



The name of the approach is coined due to iterative development at the extreme levels. In extreme programming requirements are expressed as scenarios (called user stories) which are implemented directly as a series of tasks. Programmers work in pair and develop tests for each task before writing the code. All tests must be successfully executed when new code is integrated into the system. There is short time gap between releases of the system. Extreme programming involves a number of practices as follows:

Principle	Description
Incremental planning	Requirements are recorded and to be included based on their priority.
Small releases	The functionality required is developed first. Releases of the system incrementally add functionality to the first release
Simple design	Enough design is carried out to meet current requirements
Test first development	An automated unit test is used prior to the functional code
Refactoring	Changing the structure of the code without altering its behavior for code improvements
Pair programming	Developers work in pairs, checking each other's code and providing the support for performing a good job
Collective ownership	The pair of developers work on all areas of the system, so that all developers take responsibility for all the code. Anyone can change anything.
Continuous integration	As soon as the work on the task is complete, it is integrated into the whole system
Sustainable pace	Large amounts of overtime are not considered
On – site customer	A representative of the end-user of the customer should be available full time for the use of the team

Extreme Programming approach

First, start off by describing the desired results of the project by having customers define a set of stories. As these stories are being created, the team estimates the size of each story. This size estimate, along with relative benefit as estimated by the customer can provide an indication of relative value which the customer can use to determine priority of the stories.

If the team identifies some stories that they are unable to estimate because they don't understand all of the technical considerations involved, they can introduce a spike to do some focused research on that particular story or a common aspect of multiple stories.

Spikes are short, time-boxed time frames set aside for the purposes of doing research on a particular aspect of the project. Spikes can occur before regular iterations start or alongside ongoing iterations.

Next, the entire team gets together to create a release plan that everyone feels is reasonable. This release plan is a first pass at what stories will be delivered in a particular quarter, or release. The stories delivered should be based on what value they provide and considerations about how various stories support each other.

Then the team launches into a series of weekly cycles. At the beginning of each weekly cycle, the team (including the customer) gets together to decide which stories will be realized during that week. The team then breaks those stories into tasks to be completed within that week. At the end of the week, the team and customer review progress to date and the customer can decide whether the project should continue, or if sufficient value has been delivered.

Test-first development

Testing is central to XP and XP has developed an approach where the program is tested after every change has been made.

Test-driven development: writing tests before code clarifies the requirements to be implemented. Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly (usually relies on a testing framework such as Junit). All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.

Customer involvement: The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system. The customer who is part of the team writes tests as development proceeds. All new code is therefore validated to ensure that it is what the customer needs. However, people adopting the customer role have limited time available and so cannot work full-time with the development team. They may feel that providing the requirements was enough of a contribution and so may be reluctant to get involved in the testing process.

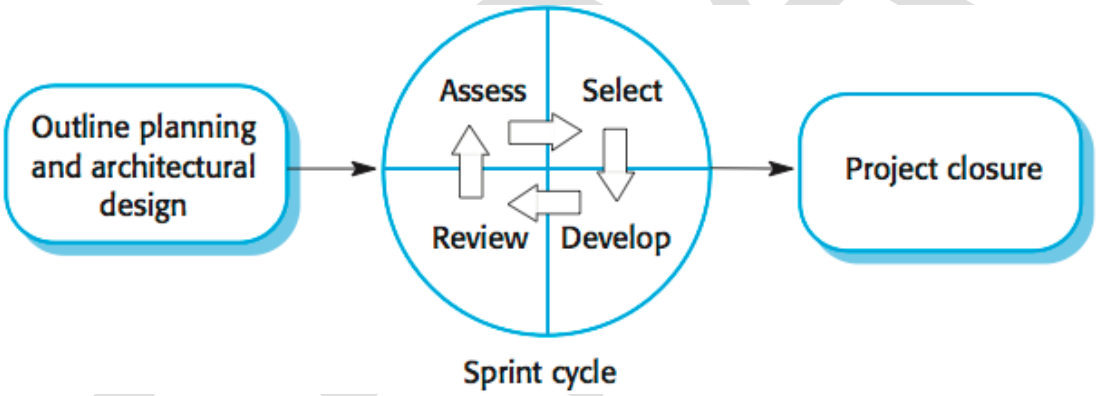
Pair programming

Pair programming involves programmers working in pairs, developing code together. This helps develop common ownership of code and spreads knowledge across the team. It serves as an

informal review process as each line of code is looked at by more than one person. It encourages refactoring as the whole team can benefit from improving the system code. In pair programming, programmers sit together at the same computer to develop the software. Pairs are created dynamically so that all team members work with each other during the development process. The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave. Pair programming is not necessarily inefficient and there is some evidence that suggests that a pair working together is more efficient than two programmers working separately.

Agile Project Management (SCRUM PROCESS)

The principal responsibility of software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project. They supervise the work of software engineers and monitor how well the software development is progressing. The standard approach to project management is plan driven.



There are three phases in Scrum:

1) The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.

2) This is followed by a series of sprint cycles, where each cycle develops an increment of the system.

3) The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

The innovative feature of Scrum is its central phase namely the sprint cycles. **A Scrum sprint is a planning unit in which work to be done is assessed, features are selected for development and the software is implemented**. At the end of a sprint the completed functionality is delivered to the stakeholders. Key characteristics of this process are as follows:

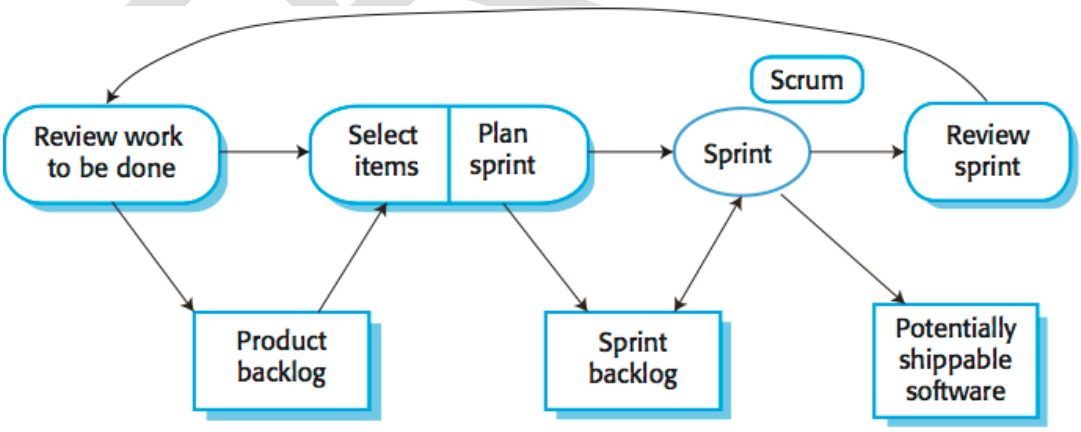
1) Sprints are fixed length, normally 2-4 weeks. They correspond to the development of a release of the system in XP.

2) The starting point for planning is the **product backlog**, which is the list of work to be done on the project.

3) The selection phase involves all of the project team who work with the customer (**product owner**) to select the features and functionality to be developed during the sprint.

4) Once these are agreed, the team organize themselves to develop the software. During this stage the team is relatively isolated from the product owner and the organization, with all communications channeled through the **ScrumMaster**. The **role of the ScrumMaster** is to protect the development team from external distractions.

5) At the end of the sprint the work done is reviewed and presented to stakeholders (including the product owner). **Velocity** is calculated during the **sprint review**; it provides an estimate of how much product backlog the team can cover in a single sprint. Understanding the team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring and improving performance. The next sprint cycle then begins.



The ScrumMaster is a facilitator who arranges short daily meetings **(daily scrums**), tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with the product owner and management outside of the team. The whole team

attends daily scrums where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.

Scaling agile methods

Agile methods have proved to be successful for small and medium sized projects that can be developed by a small co-located team. It is sometimes argued that the success of these methods comes because of improved communications which is possible when everyone is working together. Scaling up agile methods involves changing these to cope with larger, longer projects where there are multiple development teams, perhaps working in different locations.

Two perspectives on scaling of agile methods:

'Scaling up'

Using agile methods for developing large software systems that cannot be developed by a small team. For large systems development, it is not possible to focus only on the code of the system; you need to do more up-front design and system documentation. Cross-team communication mechanisms have to be designed and used, which should involve regular phone and video conferences between team members and frequent, short electronic meetings where teams update each other on progress. Continuous integration, where the whole system is built every time any developer checks in a change, is practically impossible; however, it is essential to maintain frequent system builds and regular releases of the system.

'Scaling out'

How agile methods can be introduced across a large organization with many years of software development experience. Project managers who do not have experience of agile methods may be reluctant to accept the risk of a new approach. Large organizations often have quality procedures and standards that all projects are expected to follow and, because of their bureaucratic nature, these are likely to be incompatible with agile methods. Agile methods seem to work best when team members have a relatively high skill level. However, within large organizations, there are likely to be a wide range of skills and abilities. There may be cultural resistance to agile methods, especially in those organizations that have a long history of using conventional systems engineering processes.