Unit 1 – Chapter 4

Alternate Software Development Models

4.0 Objectives

The objective of this part of the unit is to introduce you to the alternative software development models. At the end of this chapter you should be able to:

- Understand Agile methodology
- Understand Rational Unified Process.
- Compare alternate software development models

4.1 Non-Traditional Software Development Processes

In recent years, a number of ideas have emerged on novel software development processes. The common features of all these processes is iterative and incremental development, with a view to complying with changing user requirements.

4.2 Rapid Application Development (RAD) Model

This model was developed at IBM in response to the deficiencies of the waterfall model. The features of this model are the following:

- It is based on prototyping and iterative development with no specific planning involved.
- Using the RAD model, software product is developed in a short period of time(60-90 days).
- The critical feature of this model is the use of powerful development tools and techniques.

- In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype.
- This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.
- Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

The phases of the RAD model are shown in fig 1.4.1.

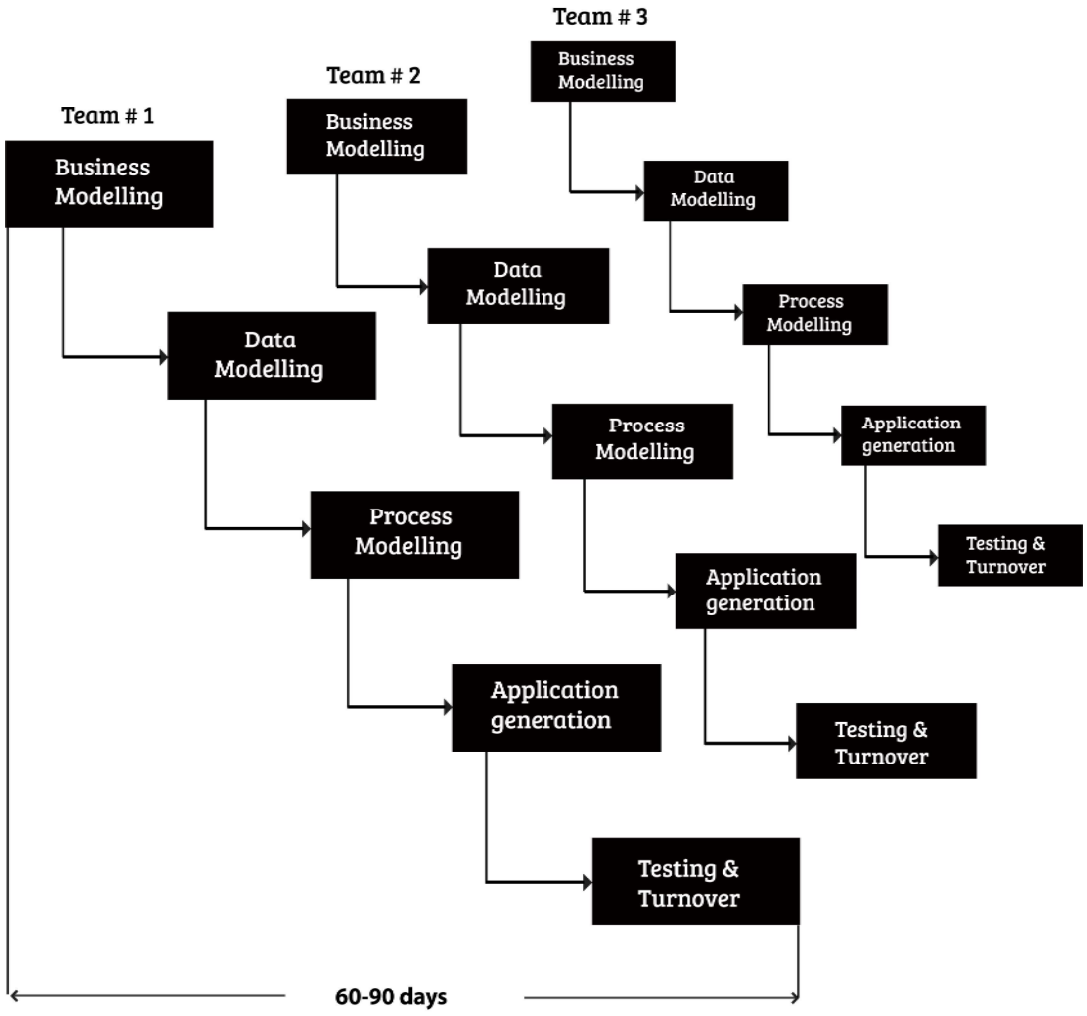

Fig 1.4.1: Phases of the RAD model

The phases of the RAD model are the following:

1. **Business Modeling**:The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.
2. **Data Modeling**:The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.
3. **Process Modeling**:The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
4. **Application Generation**:Automated tools are used to facilitate construction of the software.
5. **Testing & Turnover**: Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new features must be tested, and all interfaces must be fully exercised.

Advantages of the RAD model:

- Reduced development time.
- Increases re-usability of components.
- Quick initial reviews occur.
- Encourages customer feedback.
- Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/designers.
- High dependency on modeling skills.
- For smaller projects, we cannot use the RAD model.
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

4.3 Rational Unified Process

Rational Unified Process (RUP) is a process-independent life cycle approach. The RUP is a phased model that identifies four discrete phases in the software process. However, unlike the waterfall model where phases are equated with process activities, the phases in the RUP are more closely related to business rather than technical concerns.

The RUP recognizes that conventional process models present a single view of the process. In contrast, the RUP is normally described from three perspectives:

1. A *dynamic* perspective, which shows the phases of the model over time.
2. A *static* perspective, which shows the process activities that are enacted.
3. A *practice* perspective, which suggests good practices to be used during the process.

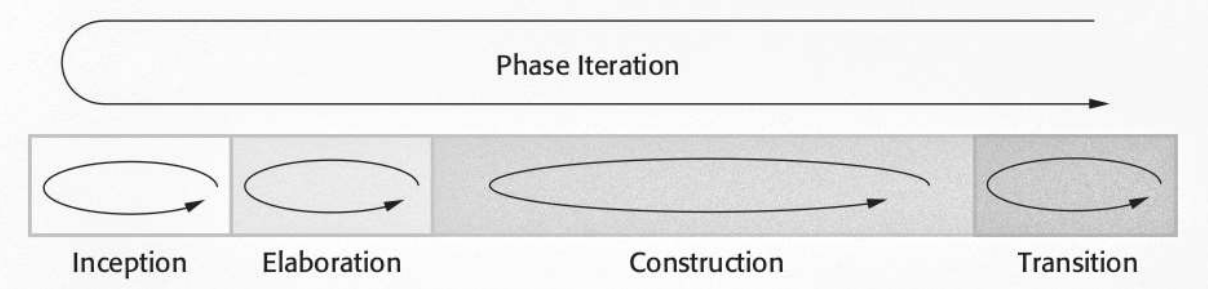

Fig 1.4.2: Phases of RUP

Iterations within the RUP is supported in two ways. Each phase may be enacted in an iterative way with the results developed incrementally. In addition, the whole set of phases may also be enacted incrementally, as shown by the looping arrow from Transition to Inception in Figure 1.4.2. The time-span to deliver an iteration is usually 45 days.

The following are the phases of RUP:

Inception: The goal of the inception phase is to establish a business case for the system. All external entities (people and systems) that will interact with the system are identified and these interactions are defined.

Elaboration: The goals of the elaboration phase are to develop an understanding of the problem domain, establish an architectural framework for the system, develop the project plan, and identify key project risks.

Construction: The construction phase involves system design, programming, and testing. Parts of the system are developed in parallel and integrated during this phase.
Transition: The final phase of the RUP is concerned with moving the system from the development community to the user community and making it work in a real environment.

The static view of the RUP focuses on the activities that take place during the development process. These are called workflows in the RUP description. There are six core process workflows identified in the process and three core supporting workflows. The core engineering and support workflows are described in Table 1.4.1.

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models, and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users, and installed in their workplace.
Configuration and change management	This supporting workflow manages changes to the system.
Project management	This supporting workflow manages the system development
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Table 1.4.1: Static workflows in RUP

The practice perspective on the RUP describes good software engineering practices that are recommended for use in systems development. Six fundamental best practices are recommended:

1. *Develop software iteratively:* Plan increments of the system based on customer priorities and develop the highest-priority system features early in the development process.
2. *Manage requirements Explicitly:* document the customer's requirements and keep track of changes to these requirements. Analyze the impact of changes on the system before accepting them.
3. *Use component-based architectures:* Structure the system architecture into components.
4. *Visually model software:* Use graphical UML models to present static and dynamic views of the software.
5. *Verify software quality:* Ensure that the software meets the organizational quality standards.
6. *Control changes to software:* Manage changes to the software using a change management system and configuration management procedures and tools.

4.4 Agile Development Process

Agile processes are preferred where user requirements change rapidly. The philosophy behind agile methods is reflected in the agile manifesto that was agreed on by many of the leading developers of these methods. This manifesto states:

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.

Based on this manifesto, the agile practices have the characteristics listed in Table 1.4.2.

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

Table 1.4.2: The principles of agile methods

Agile development process follows a different development sequence compared to the conventional software development as shown in Fig 1.4.3.

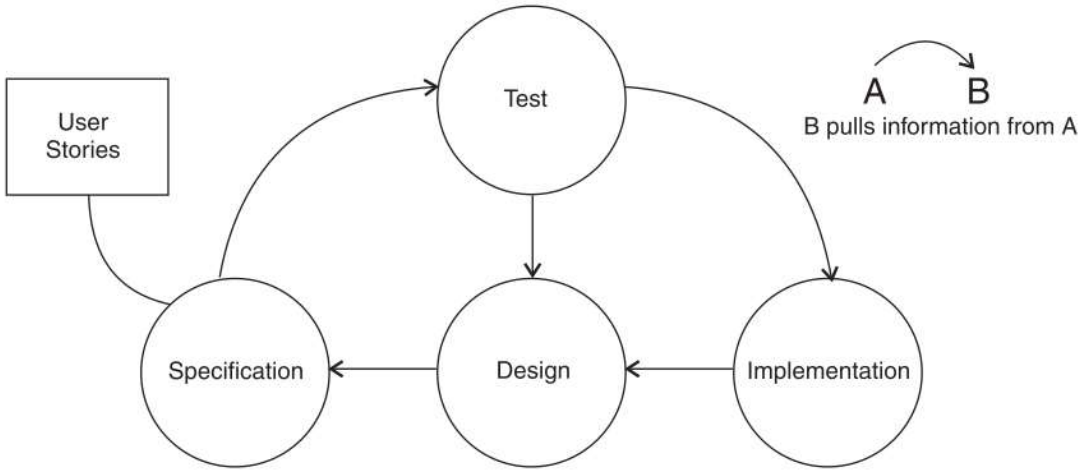

Fig 1.4.3 Agile Development Process

- At the beginning of each development scenario, system functionalities are recorded in the form of *user stories*.
- The customer and development teams derive the test situations from the specifications.
- Developers design programming interface to match the tests' needs and they write the code to match the tests and the interface.

- The developers refine the design to match the code.

In practice, the principles underlying agile methods are sometimes difficult to realize because:

- It can be difficult to keep the **interest of customers** who are involved in the process.
- Team members may be unsuited to the **intense involvement** that characterizes agile methods.
- **Prioritizing changes** can be difficult where there are multiple stakeholders.
- **Maintaining simplicity** requires extra work.
- **Contracts may be a problem** as with other approaches to iterative development.

Extreme Programming (XP) and SCRUM are two of the most popular agile processes.

4.4.1 Extreme Programming (XP)

In extreme programming, requirements are expressed as scenarios (called user stories), which are implemented directly as a series of tasks. Programmers work in pairs 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 a short time gap between releases of the system. Each iteration usually takes 2-4 weeks time. Figure 1.4.4 shows the Extreme Programming process in detail.

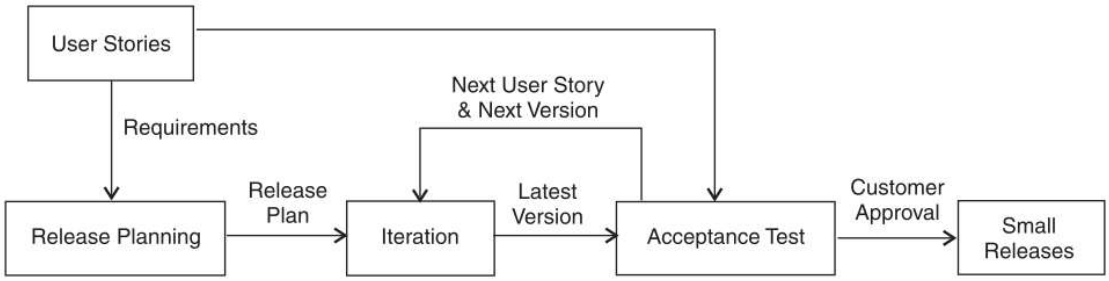

Fig1.4.4 Extreme programming—simplified process

- *User stories* are descriptions of the functionalities the system is expected to do. The customer writes a user story about each functionality in no more than three sentences in his/her own words. They just provide enough details to be able to make low-risk time estimate to develop and implement. At the time of implementation, the developers collect additional requirements by talking to the customer face to face. User stories are used to make time estimates for

implementing a solution. Each story ideally takes between 1 to 3 weeks to implement.

- User stories are used for *release planning* and creating acceptance tests. Release plan specifies the user stories which are to be developed and implemented in a particular release. Between 60 and 100 stories constitute a release plan. A release plan also specifies the date for the release. Customer, developers, and managers attend a release planning meeting. Customer prioritizes the user stories, and the high-priority stories are taken up for development first.
- Each release requires several *iterations*. The first few iterations take up the high-priority user stories. These user stories are then translated into programming tasks that are assigned to a group of programmers. The user stories to be taken up and the time to develop them in one iteration are decided in an iteration planning meeting. Iteration planning meeting takes place before the next iteration is due to start. A maximum of dozen iterations are usually done for a release plan.
- User stories are also used to plan *acceptance tests*. Extreme programming expects that at least one automated acceptance test is created to verify that the user stories are correctly implemented.
- *Coding* required for a user story is usually done by two programmers. Unit tests are carried out to ensure that each unit is 100% bug free. Programmers focus on the current iteration and completely disregard any consideration outside of this iteration. The code is group owned, meaning that any code not working is the responsibility of the whole group and not merely of the programmer writing the code.

The characteristics of Extreme programming are the following:

- *Test-first programming*–Test precedes either design or coding.
- *Incremental*–small software releases with rapid iterations.
- *Iterative development*, each iteration addressing specific user requirements.
- *Just-in-time development* with micro-planning taking place for each iteration
- *Cooperative*–client and developers working constantly together with close communication.
- *Collective code ownership*, with writing defect-free code as the responsibility of the whole group of programmers.

· *Straightforward*–the model itself is easy to learn and to modify and is well-documented.

· *Adaptive*–last-minute changes can be made.

· *Intensive user involvement* in specifying requirements, prioritizing them, making release plans, and creating acceptance tests.

4.4.2 SCRUM

SCRUM comprises a set of project management principles based on small cross-functional self-managed teams (Scrum teams). There are three phases in Scrum.

1. The first 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. Finally, 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.

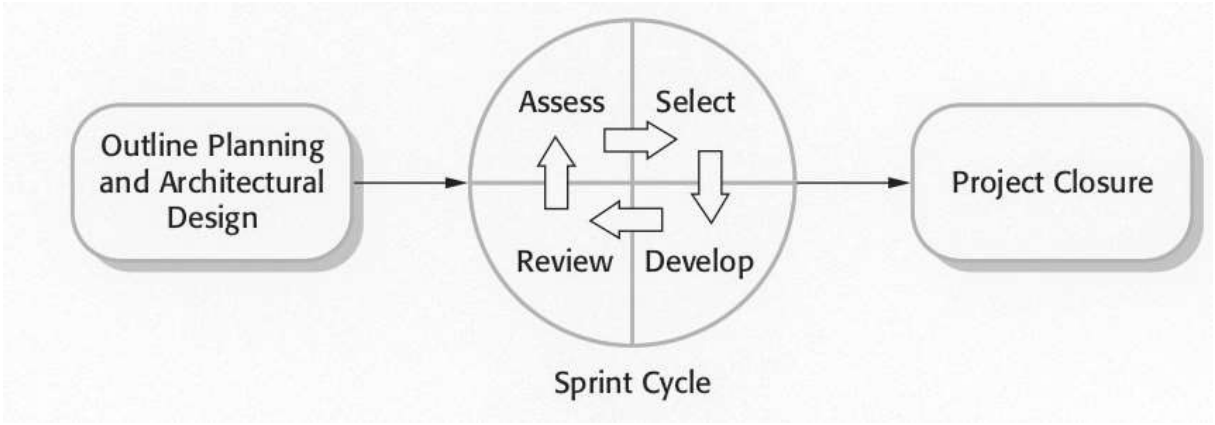

Fig :1.4.5: The SCRUM process

The key characteristics of the sprint cycle are:

- Sprints are fixed length, normally 2—4 weeks. They correspond to the development of a release of the system in XP.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint.

- Once these are agreed, the team organize themselves to develop the software. During this stage the team is isolated from the customer and the organization, with all communications channeled through the so-called 'Scrum master'.
- The role of the Scrum master is to protect the development team from external distractions.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

The idea behind Scrum is that the whole team should be empowered to make decisions so the term 'project manager', has been deliberately avoided. The 'Scrum master' is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog, and communicates with customers and management outside of the team. The whole team attends the daily meetings, which are sometimes 'stand-up' meetings to keep them short and focused. During the meeting, all team members share information, describe their progress since the last meeting, problems that have arisen, and what is planned for the following day. This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to
cope with them. Everyone participates in this short-term planning.

The following are the benefits of using the SCRUM process:

1. The product is broken down into a set of manageable and understandable chunks.
2. Unstable requirements do not hold up progress.
3. The whole team has visibility of everything and consequently team communication is improved.
4. Customers see on-time delivery of increments and gain feedback on how the product works.
5. Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

4.5 MCQs

1.Which on of the following is not an agile method?

a) XP

b) Spiral

c) SCRUM

d) Prototyping

2. In agile development it is more important to build software that meets the customers' needs today than worry about features that might be needed in the future.

a) True

b) False

3. User requirements are expressed as _____ in Extreme Programming.

a) implementation tasks

b) functionalities

c) User Stories

d) none of the mentioned

4. In XP an automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.

a) True

b) False

5. Which answer below is not one of the RUP life cycle phases?

a) Inception Phase

b) Iteration Phase

c) Elaboration Phase

d) Transition Phase

6. Which phase of the RUP is used to establish a business case for the system ?

a) Transition

b) Elaboration

c) Construction

d) Inception

7. RUP stands for _____

a)Rational Unified Process

b)Rational Unified Program

a)Relational Unified Process

a)Relational Unified Program

8. Which of the following is delivered at the end of the Sprint?

a. A document containing test cases for the current sprint
b. An architectural design of the solution
c. Wireframes designs for User Interface
d. An increment of Done software

4.5 Review Questions

1. Discuss the advantages and disadvantages of Xtreme Programming.
2. Discuss the Scrum approach to project management in Agile.
3. Explain the characteristics of the agile development process.
4. Describe the phases of the RAD model.
5. What are the advantages and disadvantages of the RAD model.
6. List the best practices to develop software with RUP.
7. Describe the key characteristics of the SCRUM sprint cycle.
8. At the end of their study program, students in a software engineering course are typically expected to complete a major project. Explain how the agile methodology may be very useful for the students to use in this case.
9. Compare and contrast the Scrum approach to project management with conventional plan-based approaches.
10. Explain why agile methods may not work well in organizations that have teams with a wide range of skills and abilities and well-established processes.

4.6 MCQ solution key

1. B 2. A 3.C 4.A 5.B 6.A 7.A 8.D
