Software Engineering

Er. Prahlad Panthi

Agile Methods: Introduction

- Incremental development methods in which the increments are small and, typically, new releases of the system are created and made available to customers every two or three weeks.
- Involve customers in the development process to get rapid feedback on changing requirements and minimize documentation by using informal communications rather than formal meetings with written documents.
- Best suited to application development where the system requirement usually change rapidly during the development process.
- 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.

Principles of agile methods:

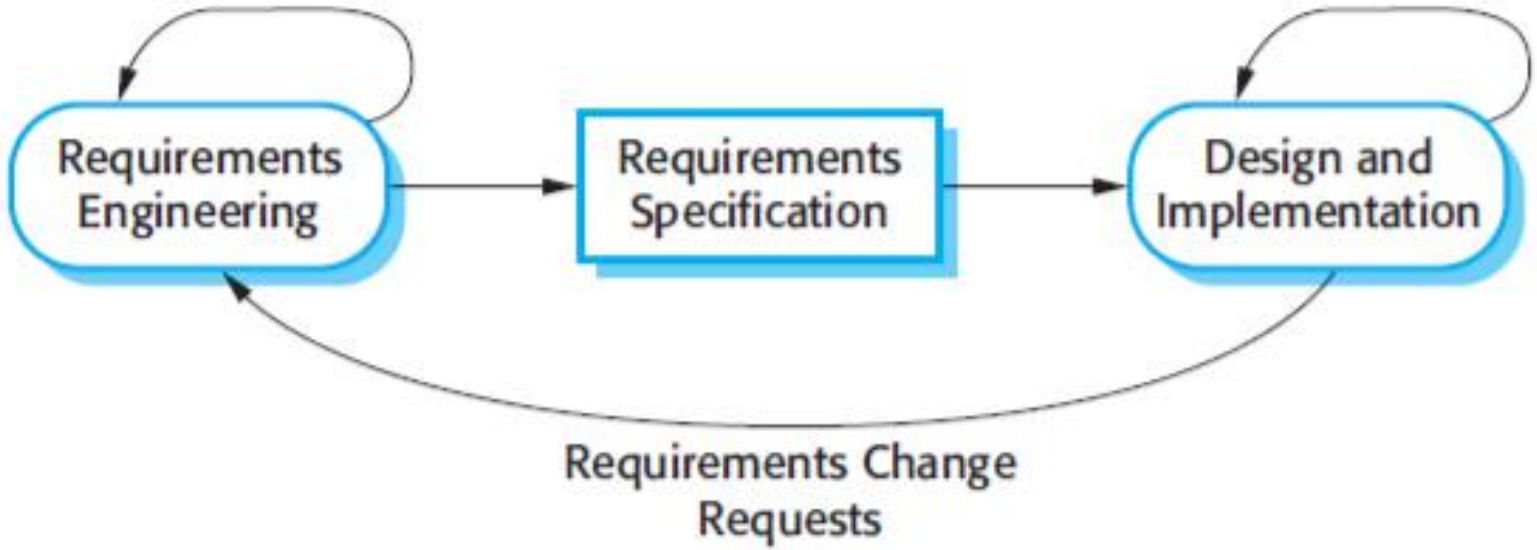
Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer 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.

- Formal documentation is supposed to describe the system and so make it easier for people changing the system to understand. In practice, however, formal documentation is often not kept up to date and so does not accurately reflect the program code. For this reason, agile methods enthusiasts argue that it is a waste of time to write this documentation and that the key to implementing maintainable software is to produce high-quality, readable code.
- Instead agile practices emphasize the importance of writing well-structured code and investing effort in code improvement. Therefore, the lack of documentation should not be a problem in maintaining systems developed using an agile approach.

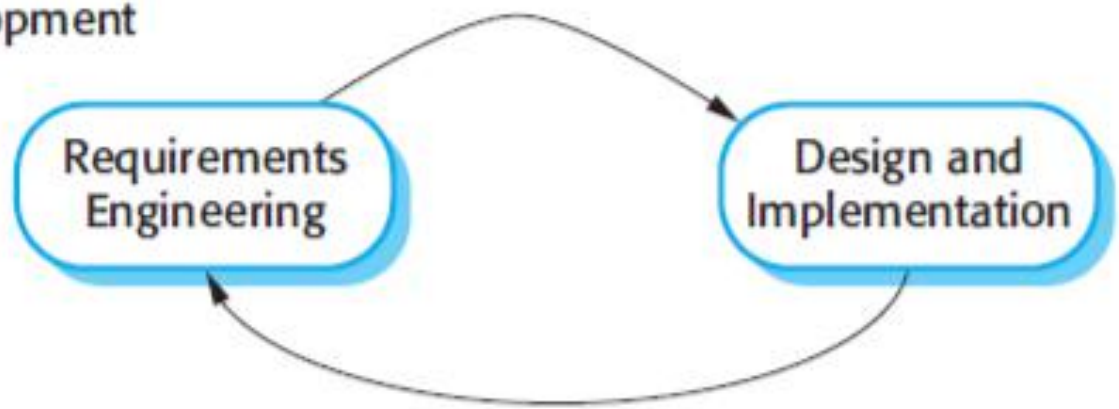
Plan driven and Agile Development

- Plan-driven approach to software engineering identifies separate stages in the software process with outputs associated with each stage. The outputs from one stage are used as a basis for planning the following process activity.
- In a plan-driven approach, iteration occurs within activities with formal documents used to communicate between stages of the process.
- In an agile approach, iteration occurs across activities. Therefore, the requirements and the design are development together, rather than separately.
- Most software projects include practices from both plan-driven and agile approaches.

Plan-Based Development



Agile Development



Extreme Programming (XP)

- Perhaps the best known and most widely used of the agile methods.
- 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.

▼ Extreme Programming practices

Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

Pair Programming

- Programmers work in pairs to develop the software. Pairs are created dynamically so that all team members work with each other during the development process.
- Advantages:
  - Supports the idea of collective ownership and responsibility for the system.
  - Acts as an informal review process because each line of code is looked at by at least two people.
  - Helps support refactoring, which is a process of software improvement.

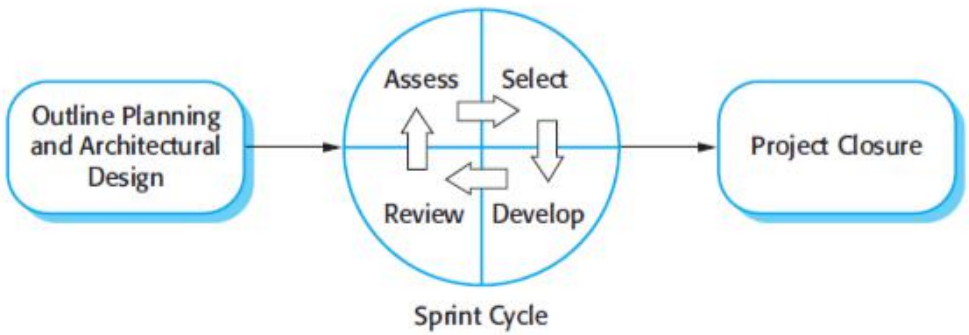
Agile Project Management

The standard approach to project management is plan-driven, where managers draw up a plan for the project showing what should be delivered, when it should be delivered, and who will work on the development of the project deliverables. However, it does not work well with agile methods where the requirements are developed incrementally; where software is delivered in short, rapid increments.

Scrum

- Focuses on managing iterative development rather than specific technical approaches to agile software engineering.
- It does not prescribe the use of programming practices like pair programming and test-first development. So, it is used with more technical agile approaches like XP, to provide a management framework for the project.

Three phases in Scrum:



1. Outline planning phase: establish the general objectives for the project and design the software architecture.
2. Each sprint cycle develops an increment of the system.
3. 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.

- A Scrum sprint is a planning unit in which the 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 stakeholders.

▼ Key Characteristics:

  - Sprints are fixed length, normally 2-4 weeks. They correspond to the development of a release of the system in XP.
  - Starting point for planning is the **product backlog,** which is the list of work to be done on the project. During the assessment phase of the sprint, this is reviewed, and priorities and risks are assigned with the close involvement of customer.
  - 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 organizes themselves to develop the software. Short daily meetings (**stand-up**) involving all team members are held to review progress and if necessary reprioritize work. During this stage, the team is isolated from the customer and the organization, with all communications channelled through the **Scrum Master**.
  - At the end of the sprint, the work done is reviewed and presented to stakeholders. The next spring cycle then begins.

▼ Scrum Master

  - The idea behind Scrum is that the whole team should be empowered to make decisions so the term '**project manager',** has been deliberately avoided, and '**scrum master**' is used.
  - facilitator who arranges daily meetings or daily stand-up, 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.
  - During daily stand-up, all team members share information, describe their progress since the last meeting, problems that have arisen, and what is planned for the following day. Thus, everyone on the team knows what is going on and if problem arise, can re-plan short-term work to cope with them.

To Be Continued…