Software Engineering

CS-303

Agile Software Development

Lecture # 7, 8
04, 06 Feb

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Topics covered

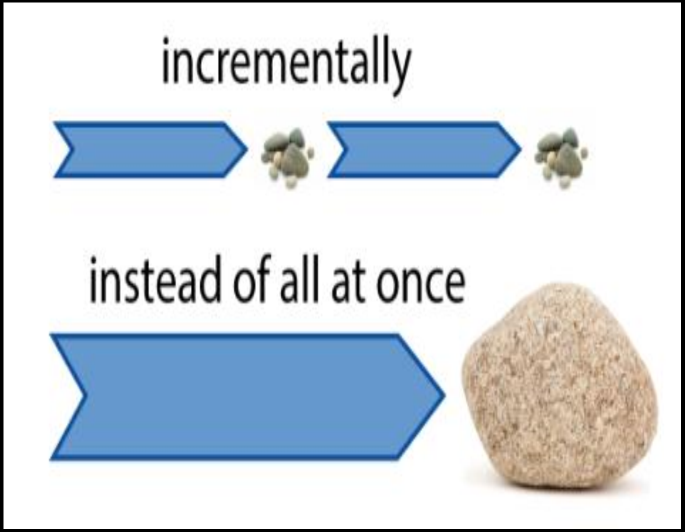
- Agile methods
- Agile development techniques
- Agile project management
- Scaling agile methods

Rapid Software Development

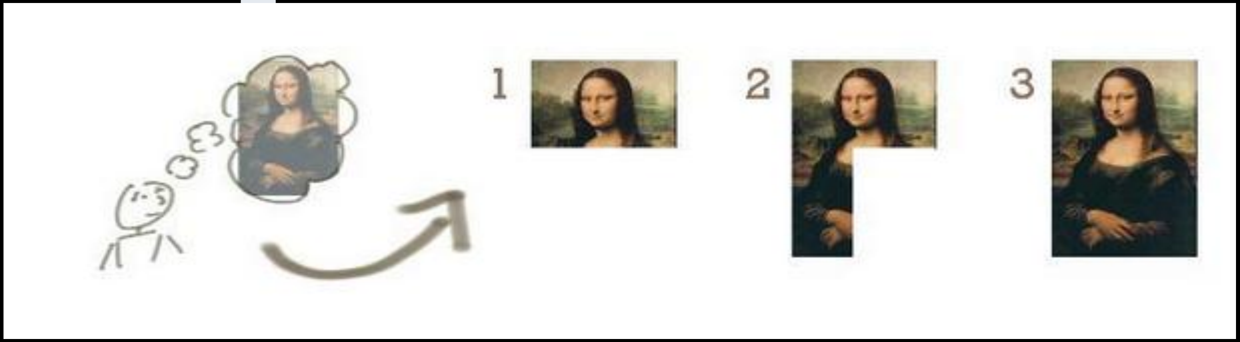
- Rapid development and delivery is now often the most important requirement for software systems
  - Businesses operate in a fast –changing requirement and it is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs.
- Plan-driven development is essential for some types of system but does not meet these business needs.
- Agile development methods emerged in the late 1990s whose aim was to radically reduce the delivery time for working software systems

About Agile

- ❑ Agile is a Software Development Methodology
- ❑ It means '<u>ability to move quickly and easily' and responding swiftly to change</u>
- ❑ It's a time boxed, iterative approach
- ❑ Develop and deliver software incrementally from the start of the project, instead of trying to deliver it all at once near the end.

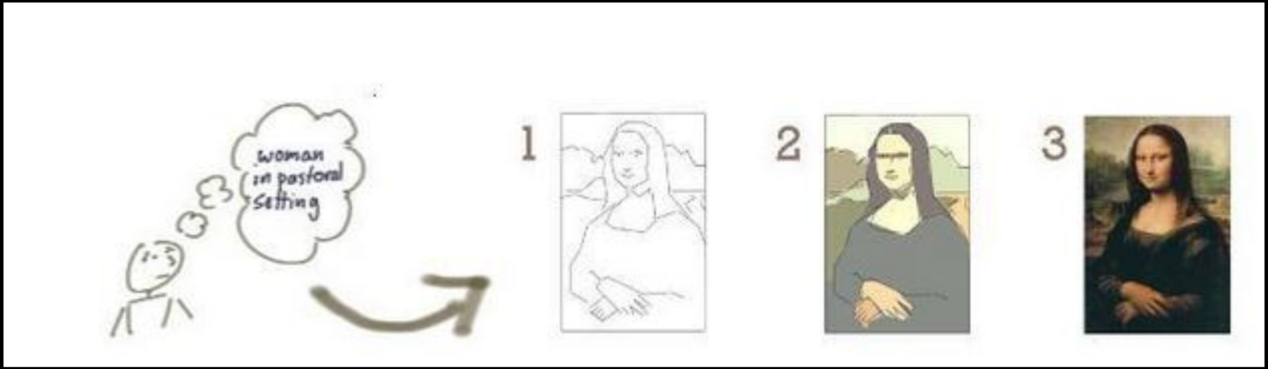


Example - Incremental



In the diagram above when we work **incrementally** we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it's complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers. Likewise in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

Example-- Iterative



In the diagram above when we work **iteratively** we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it's finished. As shown in the image above, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

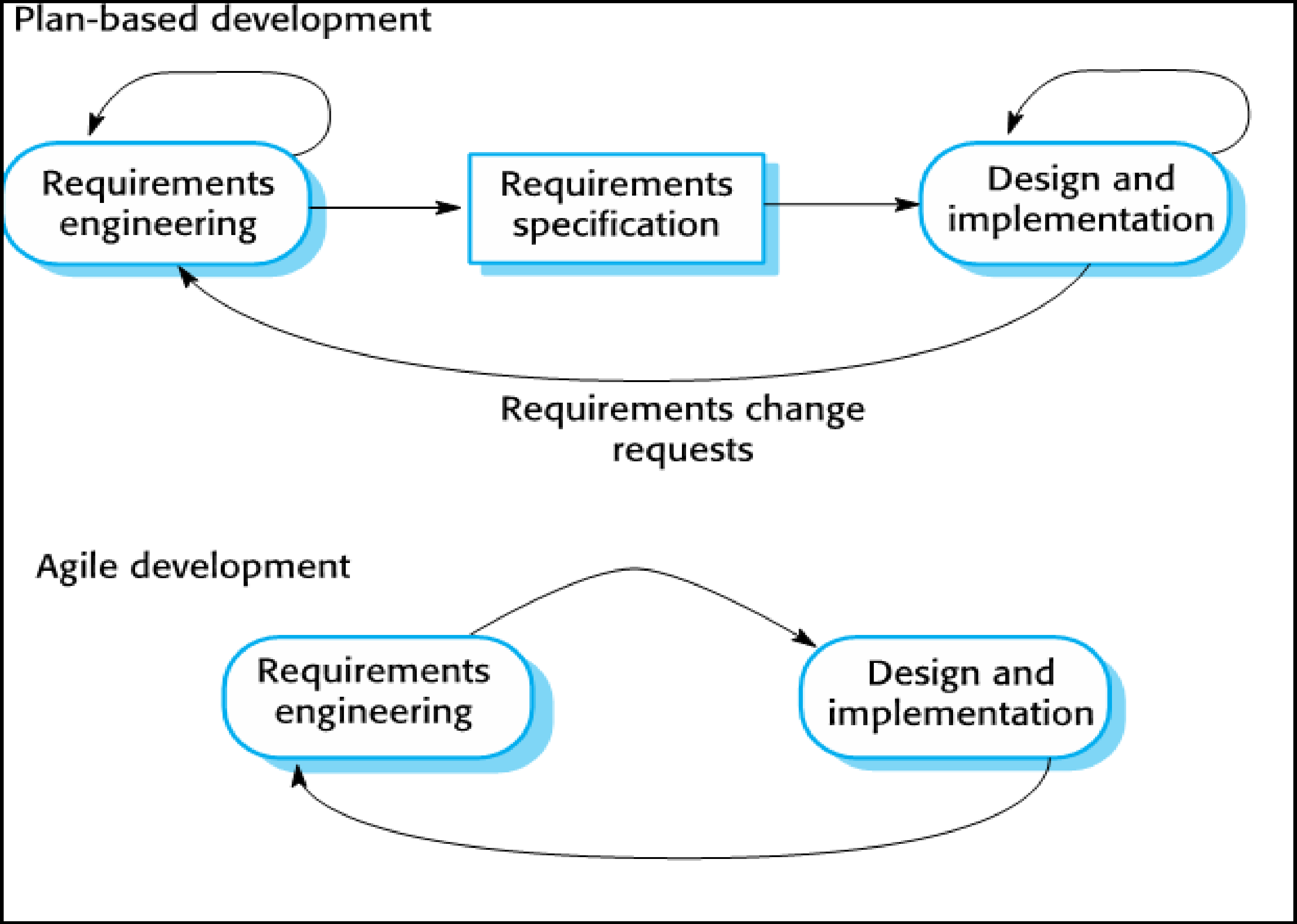
Agile Development Characteristics

- Program specification, design and implementation are inter-leaved
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation – focus on working code

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.

Plan-driven and Agile Development


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

Agile Methods

Agile Methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
    - Focus on the code rather than the design
    - Are based on an iterative approach to software development
    - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
- The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

Agile Manifesto

- *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

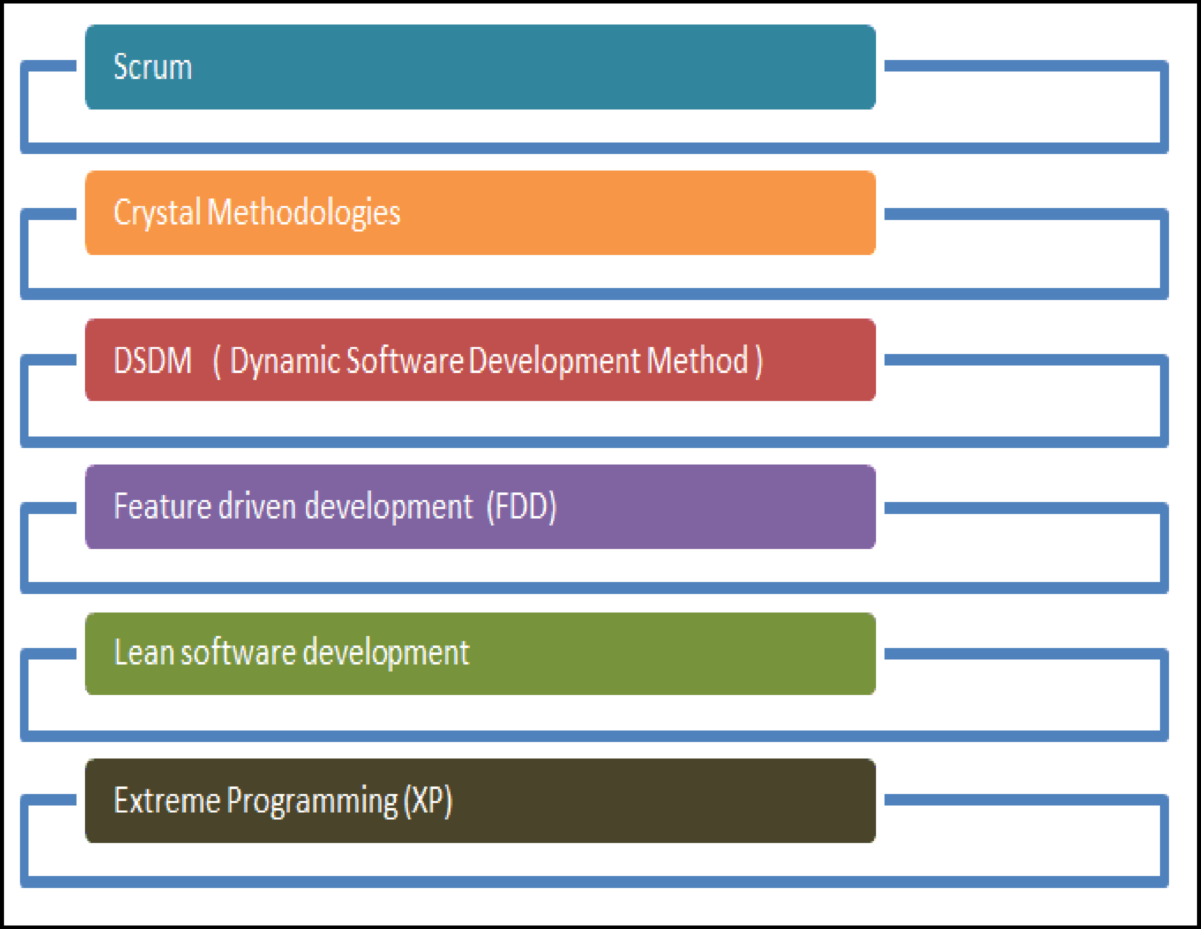
  Individuals and interactions over processes and tools
  Working software over comprehensive documentation
  Customer collaboration over contract negotiation
  Responding to change over following a plan

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

Agile Core Values

- **Individuals and interactions** over processes and tools---in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** over comprehensive documentation---working software will be more useful and welcome than just presenting documents to clients in meetings.
- **Customer collaboration** over contract negotiation---requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- **Responding to change** over following a plan--- agile development is focused on quick responses to change and continuous development.

Agile Methods



Scrum

Crystal Methodologies

DSDM (Dynamic Software Development Method)

Feature driven development (FDD)

Lean software development

Extreme Programming (XP)

The 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.

Agile Method Applicability

- Product development where a software company is developing a small or medium-sized product for sale.
  - Virtually all software products and apps are now developed using an agile approach
- 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 few external rules and regulations that affect the software.

Agile Development Techniques

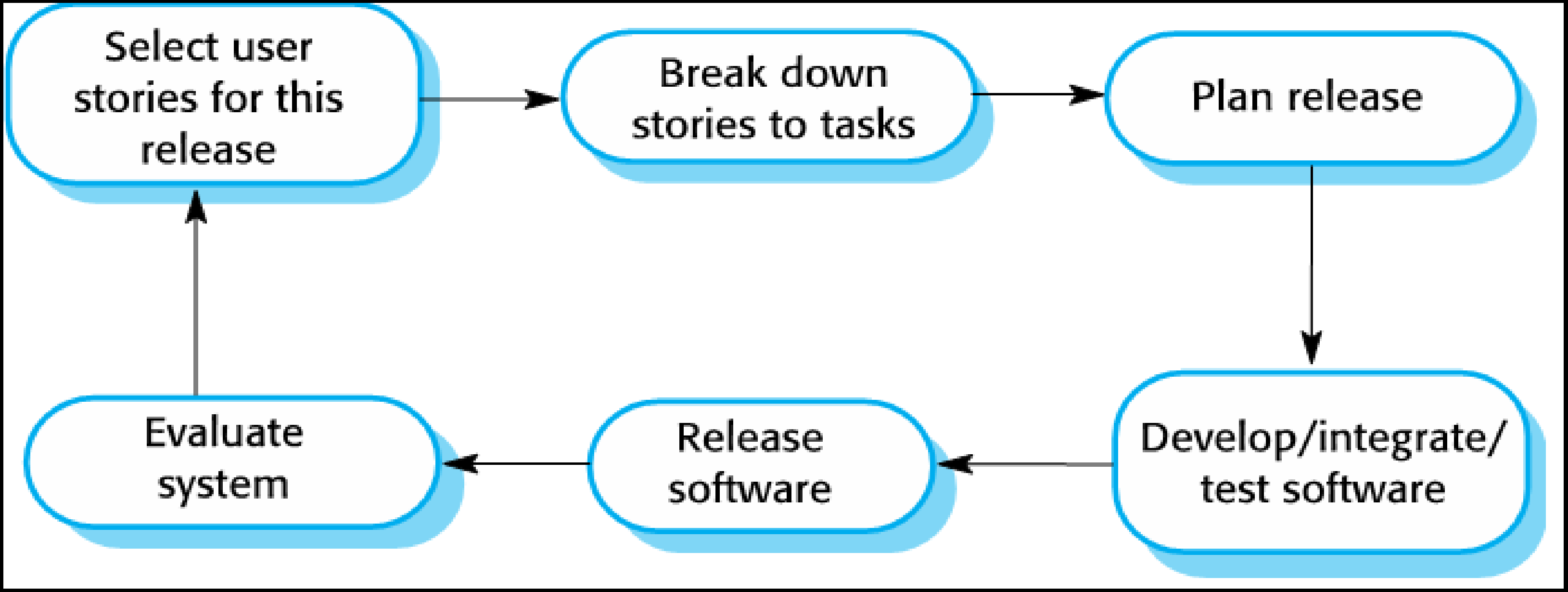
Extreme Programming(XP)

- The origin of extreme programming (XP) started in 1990s when Kent Black tried to find a better way of doing software development when he was handling a project. One significant difference in its approach is that it focuses on adaptability rather than on predictability. The reason behind this approach is that software development is a very fluid process where requirements cannot be fully predicted from the beginning but will always change as projects move on. Hence software development needs a methodology that is capable to adapt to changing requirements at any point during the project life.

Extreme Programming

- A very influential agile method, developed in the late 1990s, that introduced a range of agile development techniques.
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

The Extreme Programming Release Cycle


Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

Extreme Programming Practices (a)

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.

Extreme Programming Practices (b)

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.

XP and Agile Principles

- Incremental development is supported through small, frequent system releases.
- Customer involvement means full-time customer engagement with the team.
- People not process through pair programming, collective ownership and a process that avoids long working hours.
- Change supported through regular system releases.
- Maintaining simplicity through constant refactoring of code.

That is all