Software Development Process

Lecture 3
Agile Process

Toward Agile Process

- Traditional software processes are “heavy” and does not cope well with changes
- They emphasize
  - Planning
  - Requirement documentation
  - Design documentation
- **Problem**: We might not know what we actually require
- **Agile Alliance** is formed in 2001 to produce development framework that is efficient and adaptable
- As a result, they produced the **Agile Manifesto**
- **Agile Process** or **Agile Development** refers to any software process that captures the values in Agile Manifesto

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 (aka **Agile Values**):

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

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

1. Individuals and Interactions

- Agile promotes cooperation and communication among experts to enhance their efficiency
- The development team should be self-organized and self-motivated
- Skilled individual should be able to adapt the tools to their needs, not the other way round
- Creativity should be used to solve problems, not predetermined rules and for the team to adapt to these rules

2. Working Software

- Working software is the best indicator how well the project is going, not documents presentations
- Agile practices emphasize producing working program as soon as possible
- As the project progresses, more functionality will be incrementally added
- Stakeholders can see the progress through the produced working codes
- However, appropriate documentation is still necessary

3. Customer Collaboration

- All stakeholders should work as the same team
  - From developers to customers
  - Sometimes they are split by organizational layers
- Because:
  - All requirements may not be determined only in the beginning
  - The software may have to be changed, fixed or revised many times during the development, their combined skills would be benefit to the development
- Although contracts are important, interaction is also important especially when dealing with changes

4. Responding to Change

- Agile method embraces change and continuous development, even late in development
- Change is natural and inevitable
- When changes occur, we shall adapt to it rather than stick to the old plan
- An iteration in agile methods usually last one to six weeks to be able respond with change early

Agile Levels

- Being agile can be defined in four levels
  - **Agile Values**: Agile philosophy or core values of being agile. They are defined in the manifesto
  - **Agile Principles**: Strategic approaches that support agile values. Also defined in the manifesto
    - What are the key elements of being agile?
    - What should we do/be to be agile?
  - **Agile Methods**: Implementations of agile principles (e.g. UP, Scrum, XP)
  - **Agile Practices**: Highly specific techniques employed in agile implementations (standup meetings, planning poker, product backlogs)

Agile Principles

- Our highest priority is to **satisfy the customer** through early and **continuous delivery** of valuable software.
- **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
- **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Agile Principle (2)

- Business people and developers **must work together daily** throughout the project.
- Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is **face-to-face** conversation.

Agile Principle (3)

- **Working software** is the primary measure of progress.
- Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to **technical excellence and good design** enhances agility.

Agile Principle (4)

- **Simplicity** -- the art of maximizing the amount of work not done -- is essential.
- The best architectures, requirements, and designs emerge from **self-organizing teams**.
- At regular intervals, the team reflects on **how to become more effective**, then tunes and adjusts its behavior accordingly.

Agile Methods

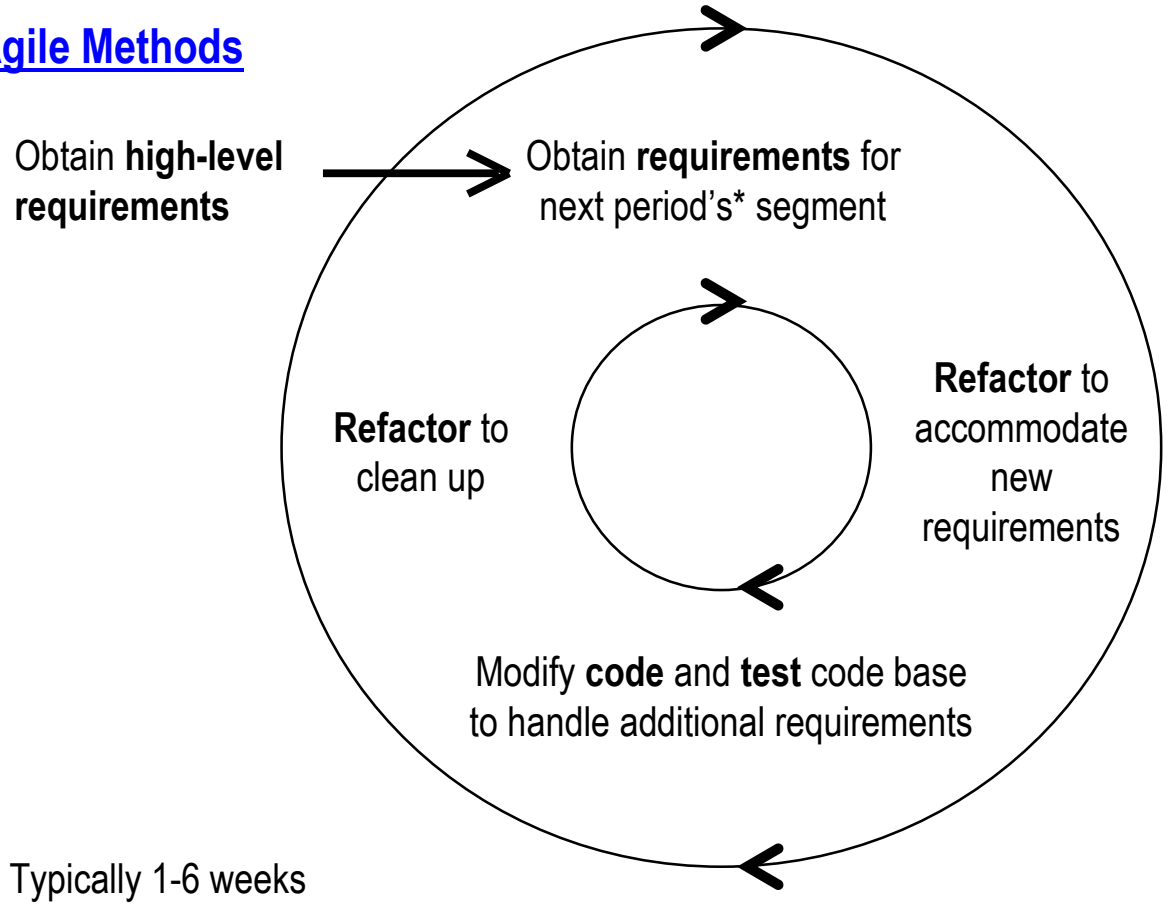
- The following table shows how agile methods implement the Agile Manifesto

Agile Processes MANIFESTO → RESPONSES:	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
a. Small, close-knit **team** of **peers**	y			y
b. Periodic **customer** requirements **meetings**	y		y	y
c. **Code-centric**		y		y
d. **High-level** requirements statements only			y	y
e. **Document as needed**			y	y
f. **Customer reps** work within team	y			y
g. **Refactor**				y
h. **Pair** programming and no-owner code	y			
i. Unit-test-intensive; Acceptance-**test-driven**		y	y	
j. **Automate** testing		y	y	

Table: Braude, E., Bernstein, M., Software Engineering: Modern Approaches 2nd Ed., John Wiley and Sons, 2011.

Agile Development Cycle

Agile Methods



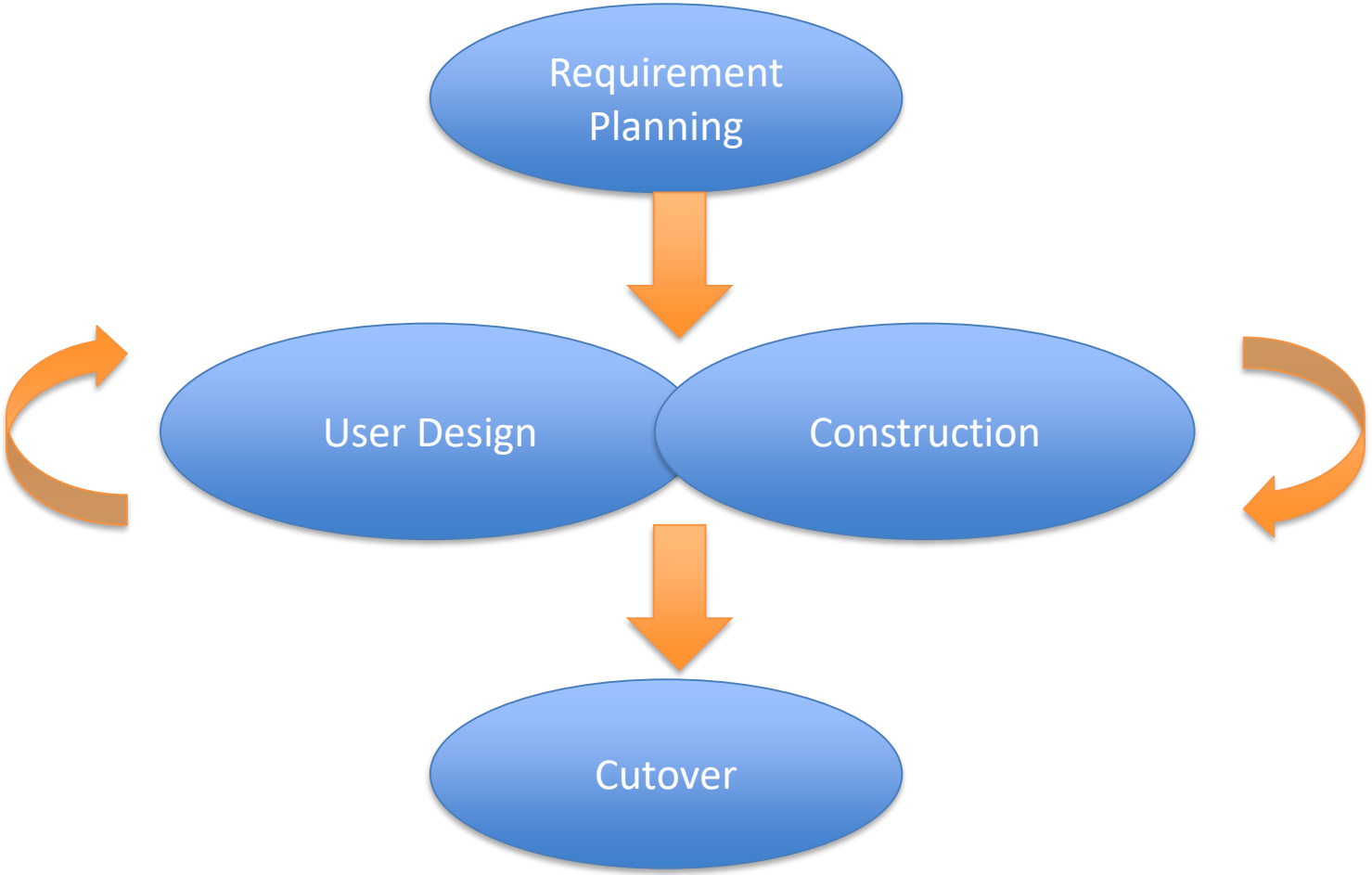
* Typically 1-6 weeks

- Agile method does not have explicit design phase
- It relies on refactoring to improve the design after each iteration

Rapid Application Development (RAD)

- **RAD**: A software development process that aims for short development time
- RAD Emphasizes
  - User involvement
  - Rapid prototyping
  - Use of automated development tools
  - Reusing resources
  - Small development teams
  - **Time box**: Fixed time frame in which activities are done. **Not scope box**!

RAD Life Cycle



Requirement Planning Phase

- In this phase, requirement elicitation and other project planning are carried out
- **Joint Requirement Planning** (**JSR**) workshop is organized in this phase
  - The goal of JSR is to obtain correct requirements right from the start
  - Key stakeholders, especially the end user, must participate in JRP
  - In JSR, requirements are gathered and prioritized
  - Result: Requirements that are sorted in **MoSCoW** manner

MoSCoW

- **MoSCoW prioritization** is used in software development to determine importance of requirements or functionalities
- MoSCoW categories:
  - **Must haves**: The functionalities must be implemented in current iteration
  - **Should haves**: The functionalities are important but is not an absolute necessary
  - **Could haves**: Desirable functionalities that are not necessary but preferred to have
  - **Won't haves**: Unimportant requirements that can wait until later releases

User Design Phase

- The goal of this phase is the initial system design
- In RAD, **Joint Application Design** (**JAD**) workshops are organized
  - Users must present in the workshop and get involved in the design process
  - The system analyst and developer brainstorm with the users and translate their requirements into models
- CASE tools maybe used to communicate with the users
- Then, a prototype is developed evaluated by the users during ***second*** JAD
- The users and developers may iterate the designs and prototype many times

Construction Phase

- The construction is done by a **Skilled With Advanced Tools**' (**SWAT**) team
- The SWAT estimate the project time and its time boxes
- They also assign themselves what to do within each time box
- Because SWAT assign their own time and scope, they would be more responsible to deliver good product in short timespan
- SWAT constructs a number of prototypes within the agreed time box
  - The users continue to closely involve in the development process
  - Time box cannot be postponed!

Cutover Phase

- Product delivery phase: System testing, deployment and user training

- There are many variations of RAD but key components are the same
  - User involvement
  - Prototyping (may not always be a “software”)
  - SWAT teams
  - Time boxes

Dynamic Systems Development Method (DSDM)

- It is an agile method which builds on RAD
- Maintained by DSDM Consortium
- DSDM Principle
  - Fixed time and resources (time box)
  - Requirements are prioritized using MoSCoW
  - Embraces agile development
- Complete set of DSDM practices are available only to consortium members

DSDM Stages

- **Feasibility**: Delivers feasibility report and outline plan, optionally fast prototype (few weeks)
- **Business study**: Carry out high-level description of business process through workshops (e.g. JSR). High-level architecture is also determined
- **Functional model iteration**: Produce analysis models and functioning prototypes
  - Time-boxed in 2-6 weeks
  - Four activities in an iteration: What to do, how will we do, do it, check if we have done it
- **Design and build iteration**: Integrate the functional components from the previous phase into one system
- **Implementation**: System testing, product deployment and train the user

DSDM Practices

- Active user involvement is imperative
- Empowered teams
- Frequent delivery of products
- Acceptance determined by fitness for business purpose
- Iterative, incremental development
- All changes are reversible
- Requirements baselined at high level
- Testing integrated in life cycle
- Collaborative, cooperative approach shared by all stakeholders is essential