CSC207H Lecture 5

Sadia Sharmin

Oct 2, 2016

Note-takers needed

- The AccessAbility Resource Centre is looking for a volunteer note-taker to take notes on behalf of students with a disability registered in this class.
- Volunteer note-takers can upload their notes online or scan them at the AccessAbility office. Note-takers will receive a certificate and a reference letter at the end of the year.
- If you are interested in this opportunity, please follow the instructions here: www.utm.utoronto.ca/accessability/potential-notetakers
- If you have any questions, please call 905-828-5422, email accessvolunteers.utm@utoronto.ca, or drop by the Centre (room 2037, Davis Building).

Model View Controller

- A software architecture that separates the application into three parts: Model, View, Controller
- Separates areas of concern, lowering their dependencies on each other
- Improves maintainability, extensibility, reusability

Model View Controller

- Model: the objects, data, business logic (internal)
- View: the presentation that allows outside world to interact with application, i.e. via user interface (external)
- Controller: connects the model with the view

"We need SMART Models, THIN Controllers, and DUMB Views"

Source: http://c2.com/cgi/wiki?ModelViewController

Model View Controller

- The controller mediates communication between the model and the view so that they can react to changes in each other's state
- This can be done directly, or via an application of the Observer design pattern

Observer Design Pattern

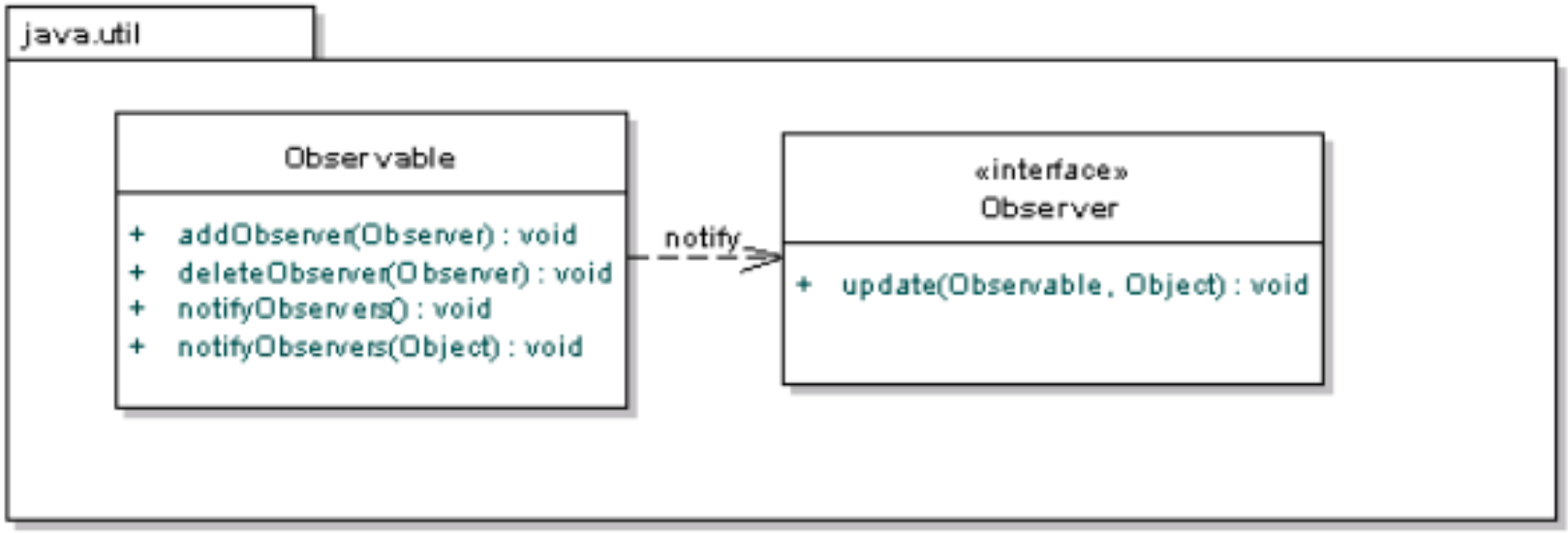
- One-to-many dependency between objects
- When one object changes state, all its dependents are notified and updated automatically

Source: http://c2.com/cgi/wiki?ObserverPattern

Observable and Observers

- In Java, you can make a class extend Observable; now this class can act as a Subject
- Other classes can extend Observer; use addObserver() to hook up an observer to the subject that it wants to observe
- Now, if a change occurs in the subject's state, call setChanged() and then notifyObservers() to let all the observers know of this change
- All observers have an update() method which is called once they are notified of a change

Observable and Observers



Software Engineering

- Historically, software used to often be written by a single developer
- As software grew more complex, teams of people started working on single projects
- Increased budgets and risk caused companies to start finding ways to try and minimize potential pitfalls

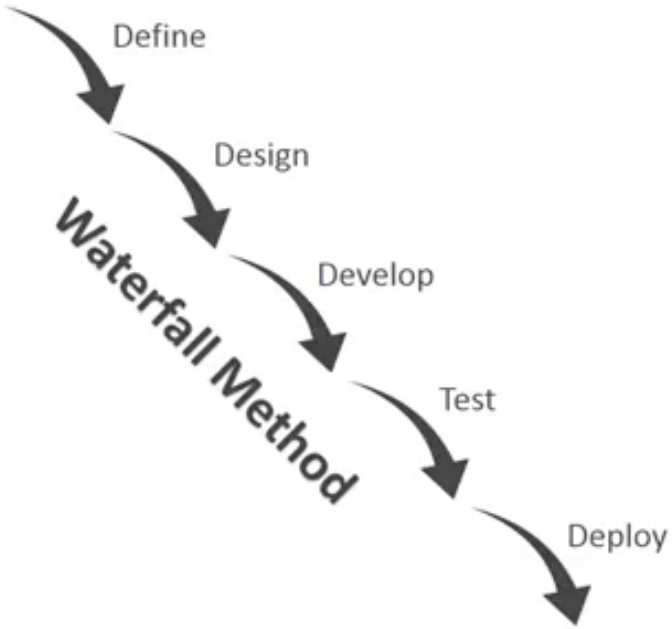
Waterfall Method

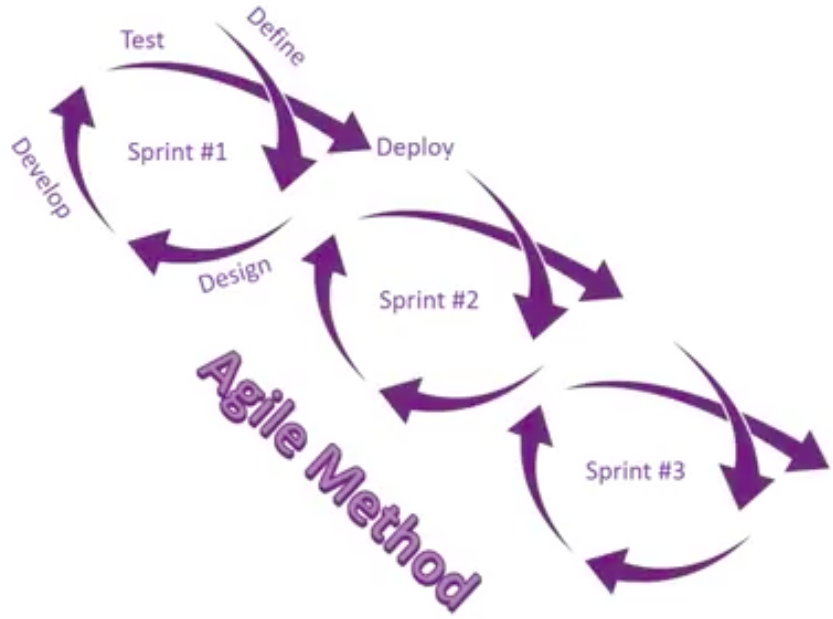
- Rigid, linear approach
- All requirements set before design, frozen before development
- Go through each phase one at a time
- Hostile to change
- No back-tracking

Agile Development

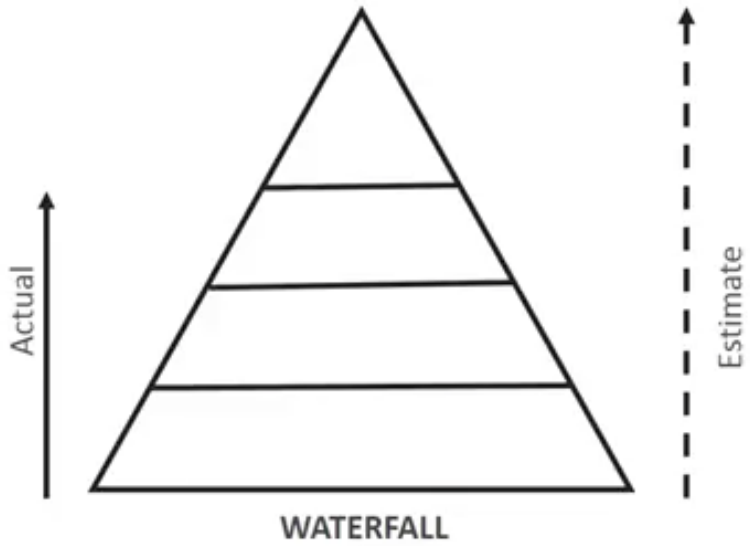
- Iterative approach
- A little bit of work done on each phase everyday
- Functional product (deliverables) ready after every increment
- Embracing change
- Continuous revisions; frequent feedback

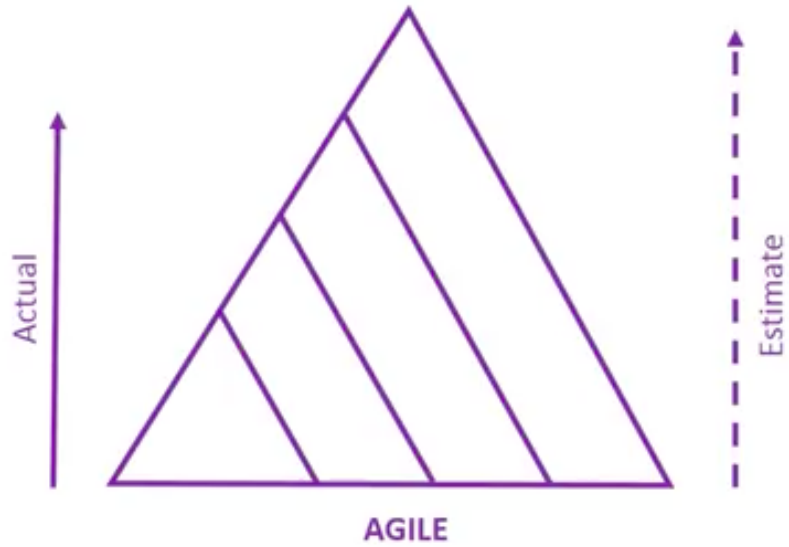
Waterfall vs. Agile





Waterfall vs. Agile





Waterfall vs. Agile

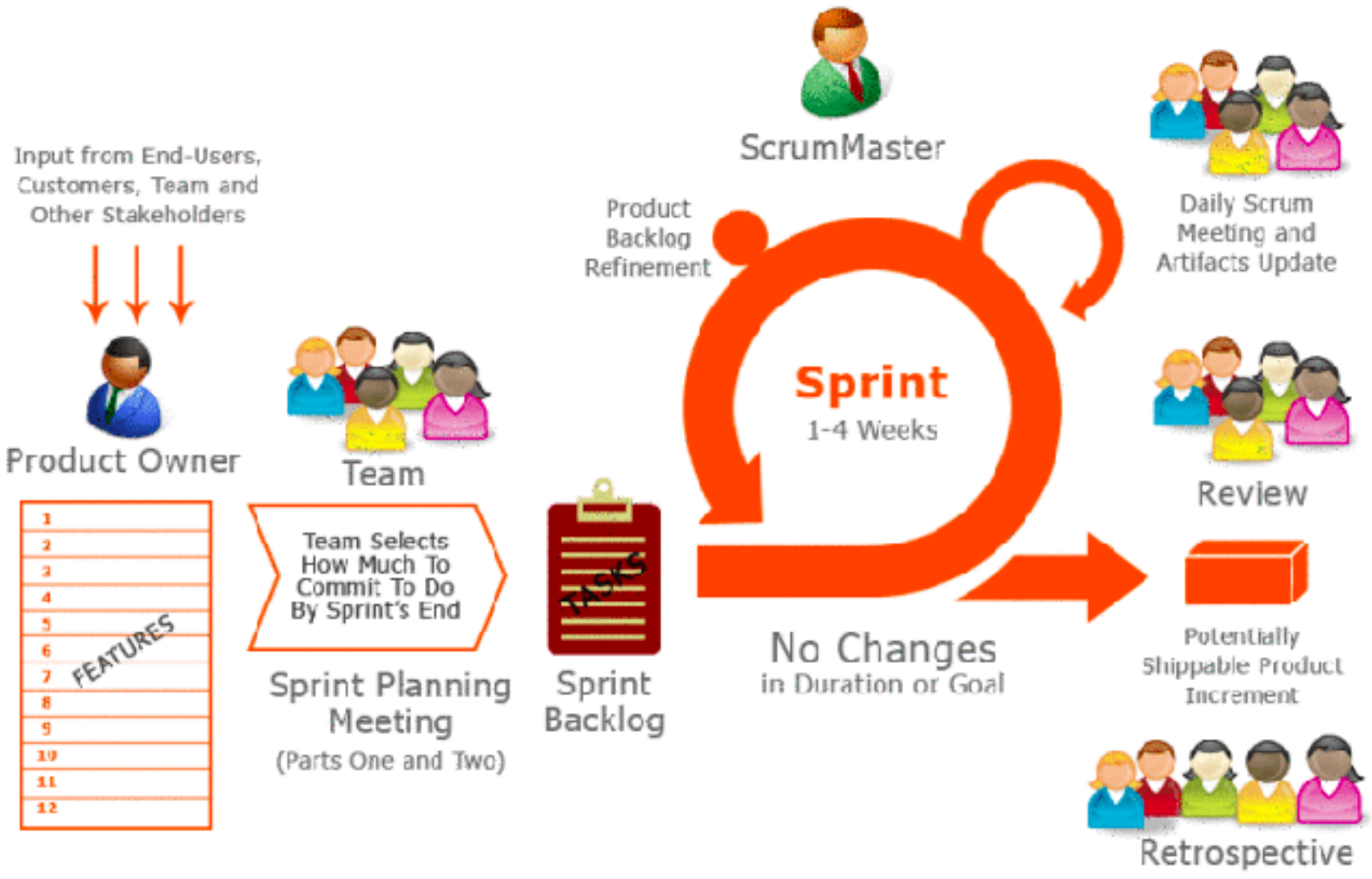
- https://www.youtube.com/watch?v=swWmVdaMIoI
- https://www.youtube.com/watch?v=6k2CDxjQVa8
- https://www.youtube.com/watch?v=PHS-ycbRwqI

Scrum

- An agile methodology
- Work is done in short (2-4 week) iterations called "sprints"
- Should have a visible, functional product at end of each sprint

A Scrum Sprint

A Scrum Sprint



Steps to Software Design

- Requirements, design, construction, testing
- "Rather than doing all of one thing at a time, Scrum teams do a little of everything all the time"

Source: The New New Product Development Game by Takeuchi and Nonaka. Harvard Business Review, January 1986.

Roles

- Product Owner
- ScrumMaster
- Team

Events

- Sprint planning
- Sprint review
- Spring retrospective
- Daily scrum meeting

The Product Owner

- Responsible for the product backlog and maximizing the product ROI.
- Represents the users
- Clearly expresses backlog items
- Orders them by value
- Ensures visibility

The Development Team

- Responsible for delivering a potentially shippable increment of working software.
- Self-organized
- 4 to 9 persons

Scrum Master

- Responsible for the scrum process
- Removes impediments
- Facilitates scrum events
- Facilitates communication

The Product Backlog

- Single source of requirements for any changes to be made to the product.
  - Living list that is never complete
  - Ordered: value, risk, priority and necessity
  - Estimated by the team

Sprint Planning Meeting

- Defines what will be delivered in the increment
- Team selects items from the product backlog and defines a sprint goal
- Defines how the increment will be achieved
- Items are converted into tasks and estimated

Sprint Planning Meeting Example

Current state of the project:

- done: `Grade`, `LetterGrade`, `NumericGrade`
- implemented: `Person`
- almost done: main activity, enter info activity
- ...

Tasks for this week:

- implement `Student` (Alex)
- test `Person` (Jen)
- add a "Save" button to enter info activity (Gary)
- get the integration between the GUI and `Person` class to work (Jen)
- ...

The Daily Scrum

- 15 minute meeting for the Team to synchronize activities.
  - What has been accomplished since last meeting?
  - What will be done before the next meeting?
  - What obstacles are in the way?

Sprint Review

- Product owner identifies what has been done
- Team discusses what went well, what problems it ran into and those that were solved
- Team demonstrates what it has done in a demo
- Product owner discusses the backlog as it stands
- Entire group collaborates on what to do next

The Sprint Retrospective

- Improves the process.
- Inspect how the last Sprint went
- Identify and order the major items that went well and potential improvements; and,
- Create a plan for implementing improvements