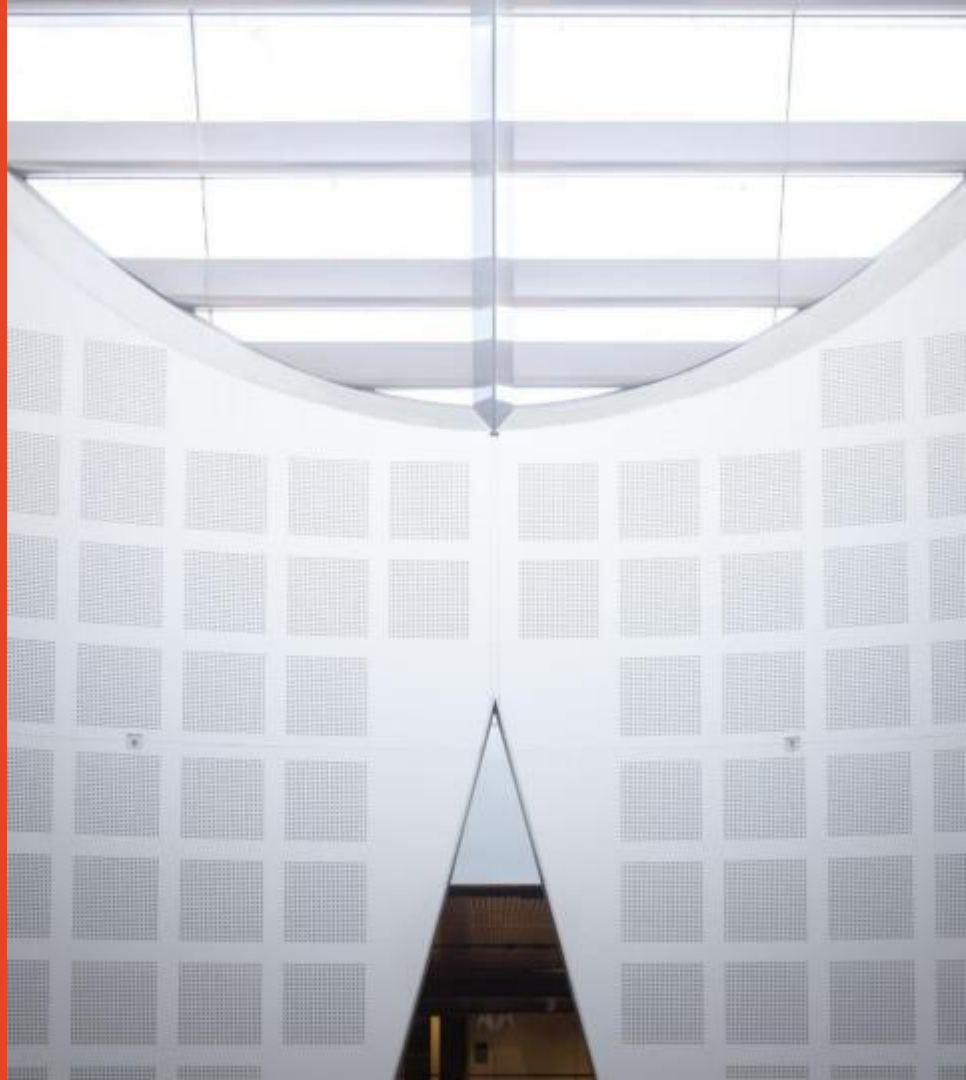
Agile Software Development Practices
SOFT2412 / COMP9412

Software Development Process Models

Xinyi Sheng
School of Computer Science

Slides based on Dr. Basem Suleiman and Dr. Farshid Hajati



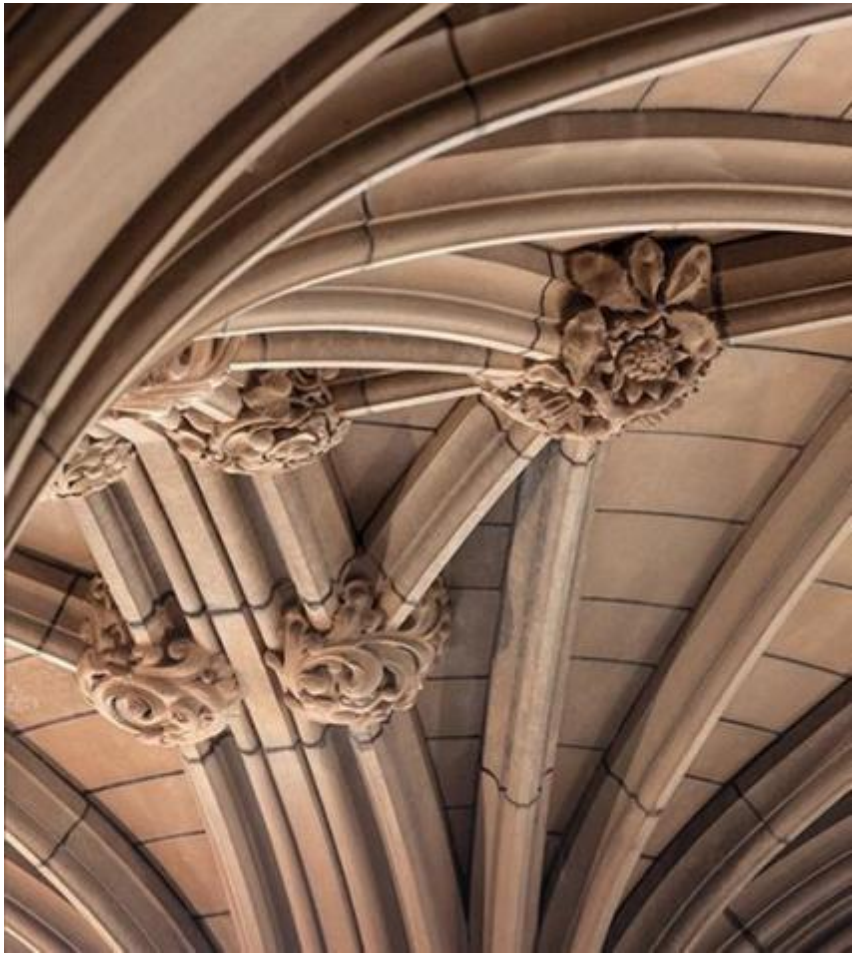
Agenda

- Software Engineering
  - Software Development

- Introduction
  - Software processes
  - Software development process models
  - Agile Development Model
  - Agile development Values and Principles

Software Engineering

- What is Software Engineering?
- Why Software Engineering?



THE UNIVERSITY OF SYDNEY

Software is Everywhere!

- Societies, businesses and governments dependent on SW systems
  - Power, Telecommunication, Education, Government, Transport, Finance, Health
  - Work automation, communication, control of complex systems
- Large software economies in developed countries
  - IT application development expenditure in the US more than $250bn/year[1]
  - Total value added GDP in the US[2]: $1.9 trillion
- Emerging challenges
  - Security, robustness, ethic, human user-interface, and new computational platforms

[1] Chaos Report, Standish group Report, 2014
[2] https://software.org/reports/software-supporting-us-through-covid-2021/

Why Software Engineering?

Need to build high-quality software systems under resource constraints

- Social
  - Satisfy user needs (e.g., functional, reliable, trustworthy)
  - Impact on people's lives (e.g., software failure, data protection)
- Economical
  - Reduce cost; open up new opportunities
  - Average cost of IT development ~$2.3m, ~$1.3m and ~$434k for large, medium and small companies respectively[3]
- Time to market
  - Deliver software on-time

[3] Chaos Report, Standish group Report, 2014

Software Failure - Ariane 5 Disaster[1]

What happened?

- European large rocket - 10 years development, ~$7 billion
- Unmanaged software exception resulted from a data conversion from 64-bit floating point to a 16-bit signed integer
- Backup processor failed straight after using the same software
- Exploded 37 seconds after lift-off

Why did it happen?

- Design error, incorrect analysis of changing requirements, inadequate validation and verification, testing and reviews, ineffective development processes and management

[1] http://iansommerville.com/software-engineering-book/files/2014/07/Bashar-Ariane5.pdf

What is the difference between SW Developers and SW Engineers?

What is Software Engineering?

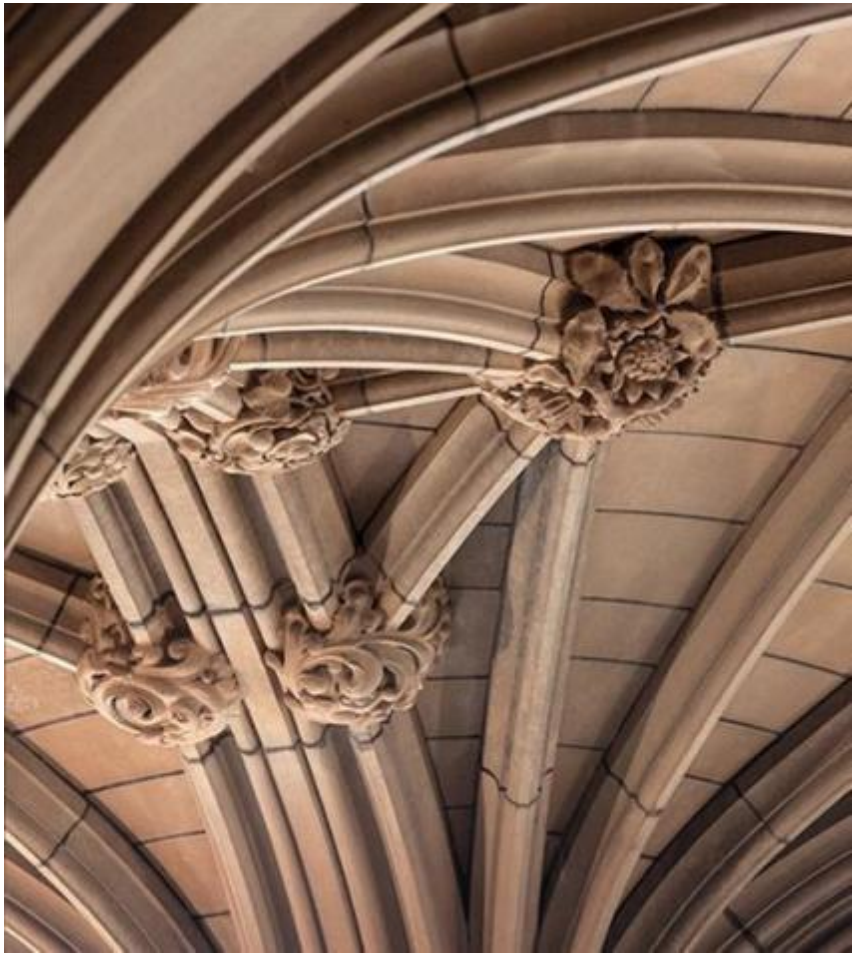
http://www.purplesoft.com.au/wp-content/uploads/2017/03/software.jpg

Software Engineering

"An engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining it after it is has gone into use."

- **NOT** programming/coding! a lot more is involved
- Theories, methods and tools for cost-effective software production
- Technical process, project management and development of tools, methods to support software production
- System Engineering (Hardware & Software) - software often dominates costs

Software Process



THE UNIVERSITY OF SYDNEY

The Software Process

- Software Development Process
  - Set of activities required to develop a software
  - Activities are to be done, and in what order
  - Lifecycle for a Software Development project
  - Processes, a set of tools, definitions of the Artifacts, etc.

- Is there a universally applicable software engineering process?
  - Many different types of software systems
  - Companies/engineers claim that they follow “methodology X”, but many times they only do some of what the methodology says

The Software Process

- Many software development processes, but all include common activities
  - **Specification (software/system requirements)**
  - **Design and implementation**
  - **Validation (testing)**
  - **Evolution**

- Software processes are complex and, rely on people making decisions and judgements
- Activities are complex and include sub-activities
  - E.g., requirements validation, architectural design, unit testing

Software Process Models

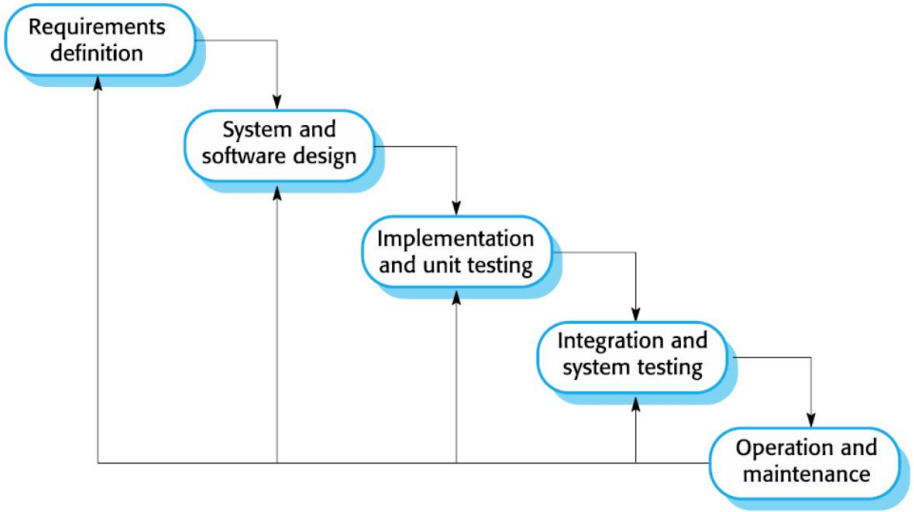
- Software Development Lifecycle (SDLC)

- Description of a process from particular perspective
  - Describe the activities and their sequence but may not the roles of people

Representative Software Process Models

- **Waterfall Model**
  - Development process activities as process phases
- **Spiral Model**
  - Incremental development risk-driven
- **Agile Model**
  - Iterative incremental process for rapid software development
- **The Rational Unified Process (RUP or UP)**
  - Bring together elements of different process models
  - Phases of the model in timer, process activities, good practices

Waterfall Model - Heavy-Weight Model



Development activities	Teams
Divide the work into stages	A separate team of specialists for each stage
At each stage, the work is passed from one team to another	Some coordination is required for the handoff from team to team – using "documents"
At the end of all of the stages, you have a software product ready to ship	As each team finishes, they are assigned to a new product

Ian Sommerville. 2016. Software Engineering (10th ed. Global Edition). Pearson

Waterfall Model Phases

- Requirement's analysis and definition
  - Requirements doc.
- System and software design
  - Design document based on requirements doc.
- Implementation and unit testing
  - Code and test it for system components (using design doc.)
- Integration and system testing
  - Software components are integrated and the resulting system is tested
- Operation and maintenance

Waterfall Model

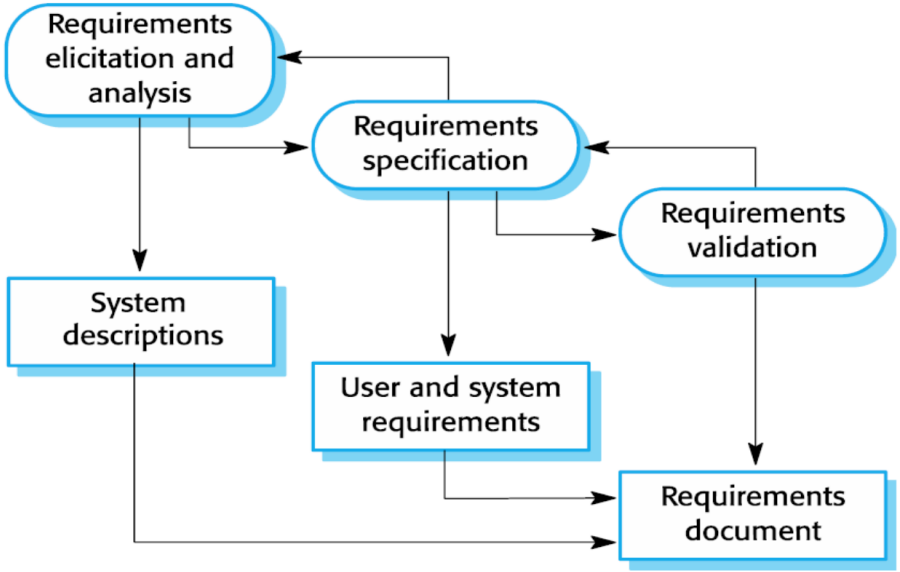
- Advantages:

- Easy to understand and implement
- Identified deliverables and milestones

- Disadvantages:

- Intensive documenting and planning
- Discovering issues in later phases should lead to returning to earlier phase!

Requirements Engineering Process



Requirements engineering
main activities and deliverables

Ian Sommerville. 2016. Software Engineering (10thed. Global Edition). Pearson

Planning in Software Development

- **Plan-driven (plan-and-document / heavy-weight)**
  - Activities are planned in advance and progress is measured against this plan
  - Plan drives everything and change is expensive
- **Agile processes (light-weight)**
  - Planning is incremental and continual as the software is developed
  - Easier to change to reflect changing requirements
- Most SW processes include elements of both plan-driven and agile
- Each approach is suitable for different types of software
  - No right or wrong software processes

Planning in Waterfall Model

- **Difficulty of accommodating change** after the process is underway

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements

- Mostly used for **large systems engineering projects** where a system is developed at several sites
  - The plan-driven nature of the waterfall model helps coordinate the work

Software Failures - Budget, Schedule, Requirements

Project	Duration	Cost	Failure/Status
e-borders (UK Advanced passenger Information System Programme)	2007 - 2014	Over £ 412m (expected), £742m (actual)	Permanent failure - cancelled after a series of **delays**
Pust Siebel - Swedish Police case management (Swedish Police)	2011 - 2014	$53m (actual)	Permanent failure – scraped due to **poor functioning, inefficient** in work environments
US Federal Government Health Care Exchange Web application	2013 – ongoing	$93.7m (expected), $1.5bn (actual)	Ongoing problems - **too slow, poor performance,** people get stuck in the application process (**frustrated users**)
Australian Taxation Office's Standard Business Reporting	2010 - ongoing	~$1bn (to-date), ongoing	Significant spending on **contracting fees** (IBM & Fjitsu), significant **scope creep** and **confused objectives**

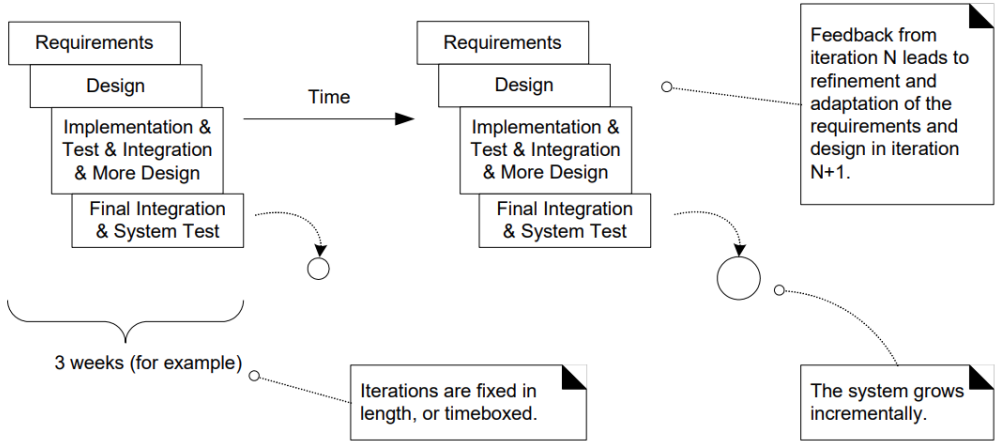
https://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

Software Evolution

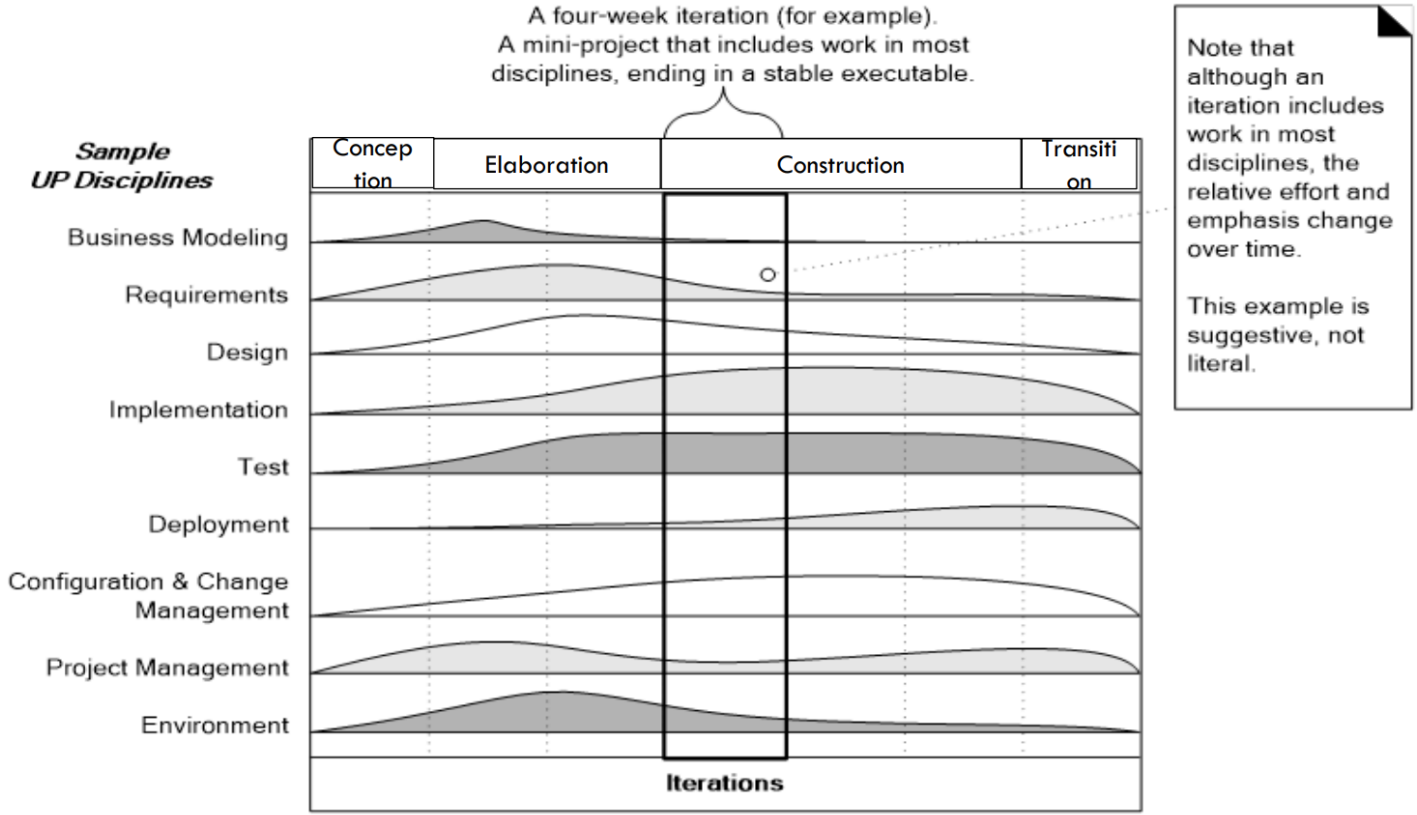
- Software is inherently flexible and can change
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change
- Business software needs to respond to rapidly changing market
  - Time-to-market
- Plan-driven software development processes are not suitable for certain types of SW systems

Rational Unified Process (RUP or UP)

- Software development process utilizing iterative and risk-driven approach to develop OO software systems
- Iterative incremental development
- Iterative evolutionary development



UP Phases and Disciplines



Agile Development Model

Project Failure - the trigger for Agility

- One of the primary causes of project failure was the **extended period of time it took to develop a system**
- Costs escalated and requirements changed
- Agile methods intend to **develop systems more quickly with limited time spent on analysis and design**

Agile Manifesto (2001) - An Eloquent Statement of Agile Values or Goals

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

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Agile Manifesto: http://agilemanifesto.org/

Agile Process

- Agile advocates believe:
  - Current SW development processes are too heavy-weight or cumbersome
  - Current software development is too rigid
    - Incomplete or changing requirements
  - More active customer involvement needed

Agile Process

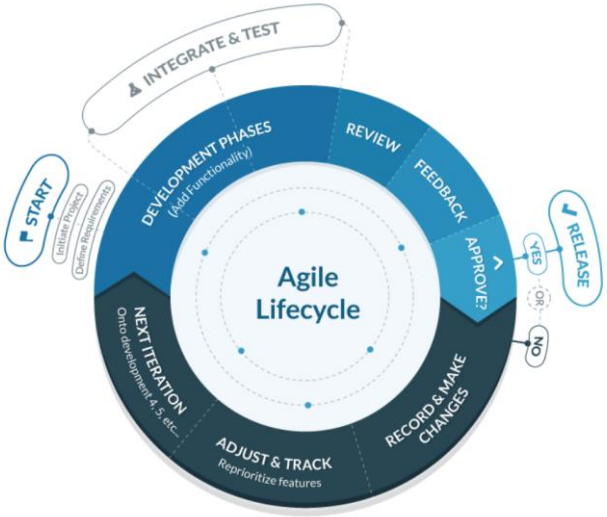
- Agile methods are considered
  - **Light-weight**
  - **People-based** rather than Plan-based
- Several agile methods
  - No single agile method
  - Extreme Programming (XP), Scrum
- Agile Manifesto closest to a definition
  - Set of principles
  - Developed by Agile Alliance

Agile Principles

1. Our highest priority is to **satisfy the customer** through early and **continuous delivery of valuable software**.	5. Build projects around **motivated individuals**. Give them the **environment** and support they need, and **trust** them to get the job done.	9. **Continuous attention** to technical excellence and good design enhances agility.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.	6. The most efficient and effective method of **conveying information** to and within a development team is **face-to-face conversation**.	10. **Simplicity--**the art of maximizing the amount of work not done--is essential.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the **shorter timescale**.	7. **Working software** is the primary **measure of progress**.	11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
4. Business people and developers must **work together daily** throughout the project.	8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a **constant pace** indefinitely.	12. At regular intervals, the **team reflects** on how to become more **effective**, then **tunes and adjusts its behavior** accordingly.

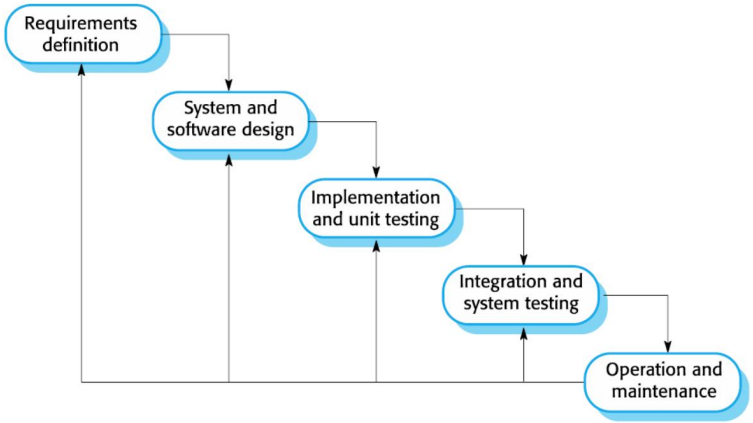
Agile Alliance: http://www.agilealliance.org

Software (Development) Process Models



Agile model
Incremental & iterative development

https://blog.capterra.com/agile-vs-waterfall/



Waterfall model
plan-driven development

Ian Sommerville. 2016. Software Engineering (10th ed. Global Edition). Pearson

Agile - Requirements process

- There is no "standard way" to do requirements in Agile development
  - Could be a normal "Software Requirements Document"
  - But it is better to be more lightweight
  - Based on user stories
  - In each iteration, elaborate a small set of the functional requirements (the high-priority behavior)
  - For some key requirements, create some acceptance tests at the same time as you write the requirements

Agile - Questions and Challenges?

- **Documentation**
  - Still important in an Agile development
  - If it is the only kind of communication in your project, it isn't good
  - Real working code is more valuable than documents

- **Development plans**
  - Still important in an Agile project
  - the format of an Agile development schedule is a bit different
  - Development plan includes "iterations"
  - Each iteration gives the team a chance to incorporate what they learn

Agile only works with the best developers

- Every project needs at least one experienced and competent lead person. (Critical Success Factor)
- Each experienced and competent person on the team permits the presence of 4-5 “average” or learning people.
- With that skill mix, agile techniques have been shown to work many times.

Mutual Respect

- *Developer*: “Testers are failed programmers, they shouldn’t be called engineers”

- *Tester*: “Developers are only able to produce bugs, the world would be better with more testers and less developers”

- *Business Analysts*: “I don‘t know why I bother talking to testers and developers, they are total idiots”

Not My Responsibility

- *Developer*: “It works in our environments, it’s operations responsibility to make it work in production”

- *Tester*: ”Listen, it worked in User Acceptance Testing, it must be a configuration issue, or a missing firewall hole and nothing I could have spotted during testing…”

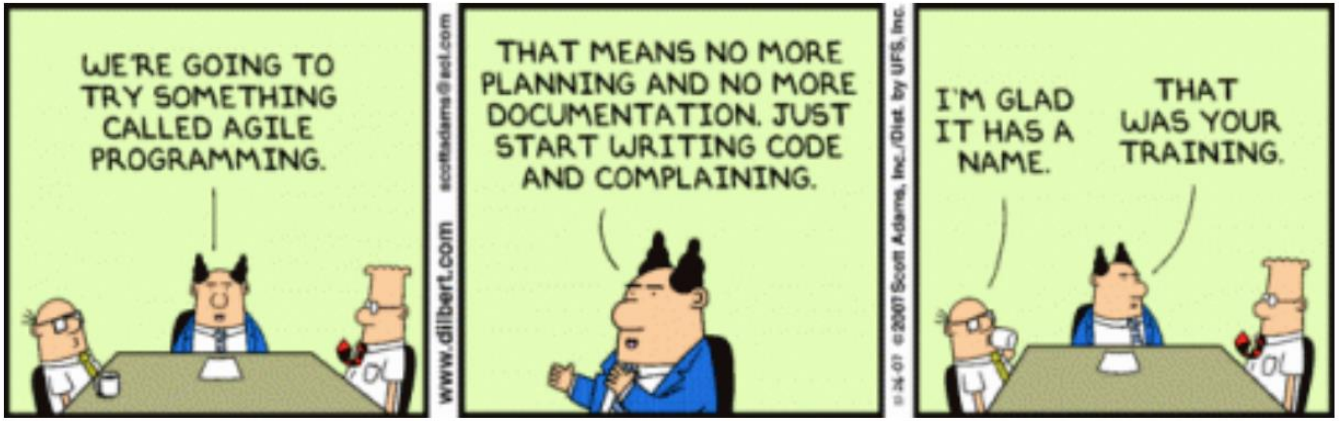
- *Customer*: “Hello! Nothing works here…”

How hard is it to be Agile?

- "Don't do Agile, be Agile"
  - Just doing "development in iterations" isn't enough

- Agile Development is about:
  - Keeping the process lightweight
  - Making real progress in each iteration
  - Communicating – face-to-face when possible
  - Actively gathering customer input – early and often
  - Willing to make minor changes to your process

Thinking Agile!



https://www.eylean.com/blog/2016/04/best-agile-comic-strips/

References

- Armando Fox and David Patterson 2015. Engineering Software as a Service: An Agile Approach Using Cloud Computing (1st Edition). Strawberry Canyon LLC

- Andrew Stellman, Margaret C. L. Greene 2014. Learning Agile: Understanding Scrum, XP, Lean and Kanban (1st Edition). O'Reilly, CA, USA.

- Ivar Jacobson, Grady Booch, and James Rumbaugh. 1999. The Unified Software Development Process. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

- Ian Sommerville. 2016. Software Engineering (10thed.) Global Edition. Pearson, Essex England

Tools and Techniques for Controlling Software Artifacts

- Artifacts in Software Development
- Version Control Systems
- Version Control with Git
- Using Git Commands

Next week Lecture and tutorial