Software Engineering

Lecture 1

Dept. of Computer Science

Dr Shekoufeh Rahimi

Dr Sobhan Yassipour

University of Roehampton London

Aim of the Course

The aim of the Software Engineering module is to strengthen students' capabilities in programming and software development. The module will require students to implement and debug their programs and utilise modern software development tools, such as Integrated Development Environments and debuggers.

Assessment

- Project (group work): 70%
- Examination: 30%

Assessment		Weight	Submission method	Due date
Coursework	Sprint 1	70%	Moodle	12/02/2023
	Sprint 2		Moodle	12/03/2023
	Sprint 3		Moodle	02/04/2023
	Sprint 4		Moodle	30/04/2023
	Individual reflection		Moodle	30/04/2023
Exam	Written Exam	30%	in person	TBC

Points to Consider

- You need a team that will be able to complete the coursework. Sometimes that means finding new people to work with
- Your team **must** come to the same lab session for code reviews and weekly lab meetings
- Your team **must** be four people. Only once all teams are formed will we look at how the numbers work out
- You are a team member. You are committing to the team and working together

Get into a Team

- Get up
- Find your team of four
- Once in a team sit down as close to the front **as possible**
  - Not as close to the front as you want – we are making space for everyone
- Once seated, your team need to do the following:
  - Define a Code of Conduct
  - Start reading the notes for Lecture 2
  - Decide on a Product Owner
  - Decide on a Scrum Master
  - Schedule necessary meetings

University of Roehampton London

Code of Conduct

You require rules for your team. For example:

1. If a team member is late more than twice they have to buy everyone a coffee.
2. If a team member does not communicate for a week they will be reported at the weekly lab meeting.
3. etc.

The point of the Code of Conduct is for everyone to agree how they will act as a team. Your Code of Conduct must be added to your GitHub repository. See https://help.github.com/articles/adding-a-code-of-conduct-to-your-project/

Continuous Improvement

One final point comes from ideas of lean. We have two objectives:

1. **Have respect for people**
2. **Continuously improve**

Education is a form of continuous improvement - you will never stop learning. However, a key concept is this:

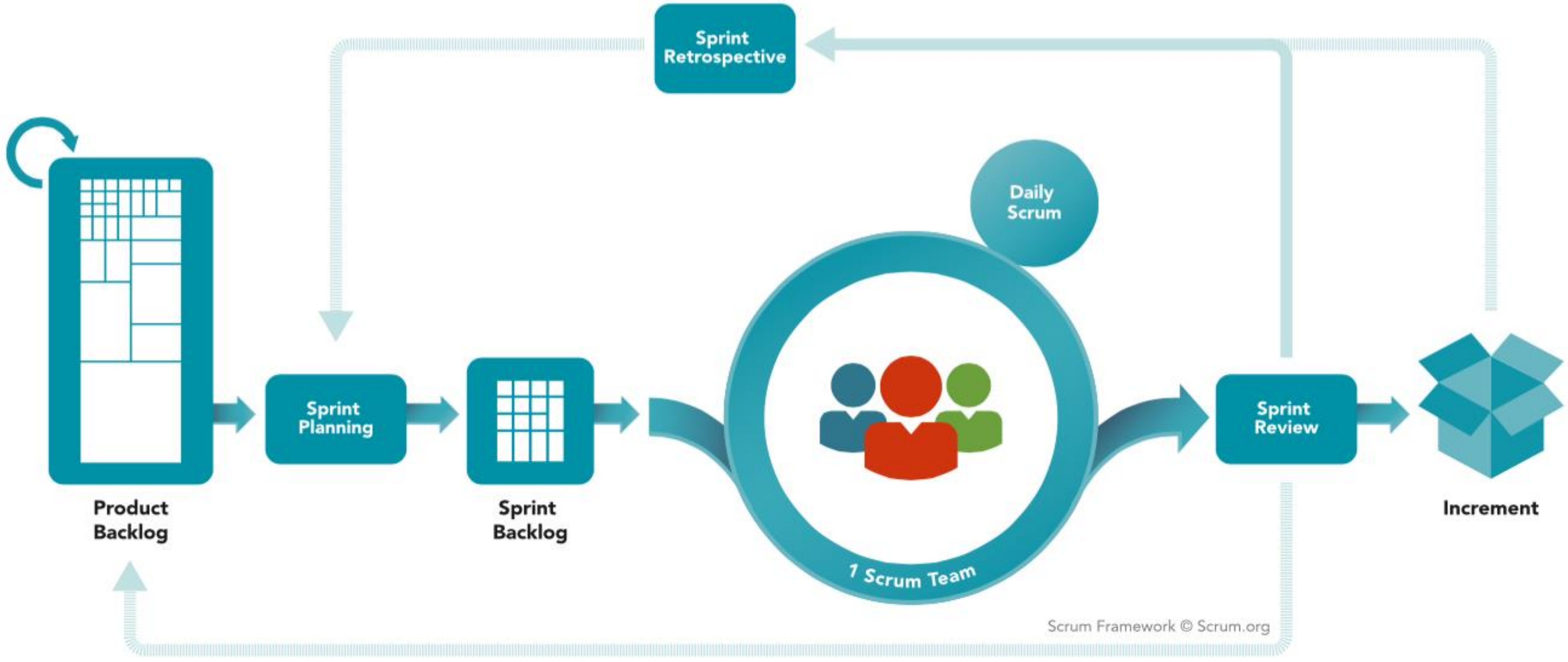
- *Respect for people* **supports** *continuous improvement*
- *Continuous improvement* **does not in itself support** *respect for people*

This can be expanded to the class - *respect* each other, and give each other support, and you can improve yourself and your class mates together

Education is not a competition

University of Roehampton London

Scrum Framework



Scrum Process

- Create a Product Backlog
- Estimate the Product Backlog
- Plan work to be done in a Sprint
- Start work
- Check-up each day (Daily Scrum)
- End of Sprint, review
- End of Sprint, reflect
- Check velocity, return to 2

Scrum Roles

- Product Owner prioritises the Product Backlog and generally maintains it
- Scrum Master encourages the team and gets problems out of their way. They schedule the meetings, but they are not in charge
- Team member does the work. In your teams, everyone is a team member
- The Product Owner and Scrum Master cannot be the same person

Your Task

READ THE LECTURE NOTES! They provide more information and detail than we can cover today

Get your team organised and working. Work on everything together. You are a team now and need to support each other

During next week's lab we will start team meetings. Be prepared and make sure everything is set-up

Web servers

Angular.js

CSS styling

Visual Studio

Scrum

Docker

Agile

Kanbon

User stories

Java

SQL database

HTML markup

IntelliJ

GitHub

Javascript

Node.js

Team working

Code of Conduct

Software Engineering Method

Two concepts to define:

- **Software Engineering**
- **Method**

We will start with Software Engineering – from Wikipedia:

- Software engineering is the **application of engineering** to the **development of software** in a **systematic method**.

Software Development

Again from Wikipedia:

- Software development is the process of **conceiving**, **specifying**, **designing**, **programming**, **documenting**, **testing**, and **bug fixing** involved in creating and maintaining applications, frameworks, or other software components

- **Conceiving** or coming up with an idea
- **Specifying** the requirements.
- **Designing** (we use UML)
- **Programming**
- **Documenting** is fundamental for reuse
- **Testing** (we will use unit testing).
- **Bug fixing** in production

Software Engineering

From Wikipedia:

- Engineering is the **creative application of science**, **mathematical methods**, and **empirical evidence** to the innovation, design, construction, operation and maintenance of structures, machines, materials, devices, systems, processes, and organizations.

Engineering has three strands:

1. The **application of science** – computer science underpins software development
2. **Mathematical methods** to model (not in the module). From computer science
3. **Empirical evidence** means that we measure something to gain information (e.g. testing)

Software Engineering

The working definition for the module:

A collection of techniques (e.g. designing, programming, testing) to develop software using ideas from computer science in a manner that is systematic. We require evidence about our software to determine if it is working as expected.

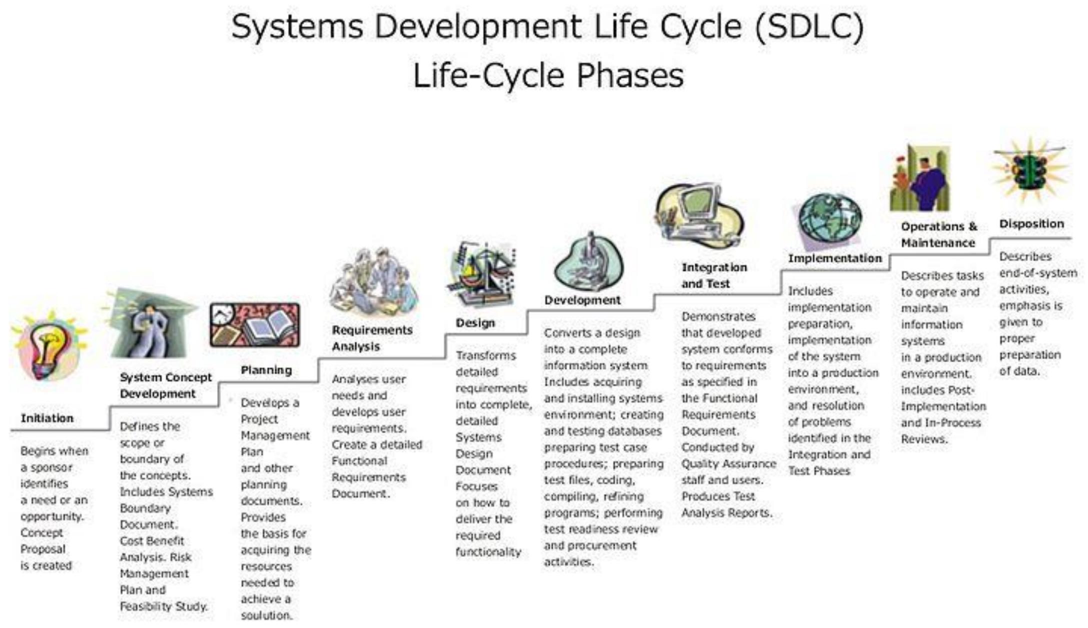
What is a Method?

Effectively, an approach to do something – a series of steps.

Software engineering is underpinned by the *Software Development Lifecycle.*

Three key methods:

1. Waterfall
2. Spiral or incremental
3. Agile



Initiation
Begins when a sponsor identifies a need or an opportunity. Concept Proposal is created

System Concept Development
Defines the scope or boundary of the concepts. Includes Systems Boundary Document. Cost Benefit Analysis. Risk Management Plan and Feasibility Study.

Planning
Develops a Project Management Plan and other planning documents. Provides the basis for acquiring the resources needed to achieve a soulution.

Requirements Analysis
Analyses user needs and develops user requirements. Create a detailed Functional Requirements Document.

Design
Transforms detailed requirements into complete, detailed Systems Design Document Focuses on how to deliver the required functionality

Development
Converts a design into a complete information system Includes acquiring and installing systems environment; creating and testing databases preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities.

Integration and Test
Demonstrates that developed system conforms to requirements as specified in the Functional Requirements Document. Conducted by Quality Assurance staff and users. Produces Test Analysis Reports.

Implementation
Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases

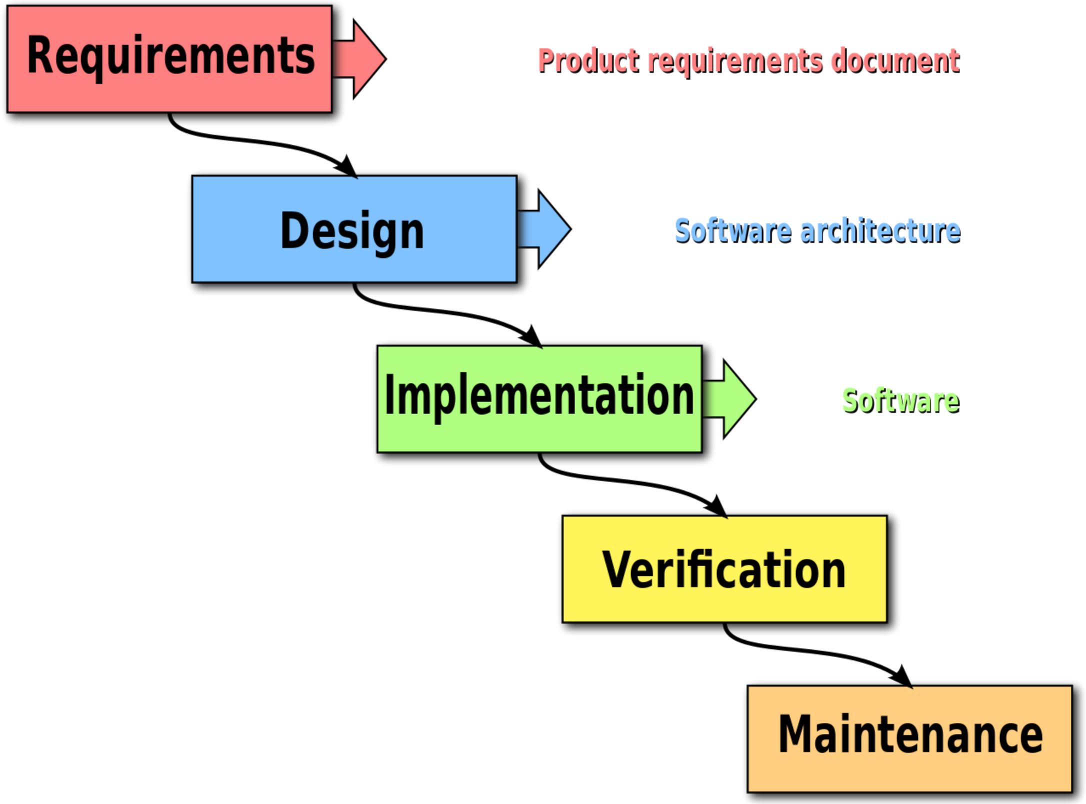
Operations & Maintenance
Describes tasks to operate and maintain information systems in a production environment. includes Post-Implementation and In-Process Reviews.

Disposition
Describes end-of-system activities, emphasis is given to proper preparation of data.

Waterfall

Each stage is completed before moving onto the next.

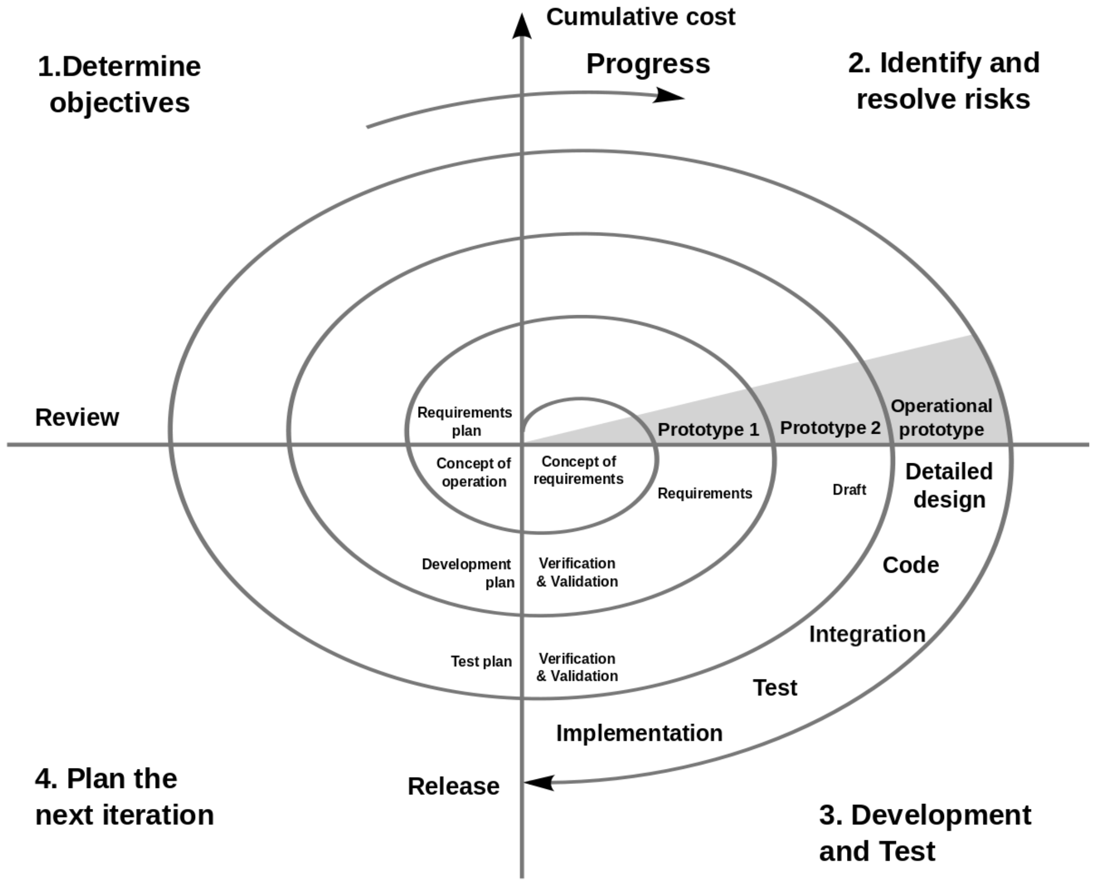
- **Requirements gathering** or defining what is needed in the software
- **Analysis of the requirements** to define models and rules
- **Design** to produce the software architecture
- **Coding** to build the software.
- **Testing** to ensure the software is working as expected
- **Operation** of the software where needed



Spiral

Software is iteratively developed through four stages to produce better prototypes.

1. **Determine objectives** for this iteration
2. **Identify and resolve risks** for this iteration
3. **Development and test** for this iteration
4. **Plan the next iteration**



University of Roehampton London

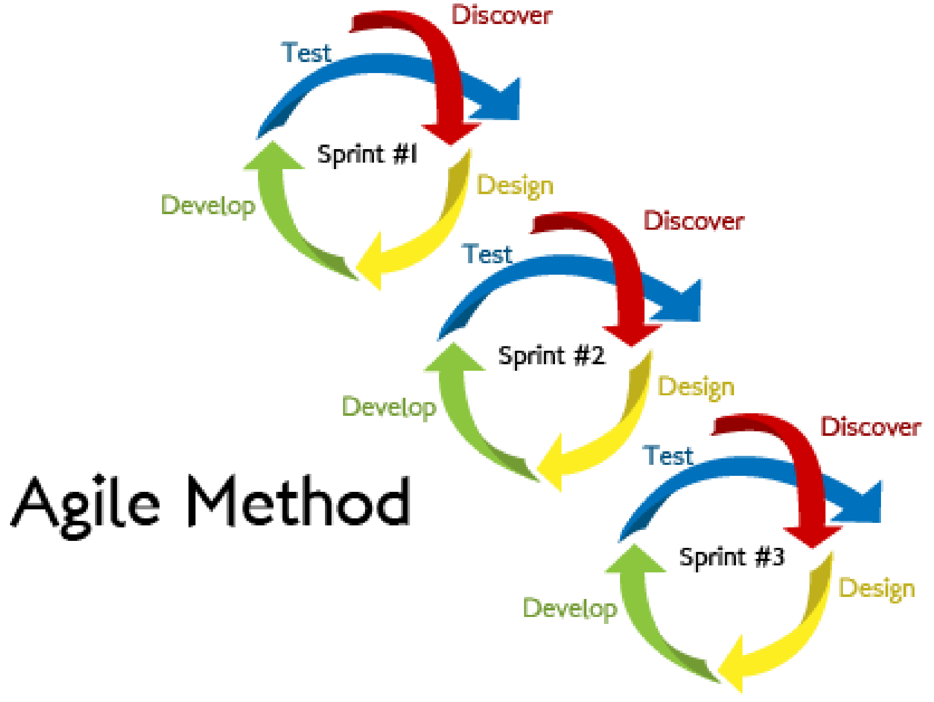
Agile Method

Extend the iterative ideas, but focus on **human-centric** ideas.

Software is **evolved** by collaboration between developers and clients.

Teams are **self-organising** and support multiple parts of the process.

Teams can **adapt** (*be agile*) as requirements and challenges evolve.



Linking to Module Learning Outcomes

1. **Demonstrate understanding** of *a modern software development lifecycle*. (**coursework**)
2. **Explain** the *different techniques supporting modern software engineering methods*. (**exam**)
3. **Define and analyse** *systems requirements and needs* and **specify** *a system design to deliver these requirements*. (**coursework**)
4. **Apply** *modern software engineering methods and techniques to a software development project*. (**coursework**)
5. **Explain** the *role of a computing professional in relation to social, ethical and legal issues surrounding projects*. (**exam**)
6. **Consider** *information security requirements in the development and delivery of software*. (**exam**)

History of Software Engineering (part 1)

Pre 1965 work on defining a discipline, but the term Software Engineering

1965 various letters to the ACM, lectures, and advertisements mention the term Software Engineering

1965 to 1985 *the software crisis* - software runs over budget, schedule, causes faults that lead to loss of life. Software quality becomes a key idea

1970s *structured programming* - using if, functions, etc.

1980s *Structured Systems Analysis and Design Methodology (SSADM)*

1985 to 1989 *'no silver bullet' idea* - no single technology or approach will solve the software crisis

1990s *Object-Oriented Programming (OOP)*

1990s *Internet* becomes dominant technology

History of Software Engineering (part 2)

1991 *Rapid Application Development (RAD)* - still common in user-interface design methods

1994 *Dynamic Systems Development Method (DSDM)* - the first agile method

1995 *Scrum*

1999 *eXtreme Programming (XP)*

2000 to today rise of lightweight methods (the focus of this module)

2000s various *agile methods* defined

2001 *The Manifesto for Agile Software Development*

2008 *DevOps* (Development Operations) coined

The Manifesto for Agile Software Development

Signatories state that as developers they 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*

After class:

- Reflect on the Agile Manifesto and analyse what the individual points mean to you
- Read the Wikipedia section on the Agile Manifesto. Reflect how your original analysis is similar and different to that presented on Wikipedia

University of Roehampton London

Lean Software Development

- **Eliminate waste** – do not do work that does not add value to the customer
- **Amplify learning** – usually by short iteration cycles with feedback from the client and the team
- **Decide as late as possible** – wait until all the facts are available before deciding how features are implemented
- **Deliver as fast as possible**
- **Empower the team** – *“find good people and let them do their own job”*
- **Build integrity in** – keep the system simple to update and modify
- **See the whole** – *“Think big, act small, fail fast; learn rapidly”*