|  |
| --- |
| Complete Notes |
| Software Engineering Methods |
| Set08103 |
|  |
| **David Henning** |
|  |

Table of Contents

[Lecture 01: Introduction 3](#_Toc535756710)

[A Software Engineering Method 3](#_Toc535756711)

[Software Engineering 3](#_Toc535756712)

[Method 4](#_Toc535756713)

[History of Software Engineering Approaches 5](#_Toc535756714)

[Agile Manifesto 5](#_Toc535756715)

[Lean Software Development 5](#_Toc535756716)

[DevOps 6](#_Toc535756717)

[Software Development Metrics 6](#_Toc535756718)

[Lecture 2: Scrum / Team Forming 7](#_Toc535756719)

[Scrum 7](#_Toc535756720)

[Scrum Components 7](#_Toc535756721)

[Product Backlog 7](#_Toc535756722)

[Task / Scrum / Kanban Board 7](#_Toc535756723)

[User Stories 7](#_Toc535756724)

[Sprints 7](#_Toc535756725)

[Sprint Planning 8](#_Toc535756726)

[Story Points 8](#_Toc535756727)

[Planning Poker 8](#_Toc535756728)

[Daily Standup 8](#_Toc535756729)

[Sprint Review / Retrospective 9](#_Toc535756730)

[Scrum Roles 9](#_Toc535756731)

[Scrum Master 9](#_Toc535756732)

[Product Owner 9](#_Toc535756733)

[Team Member 9](#_Toc535756734)

# Lecture 01: Introduction

## A Software Engineering Method

### Software Engineering

* The application of engineering to the development of software in s systematic method
* Software Development: The process involved in creating and maintaining software components

Development Processes

* Conceiving: Coming up with an idea
* Specifying: Coming up with requirements to build the software
* Designing: Takes the specification to produce some form of model for the software
* Programming: Using the design and specification to write code
* Documenting: All written documents to do with the software
* Testing: Testing the program to ensure it is up to standard
* Bug fixing: Fixing bugs in production software

Engineering: The creative application of science, mathematical methods, and empirical evidence to the innovation, design, construction, operation and maintenance

Engineering Strands

* Application of Science: - A collection of theories underpinning software which we apply in our development process.
* Mathematical Methods: Any mathematical approach involved in modelling
* Empirical Evidence: To measure something to gain information

### Method

* An approach, which within software engineering can be underpinned by the Software Development Lifecycle
* Initiation – System Concept Development – Planning – Requirement Analysis – Design – Development – Integration and Test – Implementation – Maintenance – Disposition

Waterfall

* Each stage must be completed before moving onto the next change
* Lack of adapting to the changing customer needs
* Requirement Gathering – Analysis – Design – Coding – Testing – Operation

Spiral

* Built on some of the ideas of the waterfall model but provides the ability to adapt due to iterative nature
* Works by iterating through the following four stages until the software is released
* Determine Objectives – Identify / solve risks – Development / testing – Plan next iteration

Agile

* Agile methods build on the iterative approach, but focus on **human-centric** ideas where software is evolved by collaboration between teams and customers.
* Teams are self-organising, and support multiple parts of the development process
* Teams can adapt as requirements evolve with the client and the problems in development are discovered.

## History of Software Engineering Approaches

* **Pre 1965:** Work on defining a discipline, but the term Software Engineering unused.
* **1965:** Various letters to the ACM, lectures, and advertisements mention the term Software Engineering.
* **1965** – **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
* **1980s:** Structured Systems Analysis and Design Methodology (SSADM) - waterfall method of software development.
* **1990s:** Object-Oriented Programming (OOP).
* **1990s**: Internet becomes dominant technology.
* **1991:** Rapid Application Development (RAD)
* **1994**: Dynamic Systems Development Method (DSDM)
* **1995:** Scrum introduced - a common product management method.
* **1999:** eXtreme Programming (XP).
* **2000s:** various agile methods defined.
* **2001:** [The Manifesto for Agile Software Development](http://agilemanifesto.org/).
* **2008:** DevOps (Development Operations) coined.

## Agile Manifesto

* Individuals and Interactions over processes and tools.
* Working Software over comprehensive documentation.
* Customer Collaboration over contract negotiation.
* Responding to Change over following a plan.

## Lean Software Development

* **Eliminate Waste:** Do not do work that does not add value to the customer.
* **Amplify Learning:** Short iteration cycles with feedback from the client and the team.
* **Decide Late:** Wait until all the facts are available before deciding how features are implemented.
* **Deliver Fast:** Agreeing what will be delivered using a pull-based approach.
* **Empower the Team:** Let the team make decisions on the work to be done
* **Build Integrity:** Keep the system simple to update and modify
* **See the Whole**: See the system as a whole and work to ensure the component interactions work.

## DevOps

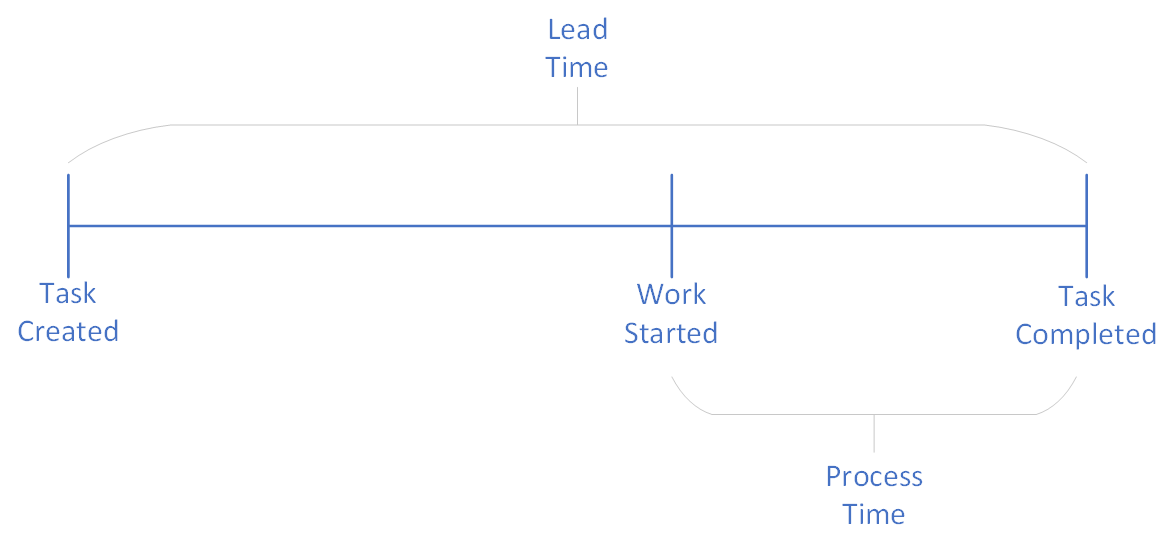
* Developers work closely with the operators to automate and monitor the entire software development lifecycle
* Can be seen as the intersection of development, operations and quality assurance
* **Development:** The software engineer’s role
* **Operations:** The management of systems where software runs
* **Quality Assurance:** Where testing resides

Five Goals:

1. Improved deployment frequency
2. Faster time to market
3. Lower failure rate of new releases
4. Shortened lead time between fixes
5. Faster mean time to recovery

## Software Development Metrics

* Lead Time: The time from when the task is created to when it is completed. Should aim to reduce this time
* Process Time: The time from when work on the task is started until when it is completed. Want work to flow.
* Percentage Complete / Accurate: Helps measure the quality of the work produced. Can do this by asking if the work produced was usable as-is



# Lecture 2: Scrum / Team Forming

## Scrum

* Scrum is a project management and group working philosophy that defines an **Inspect and Adapt** cycle. Constantly inspecting how we work, and then adapting based on that observation
* Scrum's core ethos of change to get teams working towards the goal by rewarding the behaviour of collaboration and getting work done.
* When selecting tasks to work on, the team prioritises by what returns the most value to the customer. This requires constant communication with the customer.
* Work is changed by focusing on the team. The job of management in the organisation is to free the team to increase the flow of work through the system.

## Scrum Components

### Product Backlog

* A list of all currently indentified tasks which are then prioritised by the product owner
* Focus on what is valuable as if everything is a priority, nothing is.

### Task / Scrum / Kanban Board

* Used to keep track of the task
* They flow from the backlog to done as they are completed

### User Stories

* When defining a task it should be part of a User Story.
* This allows the team to understand **why the task is needed.**

### Sprints

* A sprint is an iterations which have a fixed length of time where a set of tasks will be completed.
* The team decide how which tasks they will complete in a week and then analyse how well they did at the end of the week.

### Sprint Planning

* **Independent:** The task can be completed on its own. It should not be dependent on another task.
* **Negotiable:** Until it is being done a task can be changed.
* **Valuable:** It delivers value to the customer, user, or stakeholder.
* **Estimable:** You can size the task.
* **Small:** Task is small enough to estimate and plan. If not, break it down into smaller tasks.
* **Testable:** The task has a test it will need to pass to determine if it is complete. Write the test before you start the task.

### Story Points

* A way of estimating the cost of each task
* Each task is scored by a number

### Planning Poker

An way of estimating the value of a task

1. Select a task.
2. Everyone picks a card from their deck to score the task, but does not reveal it.
3. When everyone is ready, everyone reveals their card at the same time
4. If the cards are within two of each other then sum the cards and divide by number of voters to get the average. That is your estimated cost.
5. If the cards are not within two of each other, then the smallest and largest scorers describe why they gave the points they did. Then a re-vote.
6. Repeat until the group are within two cards of each other, then average.

### Daily Standup

At the same time every day the team meets for up to 15 minutes. Each person answers the following three questions:

1. What did you do since the last stand-up?
2. What are you going to do before the next stand-up?
3. What is getting in your way?

### Sprint Review / Retrospective

The team should ask the following questions:

1. Is there anything we can do differently to speed things up?
2. Can we offload some Backlog items (i.e. give them to other teams)?
3. Can we not do some things (i.e. remove items from the backlog)?

Each person should ask themselves these questions:

1. On a scale from 1 to 5, how do you feel about your role in the company?
2. On the same scale, how do you feel about the company as a whole?
3. Why do you feel that way?
4. What one thing would make you happier in the next Sprint?

## Scrum Roles

### Scrum Master

* The Scrum Master keeps the team working, solves blockages, and keeps the rhythm going.
* Their job is to help the team work as best they can.

### Product Owner

The Product Owner decides what the work should be.

He or she owns the Backlog, what's on it, and most importantly, what order it's in.

### Team Member

Each member should have:

* Autonomy to make decisions on what to do and how to do it.
* Mastery of the what they are doing.
* Purpose, or a reason why they are doing something.