CS4507 Advanced Software Engineering

Adrian O’Riordan

[a.oriordan@cs.ucc.ie](mailto:a.oriordan@cs.ucc.ie)

Office - WGB 180

Tuesday at 10-11 - WGB G08

Wednesday at 12-1 - WGB G18

2 Continuous Assessments - 25%

Winter Written Exam - 75%

# Intros - Lecture 1 - 13/09/2022

Only 10 pages in the first lecture which i think is pretty based

Records lectures

Build on knowledge of software engineering

* Process models for software development
* Object-orientated modelling of software-intensive systems using a software modelling language and a modern modelling tool
* Fundamental software design heuristics and design patterns
* Software maintenance, refactoring & reengineering
* Continuous delivery and DevOps

Labs

Will involve analysis and design using the UML modelling tool Modelio

## Course Contents

Modern Software engineering (1 Lecture)

**Part 1: (4 lectures)**

Software process and Software Lifecycle

* Process models overview: iterative, Spiral, RUP, MDA, Agile
* Process improvement and quality assurance

**Part 2: (10 lectures)**

Object-orientated Software Engineering in UML

* Domain modelling
* Use Cases
* UML Overview and Class and Seuence Diagrams

**Part 3 (7 Lectures)**

Topics in Software Engineering

* Software Tools
* Software Maintenance & re-engineering
* Continuous Delivery and DevOps
* Software refactoring

**Relevant books**

Go pirate these books from page 9 of intro slides if you are a nerd (literally 6 of them)

http://libgen.rs/

## Lecture 1

Only 17 pages

Software Engineering Field - Large Field of Study, with many sub-disciplines

2 definitions on page 3 of “What is Software Engineering”

Better one:

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software (ISO/IEC/IEEE 24765:2010)

All aspects of software production:

* From early stages of system specification through design, implementation, testing, deployment, to the operation and maintenance
* Not just technical aspects of development but also project and team management, tools and platforms, methods, measurement, quality assurance

Software Systems and Scale

* From small embedded systems to mobile apps, to enterprise systems, to **ultra-large-scale** in fields such as telecommunications, transport, finance, healthcare, and defence
* Focus on methods used for medium to large scale systems

History of Software Engineering:

Not gonna cover this cause it was covered last year? Proceeds to cover it for like 5 minutes?

Large Scale Software Development:

Involves a team of people, need to manage process, people, and artefacts

Need to plan - systems take a long time to build

Need to define risks, feasibility and specify requirements

Systems complex enough to require powerful tools and technologies to design, build and test

Need to try to reuse code, designs, and processes

Problems with Large-Scale software development

* Issues with cost timelines and quality
* Are among the most complex manmade systems; has intrinsic, essential properties
* Modest sized programs do not scale well to the development of large complex systems
* Pace of change in computer and software technology drives the demand for new and evolved software products
* Qualified software engineers has not kept pace witht the demand

# Lecture 2 - 14/09/2022

**Good Software in operation**

* Correct
* Acceptable
* Secure
* Reliable - more demanding in embedded systems in cars
* Safe - more demanding in embedded systems in cars
* Efficient
* Usable

**Good Software in change & evolution**

* Adaptive to changing requirements/market
* Rapid
* Testable
* Maintainable
* Extensible
* Scalable
* Reusable

**Good Software in the Process**

* Flexible
* Supports rapid development and delivery
* Disciplined/systematic
* Repeatable
* Quantifiable

**Problems with Software in operation**

* Does not meet requirements
* Not acceptable to users
* Buggy
* inefficient
* insecure
* unreliable
* difficult to test

Etc on the slides

Software Engineering has come a long way in terms of process, technologies, tools, best practice, and education

## Different Approaches to Software Engineering

Development process divides software into distinct phases to organize and improve development, product, and project management

Often uses agile methods

Roles which reflect the responsibilities of the people involved

Phases

* Inception
* Requirement Specification
* Software Architecture & design
* Implementation
* Deployment & Delivery
* Operation & Evolution

Many Process Models

* Waterfall/Linear
* V-mode
* Agile methods - eg scrum, cp, kanban
* Continuous Integration and Delivery - eg DevOps

Plan based Process

* Predictive, document-based, requirements specified in detail, upfront planning

LightWeight/Agile Process

* Flexible,rapid-delivery, incremental planning

# Lecture 3 - 20/09/2022

## Plan-based Process

Can generally be characterised as either plan based or lightweight/agile

Plan-based => Waterfall, V-Model,RUP, Cleanrom

Characteristics: predictive, document-based, requirements specified in detail, upfront planning

Advantages = Greater control and repeatability

Disadvantages can be time consuming, inflexible, and misunderstood

Suitability = very large projects, non-co-located teams, time to market critical

## Lightweight Process

Lightweight/agile => RAD, Scrum, XP, DevOps

Characteristics: adaptive, artefact-orientated, rapid-delivery

Advantages: flexibility allowing for change, more and earlier customer feedback, rapid development

Disadvantage: System may lack overall design/documentation

Suitability: Small to medium-sized projects, time to market critical, co-located teams

## Which Process?

Each company has to develop its own process or tailor an existing process

No ideal software process that applies to all organisations

Each company can develop its own process

### Code and Fix

No method being followed

No planning or phases

Doesn’t scale up

### Waterfall model

Covered last year in CS3500

“Wont go over this” - proceeds to cover this for like 5 minutes

### V-Model

Extension of waterfall model showing explicit relationships between each phase

Not commonly used

Became mandatory for software projects in germany

### Structured methods

Using graphical notations, flow diagrams, structure charts, ER diagrams

### Prototyping

Development of incomplete versions, either throwaway or evolutionary

### Incremental Development Pros and Cons

**Advantages:**

Cost

More rapid deployment

Easier to get feedback

**Cons:**

Costly system architecture

System structure can degrade as new increments are added unless refactoring is done

### Increments

Process diagram on page 18 of the second pdf

### Spiral Model Characteristics

Proposed by Boehm in 1986 based on experience in software for aerospace and defence industries

Risk-based model that makes risk assessment explicit. Risks can be internal or external to project; Risks are explicitly assessed and resolved throughout the process.

No fixed phases

Meta-model or reference model for models, rather than a lifecycle

Big fuck off spiral

Literally no one thinks in spirals, spiral deez nuts ;) woah kinda rude

Sections of the spiral are the different phases

### RUP

Rational Unnified Process

Use-case driven, risk based, architecture-centric

# Lecture 4 - 21/09/2022

## Object orientated methods

RUP cycles

Inception phase

Elaboration phase

Construction phase

Transition phase

Can have multiples of the phases

Not used in web development

## MDA

## Agile Methods

Xtreme programming and Scrum

Common practices on page 38/50 on the slides

Optimized rapid development, and better code

# Lecture 5 - 27/09/2022

## Unsuitable for Agile

Real-time systems

Safety critical systems

Projects with teams at different locations?

Large Systems?

Rapid Application Development

Hybrid Approaches

Scaling

Disciplined Agile Delivery

# Lecture 6 - 28/09/2022

## Mature Software Organization

Managing software development and maintenance process

CMM and CMMI

Capability Maturity Model

CMMI Levels

1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimising

May be required for software projects

PSP Structures

# Lecture 7 - 04/10/2022

## UML and Software Modelling

Software Modelling

Describe and communicate abstractions of the system

Models offer simplified view

Different models and notations may be needed in different phases

Structural models

Describe the elements of the system

Behavioural models

Describe the behaviour of the system over time

UML

Has like 14 different diagrams

# Lecture 8 - 05/10/2022

## 4+1 View Models

1. Logical view
2. Process view
3. Implementation view
4. Physical view

## View Models

Defines a coherent set of views to be used in construction of a system architecture

Process of Object-Orientated Analysis: Domain Modelling

Conceptual Classes first

Case Study

Identify Concepts

Rule out certain concepts

List of Associations

# Lecture 9 - 11/10/2022

Use Cases describe interactions between a role and a system to achieve a goal

* Typically defined in terms of steps but no standard way to define a use case

# Lecture 10 - 12/10/2022

Hteradion Diagrams

Seq Diagram

Commn diagram

# Lecture 11 - 18/10/2022

**Class Notations**

Name

Attributes

Operations

Minus for private

Plus for public

Basic associations between stuff represented by a line

Can give a name and a direction

Aggregation uses a diamond to show association

Composition uses a filled in diamond to imply that a part belongs to it

Dont use double inheritance unless you know your coding language can do that too

# Lecture 12 - 25/10/2022

## Controller

Responsible for handling a system event

Can have multiple

* Object that represents the overall system
* Object that represents a use case scenario

## Polymorphism

## Pure Fabrication

All 9 principles for design heuristics

# Lecture 13 - 26/10/2022

## Software Design Patterns

Way of reusing knowledge about a problem and its solution in a software design

Best Practice

Comes from architecture

Template for expressing design patterns

Pattern classifications

* Creational
* Structural
* Behavioural

### Adapter Pattern

Also known as Wrapper

### Singleton Pattern

Single instance of class is allowed

Will either make a new empty object or just return the single copy if a new one is being called

### Strategy Pattern

Different values at run time

# Lecture 14 - 01/11/2022

## Software Design Patterns

### Observer Pattern

One to many relationship

Observer and subjects

### Iterator Pattern

## Sequence Diagrams

Slight bit of code reference

# Lecture 15 - 02/11/2022

## State Machines

Finite State machine is an abstract machine that can be in on of a finite number of states

UML State machines us a run-to-completion model

Transitions Can have 3 parts

* Event
* Guard
* Effect

## Composite states - Super States

Helps remove duplicates

# Lecture 16 - 15/11/2022

## Refactoring

Superclasses

expensive

## DevOps

# Lecture 17 - 23/11/2022

## Last set of notes

Software Tools

And Environments

Late 1970s they started making tools for other parts of software design and development

UML Modelling tools

Eg Modelio, LucidChart

## Exam Stuff

Software Process and Development Lifecycle

* Introduction to software engineering and profession
* Software process models and process improvement

UML and Object-orientated Software Engineering

* UML Overview and software modelling
* OOD with GRASP, with design patterns

Topics in SE

* State modelling
* Software evolution
* Software refactoring
* DevOps and continuous delivery
* Tools

Format

3 questions each worth 25 marks

Q1: = Problem question in object-orientated analysis and design with UML

Q2: = Software engineering process./lifecycles, process improvement, DevOps, software tools, software evolution and reengineering

Q3: = UML, GRASP, design patterns, state machines

### TIPS

Not examined this year

* Modellio
* Details of Somerville or Larman case studies
* Only need to know the stuff we covered
* Dont need to know the dates when stuff was released