## Server Side Technologies Lecturer: Wesley Gorman

Email: wesley.gorman@ncirl.ie

## Server Side Technologies

### Module Aim

The aim of this module is to understand the concepts of software engineering and apply those concepts to create server side web applications that incorporate a scripting language and a web framework.

### Aim and Objectives

Aim:

* to understand the concepts of software engineering
* Create server side web applications

Objectives:

* Appreciate why software engineering is necessary
* To understand the definition of software engineering
* To understand the differences between software engineering and computer science
* Learn how to apply software engineering techniques
* Use a web framework and be able to critically evaluate to benefits of frameworks

**Learning Outcomes**

On successful completion of this module, learners will be able to:

1. Describe the software process outlining in detail the requirements modelling, design, implementation and testing phases of the process
2. Discuss software engineering principles with reference to the advantages and disadvantages of object oriented software engineering
3. Identify object associations, aggregations and dependencies for a given problem domain
4. Interpret objects within a scripting language library to ascertain their function(s) and integrate such objects within new applications
5. Compare and contrast different approaches to server side development
6. Create a server side web application using scripting languages that incorporates web frameworks
7. Create a secure web application using web frameworks

**Assessment**

## Why "engineer" software?

* Software is everywhere!
* Software is critical to modern economies and society
* Software is Complex!

### Software Crisis – 1968

* Informal software development not working!
* Projects were consistently late and overran budget
* Resulting Software was unreliable, difficult to maintain and performed poorly

### Software Engineering was born

* Techniques and methods to control software complexity were developed
* Since then software is MORE important to society and MORE complex
* Software engineering principles needed more than ever!

### What is Software Engineering?

“Software engineering is an engineering discipline that is concerned with all aspects of software production”

Engineering discipline – build through the application of theories, methods and tools within defined constraints

All aspects of software production – not just technical processes – management, documentation, testing etc. Source: Software Engineering – Ian Sommerville

### Definitions of Software Engineering

“Software Engineering: (SE) A systematic approach to the analysis, design, implementation and maintenance of software.

It often involves the use of CASE tools.

There are various models of the systems life-cycle and many methodologies for the different phases.”

[Wasserman] identifies eight fundamental notions that form the basis for an effective discipline of software engineering:

1. Abstraction - a description of the problem at some level of generalization that allows us to concentrate on the key aspects of the problem without getting mired in the details. (Separating logical from physical).
2. Analysis and Design Methods and Notations - when you work as a team, you must communicate with many other participants in the development process, and therefore need a common notation for communication and documentation. (Methodologies)
3. Prototyping - building a small version of a system, usually with limited functionality, helps the user to identify the key requirements of a system and demonstrates the feasibility of a design or approach; commonly used to design a good user interface. (Prototyping as an important methodology)
4. Software Architecture - the description of a system in terms of a set of architectural units, and a map of how the units relate to one another (Concept of layered models)
5. Software Process - the organization and discipline in the activities of the process of developing software (as well as to the products that result) contribute to the quality of the software and the speed with which it is developed . (Focus on quality)
6. Reuse - taking advantage of the commonalities across applications by reusing items from previous development .(Principle of code reuse)
7. Measurement - quantitative descriptions of improvements to processes, resources, and methods permit us to compare progress across disparate projects and support analysis and decision-making. (Metrics to measure progress)
8. Tools and Integrated Environment - **computer-aided software engineering** (CASE) tools are designed to enhance software development

(CASE) “Software engineers adopt a systematic and organised approach to… produce high-quality software” -Sommerville

## Good Software

* Maintainability
* Meet changing and evolving customer needs
* Dependability
* Reliable, secure and safe
* Efficiency
* Does not make wasteful use of system resources
* Usability (Usable by the type of user for whom it was designed)

## Software Engineering Methods

* Structured approaches to software development which include system models, rules, design advice and process guidance
* Model descriptions : Descriptions of graphical models which should be produced
* Rules: Constraints applied to system models
* Recommendations : Advice on good design practice
* Process guidance: What activities to follow

## Software Engineering Principles

* Separation of concerns
* Modularity
* Abstraction
* Anticipation of change
* Generality
* Incremental development
* Consistency

## The Software Process

* Software specification :Agreed definition of the software to be developed
* Software development : Design and programming of software
* Software validation : Ensuring software satisfies the customer requirements
* Software evolution : Software modified and adapted as per changing customer needs.

## Software Specification



### CASE (Computer-Aided Software Engineering)

Software systems which are intended to provide automated support for software process activities. CASE systems are often used for method support

Upper-CASE : Tools to support the early process activities of requirements and design

Lower-CASE : Tools to support later activities such as programming, debugging and testing

## Software Engineering versus Computer Science

Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software

Computer science theories are currently insufficient to act as a complete underpinning for software engineering

## Summary

* Software engineering is a systematic approach to handle development complexity to produce quality software
* Software engineers adopt the software process to produce quality software
* Wassermans definition of Software Engineering - Abstraction, Analysis and Design Methods and Notations, Prototyping, Software Architecture, Software Process, Reuse, Measurement, Tools and Integrated Environment.
* Computer Science - theory and fundamentals
* Software Engineering - the practicalities of developing and delivering useful software

This is a great lecture!