CO7205 Advanced System Design

Credits: 15  Convenor: Prof. J. L. Fiadeiro  Semester: 1st

Prerequisites: none
Assessment: Coursework: 40%  2 hour exam in January: 60%
Lectures: 24 hours
Surgeries: 8 hours
Laboratories: 8 hours
Private Study: 72.5 hours

Subject Knowledge

Aims Complexity is a recurrent issue in Software Engineering. Since the early days of IT, building complex software systems has been a major challenge and the problems that it raises keep making headlines on the press. In order to address these problems, more and more emphasis is being put into techniques that support the higher levels of system design, namely advanced modelling languages and software architecture. These levels allow us to move away from code and understand how systems are required to operate based on more abstract models. Models reflect an architecture for the software system in the sense that they reflect an organisation based on components that perform relatively simple computations and connectors that coordinate the interactions between the components. This module provides an introduction to the topic of software architecture in general and modelling techniques for service-oriented architecture in particular.

Learning Outcomes At the end of the course, students should be able to: understand the difference between programming and designing applications, and the techniques that support them; understand the basic concepts and role of software architectures, including the separation between computation and coordination concerns; understand the software organisational principles of service-oriented architectures in particular; model systems using a service-oriented modelling language.

Methods Class sessions, tutorials and practical sessions together with course notes, recommended reading, worksheets, printed solutions, and some additional hand-outs.

Assessment Assessed coursework, traditional written examination.

Skills

Aims To teach students abstraction and higher-level modelling skills, with special emphasis on architectural views of systems; to develop in the students the ability to separate concerns during system design.

Learning Outcomes Students will be able to: decompose system requirements according to the principles of service-oriented computing; modularise service applications by identifying the dependencies that services have on several kinds of external parties; model service orchestrations; model abstract semantic interfaces for services; model the protocols that coordinate service interactions; model service-level agreements.

Methods Class and lab sessions together with worksheets.

Explanation of Prerequisites Experience in Programming in Object Oriented Paradigms (Java) as well as general program design skills will be helpful.

Course Description Complexity is a recurrent issue in Software Engineering. Since the early days of IT, building complex software systems has been a major challenge and the problems that it raises keep making headlines on the press. In order to address these problems, more and more emphasis is being put into techniques that support the higher levels of system design, namely advanced modelling languages and software architecture. These levels allow us to move away from code and understand how systems are required to operate based on models. Models reflect an abstract architecture for the software system in the sense that they are organised in terms of components that perform relatively simple computations and connectors that coordinate the interactions between the components. This module provides an introduction to the topic of software architecture in general.
and modelling techniques for service-oriented architecture in particular.

**Detailed Syllabus**

**Software Engineering** Short history and background. Levels of abstraction: requirements, design and implementation. Complexity in software development: programming in-the-small, in-the-large, and in-the-world.

**Software Architectures** Physiological vs social complexity; architectures of usage vs interaction; components versus connectors; architectural styles; nature and role of architectural description languages.

**Modelling for Service-Oriented Architectures** Use Case diagrams for capturing requirements on business activities and services: the roles that different parties can play in a service-oriented system organisation. Conversational protocols and patterns for service-oriented interactions. Orchestration of complex services. Service-level agreements as constraints on service discovery and selection.

**Reading List**


**Resources** Study guide, worksheets, lecture rooms with data projector, computer laboratory access, tutorial rooms with data projector.

**Module Evaluation** Course questionnaires, course review.