CO7205 Advanced System Design

Credits: *15* **Convenor:** *Dr E. Tuosto* **Semester:** *1st*

Prerequisites:	*Essential: Familiarity with distributed programming, message sequence charts or sequence diagrams, state machines.*
Assessment:	*Coursework: 100%*

Lectures:	*24*	*hours*	**Problem Classes:**	*0*	*hours*
Surgeries:	*8*	*hours*	**Private Study:**	*80.5*	*hours*
Laboratories:	*0*	*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 modelling techniques for distributed applications.

Learning Outcomes At the end of the course, students should be able to: demonstrate understanding of the basic concepts and role of software architectures, including the separation between computation and coordination concerns; demonstrate understanding of the relations between models and implementations of distributed applications; apply techniques to support the modelling, testing, and programming of message-based distributed applications.

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

Assessment Assessed classtests.

Skills

Aims To teach students abstraction and higher-level modelling skills, with special emphasis on architectural and coordination aspects 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 message-based distributed applications; modularise applications by identifying the dependencies that interactions have on distributed parties; model distributed choreography; model the protocols that coordinate service interactions.

Methods Class and practical 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. Familiarity with message sequence charts (or sequence diagrams) and with state machines.

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 modelling techniques for distributed applications.

Detailed Syllabus

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

Software Architectures Architectures of usage vs interaction; components versus connectors; nature and role of architectural description languages.

Models of Communication-Centric Software Models for capturing requirements on distributed applications: the roles distributed components. Conversational protocols and patterns for interactions. Choreography of complex services.

Reading List

[A] E. Tuosto, *Choreography for Distributed Applications,* Notes.

[B] M. Shaw and D. Garlan, *Software Architectures: Perspectives on an Emerging Discipline; ISBN: 0131829572,* Prentice Hall, 1996.

[B] D. Brand and P. Zafiropulo, *On communicating finite-state machines,* Journal of the ACM, 30(2):323342, 1983.

[C] Bass, Clements, Kasman, *Software Architecture in Practice; ISBN: 0321154959,* Addison Wesley, 1998.

[C] G. Cécé and A. Finkel, *Verification of programs with half-duplex communication,* Information & Computation, 202(2):166190, 2005.

Resources Study guide, lecture rooms with data projector, tutorial rooms with data projector.

Module Evaluation Course questionnaires, course review.