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.*

Lectures:	*23*	*hours*	**Problem Classes:**	*4*	*hours*
Surgeries:	*4*	*hours*	**Independent Study:**	*72.5*	*hours*
Laboratories:	*0*	*hours*			

Assessment: *Coursework: 100%*

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. The role of models is key in the development of distributed systems. In fact, models that clearly separate "computation" from "communication" have been recently advocated to tame the complexity of this class of systems. This module provides an introduction to modelling techniques for distributed applications.

Learning Outcomes At the end of the course, students should be able to: understand the basic concepts related to coordination of distributed applications, the separation between computation and coordination concerns; understand the relations between models and implementations of distributed applications; apply techniques to support the modelling, programming, and validation of distributed applications.

Methods Class sessions, tutorials 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 coordination aspects of systems and their communication safety; 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 as well as general program design skills will be helpful. Basic familiarity with message sequence charts (or sequence diagrams) and with state machines.

Module 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 local computations) and how they coordinate with each other by some communica-

tion mechanisms. This module provides an introduction to modelling techniques for distributed applications.

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.

Semi-formal approaches Interaction diagrams for distributed systems; distributed realisability of sequence diagrams: problems and solutions.

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

[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.