00
Advanced Software Modelling and Design

Intro

Mirko Viroli
mirko.viroli@unibo.it

C.D.L. Magistrale in Ingegneria e Scienze Informatiche
Alma Mater Studiorum—Università di Bologna, Cesena

a.a. 2023/2024

Teacher and references

Teacher: Prof. Mirko Viroli

- e-mail — `mirko.viroli@unibo.it`
- homepage — `https://www.unibo.it/sitoweb/mirko.viroli`

References: `https://virtuale.unibo.it/course/view.php?id=50583`

- mostly, the place for publiching slides, announces, news

ASMD in the "Software Architect" thread

SAAP - SOFTWARE ARCHITECTURE AND PLATFORMS

Introduces the essential concepts of software architecture and related abilities, including the application of modern architectures to concrete application contexts, including cyber-physical systems and the IoT.

SPE - SOFTWARE PROCESS ENGINEERING

Introduces advanced techniques for organising the software development process, including build automation, continuous integration and delivery, domain-first analysis in a technology-independent fashion, and creation of domain-specific languages.

ASMD - ADVANCED SOFTWARE MODELLING AND DESIGN

Enhances the abilities to construct models of software systems in a variety of contexts (including cyber-physical systems and the IoT), and turn them into concrete designs of reliable and effective systems and applications.

Learning outcomes

The goal of this course is to enhance the abilities of prospective software architects to construct models of software systems in a variety of contexts (including cyber-physical systems and the IoT), and turn them into concrete designs of reliable and effective systems and applications. The student will learn to:

- model and design computational systems featuring non-determinism, stochasticity, large-scaleness, and intelligence;
- rigorously address system requirements adopting techniques of software testing, simulation and verification;
- adopt advanced programming language constructs, techniques and design patterns to address complex software system development.

Philosophy/approach

Statement

- software development and engineering are at a turning point
- complexity of problems is raising
- quality/competence is shrinking (technical debt, lowering skills, AI-generated pollution)

What is needed for a "software architect" that we provide here?

- train to software modelling and design
- train to correct transfer of specification to validation/testing
- span from industry-ready approaches... to research
- span from standard/generic contexts... to emerging scenarios:

$\Rightarrow$ large-scale IoT, Cyber-Physical systems, cloud-edge continuum

Course ingredients

Topics

- high-level patterns of system programming in Java and Scala:
  - component-oriented programming, monads, effects
- systematic validation:
  - model-checking (MC), probabilistic and approximate MC, simulation
- large-scale system modelling:
  - non-determinism, large networks of devices, fluid approximation
- probability and adaptiveness:
  - discrete- and continuous-time markov chains (DTMC and CTMC)
- self-organisation:
  - patterns, macro-programming, aggregate computing
- decision processes and learning:
  - markov decision processes, reinforcement learning (RL), multiagent RL
- program synthesis with AI:
  - LLMs, role in testing, role in program completion

Schedule

Part A: Robust software engineering

1. Pervasive validation for robust software engineering
2. Advanced testing: principles and techniques
3. Code synthesis with LLMs

Part B: High-level specification and programming

4. High-level programming patterns
5. Advanced functional programming

Part C: Modelling with complexity

6. Modelling nondeterministic systems
7. Modelling stochastic systems
8. Stochastic analysis
9. Simulation

Part D: Learning and distribution

10. Reinforcement learning
11. Aggregate computing
12. Aggregate and multiagent reinforcement learning

Labs, and exam

Lab

- each module has a lab activity
- in each lab we propose various tasks, exercising various skills
- students pick those they like most, and start working on them
- freely, students complete some tasks at home
- they work alone, or in small groups (typically, in pairs)
- the teacher provides assistance at need

Exam

- discussion of tasks completed by students, e.g.:
  - one big task, producing a software artifact or a scientific paper
  - various smaller tasks
- discussion of links to other parts of the course