Study programme(s): Computer science

Level: academic master studies

Course title: Advanced functional programming

Lecturer: Zoran D. Budimac

Status: elective

ECTS: 6

Requirements: None

Learning objectives

Introduction to advanced programming techniques in functional and hybrid programming langauges such are Haskell, Erlang, Scala and domain specific embedded languages. The course has two focuses: theoretical and practical, with emphasis to usage of functional programming languages in large-scale projects.

Learning outcomes

Minimal: At the end of a course, the successful student will be able to understand advanced concepts of functional programming languages.

Desirable: At the end of a course, apart from minimal learning outcomes, it is expected that successful student understands benefits and flaws of practical usage of functional programming in large scale projects.

Syllabus

Theoretical instruction:

Introduction to advanced constructs of functional and hybrid programming languages and the means to merge two different paradigms in a single programming language. An overview of at least three programming languages (e.g., Haskell, Erlang, Scala). Monads, functors, automatic program transformations, parallelization, verification, type inference. Advantages of using these languages in big practical projects of industrial strength.

Practical instruction

Work on a big case-study project that is written in one of mentioned languages. Analysis and adding new functionality.

Literature

Recommended

1. O'Sullivan, B., Stewart, D., Goerzen, J., Real World Haskell, O'Reilly, 2008.
2. Martin Odersky, Lex Spoon, and Bill Venners, Programming in Scala, Addison-Wesley, 2016.
3. Cesarini, F., Thompson, S., Erlang Programming, O'Reilly, 2009.

Weekly teaching load				Other:
Lectures: 2	Exercises: 0	Practical Exercises: 2	Student research:	

Teaching methodology

Classic methods of teaching are used for theoretical instruction with usage of video beam. Practical exercises are used to analyze a large case-study project, analyze the needs for extensions and new functionalities, and then implement them.

Grading method (maximal number of points 100)

Pre-exam oblications	**Points**	**Final exam**	**points**
Assignments	**30**	Oral exam	40
Tests	**30**		