SDE exit point program in Budapest

Distributed Service Systems

<table>
<tr><td colspan="2">1. semester</td><td>2. semester</td></tr>
<tr><td colspan="2">Development of Distributed Software (5 ECTS)</td><td rowspan="5">Master's Thesis Project (30 ECTS)

I&E Minor Thesis (6 ECTS)</td></tr>
<tr><td colspan="2">Service-Oriented Integration (5 ECTS)</td></tr>
<tr><td colspan="2">Analysis of Distributed Systems and Process Structure (5 ECTS)</td></tr>
<tr><td colspan="2">I&E: Business Modeling (5 ECTS)</td></tr>
<tr><td rowspan="3">Electives (4 ECTS)</td><td>Large-scale Projects for Analysis and Development of Complex Telecom Software</td></tr>
<tr><td>Design of Distributed Systems</td></tr>
<tr><td>Reverse Engineering of Complex Software Systems via Static Analysis</td></tr>
</table>

Master's Thesis Project

The Master's Thesis project will be carried out within one of the EIT ICT Labs business partners in Budapest.

Exit-point I&E module

The exit-point program contributes 10 ECTS to the I&E module through the Business Modeling course and the I&E Minor Thesis.

Course Descriptions

<u>Development of Distributed Software</u>

Learning Outcomes

The students will be able to

- propose an architecture for a distributed application,
- argue about design decisions,
- select and integrate technologies to support a design, and
- develop the code for a design.

Content

The course presents the main problems to solve during the development of a multi-tier distributed application, discusses technologies that are used in this domain in the software industry, and teaches students to design and write good quality code supported by these technologies.

The lectures provide explanations of principles and technologies, as well as tutorial-like example programs. The labs introduce further technologies, but they mainly focus on supervised programming exercises.

- Multi-tier applications
- RPC-based communication
- Message-oriented middleware
- Component life cycle

- Component discovery
- Object-relational mapping
- Transactions
- Web-services
- Web components

Study Material

- Debu Panda, Reza Rahman, Derek Lane: EJB 3 in Action. Manning, 2007. ISBN 1-933988-34-7
- Mark Cade, Humphrey Sheil: Sun Certified Enterprise Architect for Java EE Study Guide, 2nd Edition, Prentice Hall, 2010. ISBN-13: 978-0-13-148203-6

Examination

Mid-term and end-term programming exercises

Pre-requisites

Java and SQL programming skills

Service-Oriented Integration

Learning Outcomes

The students learn different design and integration techniques. The main options of the SOA elements are analysed and the students will be able to apply the most up-to-date algoritms and tools in the complex information system design.

Content

The aim of the course is to explore the powerful mechanisms to create and analyse complex SOA systems.

Syllabus:

- Typical requirements to form integrating services
- Facilities (laws, organizations, cultural things, user requirements, existing systems, infrastructure, resources, etc.), functional and non-functional requirements.
- Interoperability
- Organizational, semantic, syntactic unification. Standards.
- Service oriented architectures
- Loosely coupled cooperation with web services.
- WS-* standards, WSDL
- Higher level functions, system control, process management. Persistency.
- Secure communication (MQ systems).
- ESB (Enterprise Service Bus)
- Functions, services supporting infrastructure, control services.
- Typical function of the SOA frameworks
- Developing modes and environments
- Model-based SOA development. WS-implementation based on UML-WSDL.
- Creation of formal process description based on rules. BPEL descriptions.
- Consistency testing of processes.
- Managing the SOA projects

- Timescales, iterative development, spiral model. Zachman framework and other management frames. SOA roadmap.
- Examples from the actual problems of the Hungarian e-government

Lab:

The main goals of the lab sessions are to teach the basic knowledge and to give skills to find the proper service-oriented solutions.
A complex piece of software has to be implemented in a team.

Syllabus:

- Analyzing simple WSDLs and SOAP messages
- Writing JAX-WS type web services using code-first methods
- Writing JAX-WS type web services using WSDL-first methods
- Writing simple BPEL services
- Writing REST services and REST clients

Study Material

- Tannenbaum - Van Steen: Distributed Systems: Principles and Paradigms, Prenctice Hall, 2006.
- Bieberstein and others: Executing SOA, IBM Press, 2008
- Nicolai Josuttis: SOA in Practice, O'Reilly, 2007
- Official documents from W3C/Web Of Services section, http://www.w3.org/standrads/webofservices/
- Axis2 framework http://axis.apache.org/axis2/
- CXF framework http://cxf.apache.org/
- Metro framework http://metro.java.net/
- Jersey-REST http://jersey.java.net/

Analysis of Distributed Systems and Process Structure

Learning Outcomes

The goal of the subject is to give an overview for the student about how can we explain the parallel behaviour by algebraic methods and Petri-nets, and how work applications based on that models in practice.

Content

The basic concepts of the course are processes, computational processes, parallelism, operations of processes, compositions of processes and properties of processes (liveness, deadlock-free, etc.). The theory of Petri-nets is explored more partially with many modeling example. The behavioural and structural properties, methods of analysis, famed subclasses and relationships between these subclasses are investigated. We define theorems about liveness, safetyness and reachability and present transformation, which preserve these properties. The course introduces the Petri-boxes, a special class of Petri-nets, which help us to model the program structures (sequences, branches and loops). Some tools for simulation and analysis of Petri-nets are also investigated. The second part of the course introduces the theory of algebraic models through a given example. The properties of the models, the methods of descriptions of processes and the possible compositions are examined. The

denotational, operational and axiomatic semantics of the model is given and the relationships of these different descriptions are investigated.

Teaching methods:

There will be lectures introducing the formal specification and properties of Petri nets and algebraic models and exercises where the students will create concrete examples. There will be also programming exercises where the students can use the learned methods.

Syllabus:

- processes
- Petri-nets
- behaviour and structural properties
- program compositions
- simulation of Petri-net
- analysis of different properties

Study Material

- Murata, T.: Petri Nets, Properties, Analysis and Applications (Proc. of the IEEE. Vol. 77., no. 4, ASpr 1989, 541-580)
- Best, E., Devillers, R., Koutny, M.: Petri Net Algebra (Springer 2001)
- Hennessy M.: Algebraic Theory of Processes (MIT, 1989)
- Hoare, C.A.R.: Communicating Sequential Processes (Prentice-Hall, 1985)

Examination / requirements:

- Some smaller exercises have to be solved during the module. Two written exam from the practice. An oral exam takes place at the end of the module.

Pre-requisites

- Familiarity with propositional and predicate logic and basic (graph) algorithms is assumed.

Design of Distributed Systems

Learning Outcomes

Students will be able to express and verify the properties of the distributed programs using formal methods, apply different ways to create advanced compositions of simple programs, and solutions for interesting and difficult problems in a distributed way.

Content

Dining/drinking philosophers, formal specification of distributed problems, properties of distributed systems, safety and progress properties of distributed programs, verification of safety critical properties, program compositions from components with proven properties, computing the value of an associative function, message channels, pipelined networks programming exercises where the students apply the learned methods in the practice.

Study Material

- Misra, J.: A discipline of multiprogramming: programming theory for distributed applications (Springer, 2001)
- Mani Chandy and Jayadev Misra: Parallel Program Design: A Foundation (Addison-Wesley, Reading, MA, Reading, Mass., 1988)

- Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers (Addison-Wesley 2002)
- Schmidt, D., C. et al.: Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects (Wiley& Sons, 2000)

Examination / requirements:

The final exam of this course consists of:

- Implement and document two parallel algorithms, and give a demo about the programs.
- A written exam with exercises.
- An oral exam from the theory.

Pre-requisites

- Basic programming skills using the programming language C or C++.

Large-scale Projects for Analysis and Development of Complex Telecom Software

Learning Outcomes

Students can master how to analyze, design, implement, test and maintain software through real world projects.

Content

They work in groups in order to learn cooperation and how a team should work in software development process. This particular subject is ideal for practicing the theories learned in the other obligatory subjects that deal with high quality software development.

In this course students will have the opportunity to participate in projects of the industrial partners and the universities, where they can further develop their skills, and build professional relationships. Although this is not technically an internship, it is quite similar, as it is a common practice for the participating universities to work together with industrial partners and develop close ties with them, resulting in an effective and mutually beneficial co-operation.

Syllabus:

- analysis and design by UML diagrams
- project management, scheduling and tracking, version control systems team management, project organization component-based development documentation tools implementation of a student project

Study Material

- R. S. Pressman: Software Engineering A practitioners Approach: European Adaptation, 6th Edition
- Sommerville: Software Engineering (Panem, 2002) S. McConnel: Code Complete, 2nd Edition (Amazon)
- C. Michael Pilato, Ben Collins-Sussman, Brian W. Fitzpatrick: Version Control with Subversion (O'Reilly)

Examination / requirements:

- A complex piece of software has to be implemented in a team.

Reverse Engineering of Complex Software Systems via Static Analysis

Learning Outcomes

Students will be able to apply different static analysis offer for analysis of complex software systems. The students will be familiar with different source code representations and information retrieval techniques from those representations.

Content

Different kinds of static analysis techniques and intermediate source code representations. Programming exercises where the students will use these learned methods.

Syllabus:

- symbol table;
- abstract syntax tree;
- program graph;
- code generation;
- control and data-flow analysis (CFG, DFG);
- dependency analysis (PDG, SDG);
- code, software model re-engineering

Study Material

- Muchnick, S. S. Advanced Compiler Design and Implementation. Morgan Kauffmann Publishers, 1997.
- Shivers, O.: Control-Flow Analysis of Higher-Order Languages. PhD thesis, Carnegie Mellon University, 1991.
- Brian Chess, Jacob West: Secure Programming with Static Analysis. Addison-Wesley.
- Flemming Nielson, Hanne R. Nielson, Chris Hankin: Principles of Program Analysis, Springer. ISBN 978-3540654100, 2000.

Pre-requisites

- Basic programming skills.