FILS

Software Development Method, III/6(2), 3C-1L-1P, E, 5

Prof. Luca Dan Serbanati

Course description	Software development methods are used to contribute to the software process. They prescribe human actions and activities, tools, and intermediate products for developing software systems and claim to improve characteristics of the software (e.g., reliability) or its production (e.g., total cost or schedule predictability). As a course topic Software Development Methods is a subset of the Software Engineering field. Many software development methods have come into practice over the last decades, and new methods are emerging. The new methods are more powerful than the old ones, but many of their concepts, guidelines, or techniques are alike. The course chose the object-oriented method, but aims to put emphasis on what is inherent, persistent in all software development methods. Usually, the software development methods focus on two activities in the software process: *analysis* and *design*. Analysis is concerned with understanding the "real world" we are studying (the problem domain) sufficiently well to allow its modeling. Design is concerned with the mapping of the analysis results in an implementable-by-software form. Analysis and design methodologies promote development of software systems using development visual languages and CASE tools as part of a team's project. The Unified Modeling Language (UML) is currently the industry standard for communicating many of object-oriented development artifacts. In this course UML is the vehicle for learning Object-Oriented Analysis and Design. The last but not the least, software comes out from an analysis of its future environment. This is why the course outlines some techniques of the system analysis field, too.
Prerequisite(s) & Corequisite(s)	“Object Oriented Programming”, “Algorithms and Data Structures”, and “Formal Languages & Compilers” . Knowledge and experience of programming in a high-level imperative language (Java, C++, or C). This course will find its natural achievement in the course “Software Engineering”.
Textbook(s) and web materials	**The Software Process** • R.S.Pressman, Software Engineering: A Practitioner's Approach , 7/e McGraw-Hill, 2003. • L.D.Serbanati, Integrating Tools for Software Development , Yourdon Press Computing Series, Prentice Hall, 1992. **Object Oriented Methods** • G. Booch, J, Rumbaugh, I. Jacobson, The Unified Modeling Language. User Guide , Addison-Wesley, 1999. • M. Fowler, Uml Distilled: A Brief Guide to the Standard Modeling Language , 2/e, Addison-Wesley, 1999. • C. Larman, Applying UML and Patterns-An Introduction to Object-Oriented Analysis and Design and the Unified Process , 3/e, Prentice Hall, 2004. • www.uml.org/

Course objectives	The objective of this course is to explore the methods and techniques of systems development from both a theoretical and applied perspective. Upon successful completion of the course, the student should be able to meet the following specific objectives: • Be able to perform systems and software analysis and design • Have skills to partition a system in logical and physical views according to relevant aspects and then integrate them in a unique artifact • Be able to compare and choose intelligently from among methods, tools, and techniques of systems and software analysis and design • Contribute as a team member in analysis and design projects • To understand how to specify analysis and design in terms of several conceptual modeling tools as UML's diagrams.
Topics covered	• Introduction to Systems Engineering . Systems, information systems, and software systems. Separation of concerns (3h) • Introduction to Software Engineering. The Software Process . Capability Maturity Model. Software life cycle models. The Unified Process. (3h) • Introduction to object-orientation . Concepts. Semantic relationships. Abstraction, encapsulation. Classes and objects. UML notation for classes. Relationships. Generalization vs. inheritance. Polymorphism. Association. Aggregation and composition. Class diagrams (5h) • Business Modeling . Business components: Business Cases, Business Objects, and Business Process. Control and object flows. Activity diagrams. (6h) • Requirements Analysis. Functional and Non-Functional Requirements. Actors. System boundaries. Context diagram. Use case descriptions. Use case extension mechanisms. Identifying actors and use cases (3h) • Sequence diagrams. System operations. Operation contracts. (3h) • Software System Analysis. Structural Modeling . Identification of classes and relationships between them. Identifying attributes, operations and relationships. The Problem Domain Model. (3h) • Behavior Modeling. Inter-objects messages. Interactions. Responsibility and collaborations. CRC cards. Sequence Diagrams. Overview Diagrams. Communication Diagrams. Patterns for Assignment of Responsibilities (6h) • Architectural Design. Software modularity: cohesion and coupling. Subsystems. Subsystem communications. Package diagrams. Architectural Styles (4h) • Design Classes . UI classes. Controllers. Persistence classes. Application Class Diagram (3h) • Modeling objects’ dynamics . States, events, automata. Statechart diagrams. Concurrency in UML. (2h)

• Component-based software development . Component diagrams. (2 h)

• System deployment . Deployment diagrams. (1h)

Laboratory

This is a list of the main laboratory topics:

1. Business Analysis and Modelling
Deliverable: Business Model - Activity Diagrams
2. Software Requirements Analysis
Deliverable: Use Case Model
3. Software Requirements Analysis
Deliverable: Functional Model - System Sequence Diagram & System Operation Contracts
4. Software System Analysis. Domain Modeling
Deliverable: Structural Model - Class Diagram
5. Software System Analysis. Behavior Modeling
Deliverable: Interaction Diagrams
6. From Requirements To Design . Design Classes. Object Dynamicity Modeling
Deliverable: Application Class Diagram and Statechart Diagrams
7. Refining the Models and Architectural Design
Deliverable: Package Diagram

There are about 7 assignments, due two weeks after the student get them.
They put into practice modelling activities common in software development.
Assignments should be prepared for the next class period. Some may be collected for grading; others will be reviewed in class with the aid of available CASE tools.
Attendance at each laboratory is required!

Project

The project to be undertaken may be a team or one person project .
When the project is a team-based project each student is expected to work as a member of your group in this course and cooperate with his/her colleagues.
Each group will be responsible for assigning tasks to its team members, but the instructor must be informed about this assignment.
Each of the seven homework assignments will contribute to fulfil the final project. The project is a requirement for the competition of the course and will be graded individually, thus it is the responsibility of the students to document and give full details on his/her contribution to the project.
The project has to be entirely implemented.

Grading

Grading will be as follows:
- 40% Final exam consists of a written answer to a quiz and a closed-book test consisting in analysis and design of a small application.
- 20% Homework assignments, two tests and personal contribution to laboratory classes. The points per assignment will vary depending importance and effort.
- 40% Project assignments. The points will vary depending the proposed solution, CASE tools usage, documentation quality, and effort.

Professional

The course is a very conceptual class. Its main goal is the development of an

significance	understanding of abstract analysis, design, and modeling techniques and to provide attendees with in-depth coverage of the concepts needed to effectively analyze and design software systems. This class provides skills suitable to some roles in the software process as the business analyst, software analyst, and software architect.