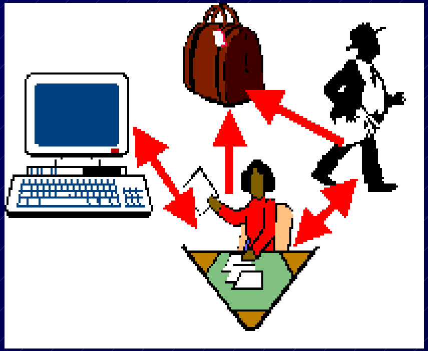
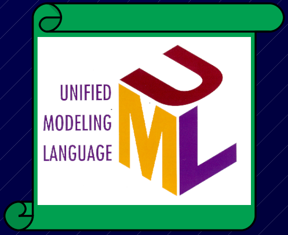
Software Engineering - Introduction

- Course objectives
- Literature
- Course contents
- Timetable
- Marking scheme





Good morning

Albert Ambroziewicz: a.ambroziewicz@iem.pw.edu.pl,

I work at the Warsaw University of Technology, Poland within the EU-founded project ReDSeeDS (Requirements Driven Software Development System) http://www.redseeds.eu

I also lecture at the Warsaw University of Technology, Poland

I have industry experience (J2EE applications development, R&D).

My professional interests include programming (web applications), metamodeling, requirements analysis, software architecture, software engineering processes.

What is Software Engineering?

Software engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

-- IEEE definition

Software Engineering

- Knowledge
- Tools
- Methods for defining software requirements
- Methods for performing software design
- Computer programming
- User interface design

Objectives

The main objective of the course is to teach proper organization of the software engineering (SE) process for building software applications.

This objective is realized through teaching of four SE disciplines:

- user requirements elicitation (UR)
- software requirements formulation (SR)
- architectural design (AD)
- detailed design (DD)

These disciplines are taught as parts of the overall software engineering methodology based on an iterative lifecycle.

SE covers first two of the above disciplines.

Literature - main

ESA Board for Software Standardisation and Control (BSSC). ESA Software Engineering Standards. European Space Agency, February 1991.

Braude, E. Software Engineering. An Object-Oriented Perspective. John Wiley & Sons, 2001.

Leffingwell, D., Widrig, D. Managing Software Requirements. A Unified Approach. Addison-Wesley, 2000.

Fowler M., Scott K., UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional; 2nd edition (August 25, 1999)

Literature - supplementary

Jim Arlow, Ila Neustadt. UML and the Unified Process. Practical Object-Oriented Analysis & Design. Addison-Wesley, 2002.

Martin Fowler, Kendall Scott. UML Distilled, A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2000.

Perdita Stevens, Rob Pooley. Using UML, Software Engineering with Objects and Components. Addison-Wesley, 2000.

Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User Guide. Addison-Wesley, 1999.

Ivar Jacobson, Grady Booch, James Rumbaugh. The Unified Software Development Process. Addison-Wesley, 1999.

James Rumbaugh, Ivar Jacobson, Grady Booch. The Unified Modeling Language Reference Manual. Addison-Wesley, 1998.

Object Management Group. Unified Modeling Language Specification. Version 2.1.2, 2007 (http://www.omg.org/).

Course contents (URS)

Software engineering process and requirements

- Problems in software engineering projects
- Software engineering methodologies
- Core practices of software engineering
- Requirements in the software engineering process

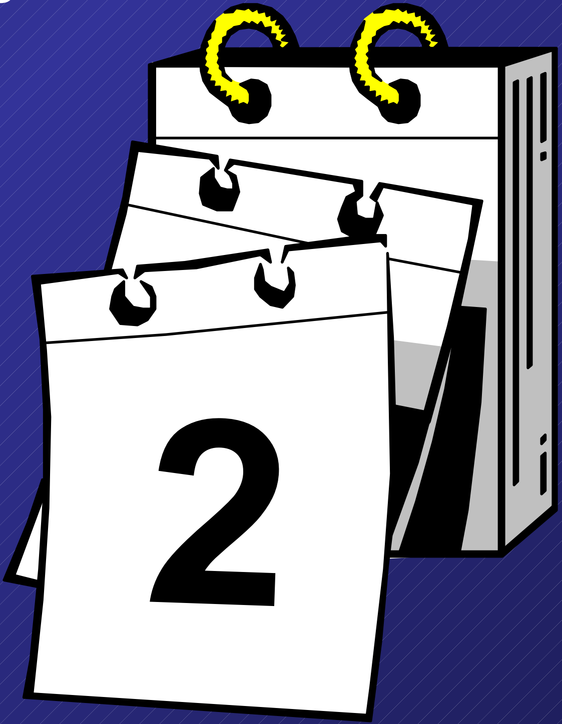
Specifying user requirements

- Modelling of the requirements
- Requirements documentation
- System vision
- Business process description
- Determining the scope of a software system

Course contents (URS)

Use case, vocabulary and business modelling

- Use case model overview and details
- Vocabulary construction
- Vocabulary and the use case model
- Business use cases
- Activities – description of use cases
- Transformation from the business model to the user requirements model



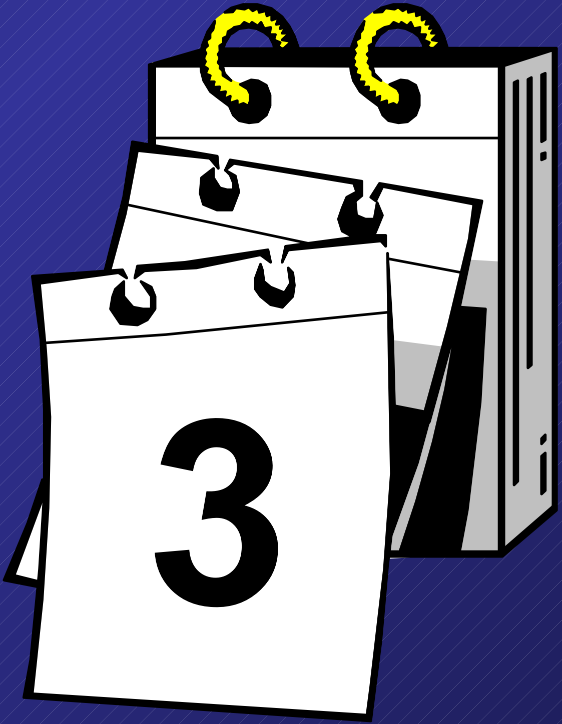
Course contents (SRS)

Software requirements – structure and associated process

- Software requirements in an iterative process
- Criteria for use case prioritization
- Software requirements and acceptance testing
- User documentation vs. requirements

Software requirements modelling (static)

- Class model
- Class modelling on the requirements level
- Classes mapped from vocabulary notions



Course contents (SRS)

Software requirements modelling

- Scenario model
- Details of scenario modelling
- Scenarios and activities
- Scenarios mapped from use cases

Organization and quality of software requirements

- Non-functional requirements types
- Properties of good software requirements specification
- Requirements realization – transformation to design

Timetable (1) – User Requirements

„0” lecture (26.02): Introduction

1st lecture (5.03): Software engineering process and requirements

2nd lecture (12.03): Specifying user requirements

3rd lecture (19.03): Use case, vocabulary and business modelling

Timetable (2) – Software Requirements

4th lecture (26.03): Software requirements – structure and associated process

5th lecture (2.04): Software requirements modelling (static)

6th lecture (9.04): Software requirements modelling (dynamic)

7th lecture (30.04): Organization and quality of software requirements

Marking scheme

The course requires the realization of a team work during the separate exercises in the laboratory (70% of the final mark), as well as a theoretical tests after lectures (30% of the final mark).

70% Practical part of the mark:

- Points earned during the separate exercises in laboratory

30% Theoretical part of the mark:

- Few 15-25 minute tests during lectures (concerning past and current lecture material)
- Individual tasks („homework" assignments) – bonus points

  Students need 50% of points + 1 point from the laboratory classes to pass and 50% + 1 of overall points to pass.

At the end of semester above mark will be confirmed during oral exam.

Course information

- **WikiDyd resources**

  http://wikidyd.iem.pw.edu.pl/index.cgi/SoftEng

- **Consultations**
  - notify by e-mail (a.ambroziewicz@iem.pw.edu.pl)
  - room 224 in the Electrical Engineering building