Software Engineering

Laboratory instruction

2019/2020 edition

Overview:

The aim of the laboratory part of the course is to provide students with a practical ability to develop analytical and design models and to use CASE (Computer Aided Software Engineering) tools. The models are developed using UML (Unified Modeling Language) and **Enterprise Architect** (version **12**) case tool.

Lecturers:

- Anna Bobkowska, PhD, annab@eti.pg.edu.pl, room 647.
- Aleksander Jarzębowicz, PhD, olek@eti.pg.edu.pl, room 648.
- Magdalena Mazur-Milecka, MsC, magmilec@pg.edu.pl, room 106.
- Katarzyna Poniatowska, MSc, katarzynaponiatowska94@gmail.com.
- Olga Springer, MSc, olga.springer@pg.edu.pl, room 648.
- Andrzej Wardziński, PhD, andrzej.wardzinski@eti.pg.edu.pl, room 646.

Additional resources:

The most essential sources of information are lecture slides and additional lab instructions. An interested student can find more information on modeling and UML in the following literature:

1. Fowler M., Scott K., UML Distilled (3rd edition), Addison Wesley, 2003.
2. Booch G., Rumbaugh J, Jacobson I., The Unified Modeling Language User Guide (2nd Edition), Addison-Wesley, 2017.
3. OMG Unified Modeling Language Specification, Version 2.5.
4. Maciaszek L., Requirements Analysis and System Design, Addison Wesley, 2007.
5. McLaughlin B., Pollice G., West D., Head First Object-Oriented Analysis and Design, O'Reilly Media, 2006.

Course schedule overview for the winter semester of 2019/2020:

	Intro-duction	Vision document (6 pts)		Use cases (6 pts + 5 pts)			Object modeling (6 pts + 5 pts)			Dynamic diagrams (6 pts + 5 pts)			Selected design issues (6 pts + 5 pts)					Reserve
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
MON		7-10	14-10	21-10	28-10	4-11	11-11	18-11	25-11	2-12	9-12	16-12	23-12	30-12	6-01	13-01	20-01	27-01
TUE	1-10	8-10	15-10	22-10	29-10	5-11	12-11	19-11	26-11	3-12	10-12	17-12	24-12	31-12	7-01	14-01	21-01	28-01
WED	2-10	9-10	16-10	23-10	30-10	6-11	13-11	20-11	27-11	4-12	11-12	18-12	25-12	1-01	8-01	15-01	22-01	29-01
THU	3-10	10-10	17-10	24-10	31-10	7-11	14-11	21-11	28-11	5-12	12-12	19-12	26-12	2-01	9-01	16-01	23-01	
FRI	4-10	11-10	18-10	25-10	1-11	8-11	15-11	22-11	29-11	6-12	13-12	20-12	27-12	3-01	10-01	17-01	24-01	
	I	C	G	T I	C	G	T I	C	G	T I	C	G		T I	C	C	G	

T	test
I	introduction to an assignment
C	consultation of an assigment
G	grading and feedback

Holidays marked with red color

Days with changed classes schedule marked with blue color

Attention – this is just a generic schedule – the exact dates for all tests, assignment deadlines and other events will be announced by the lecturer assigned to a particular lab group.

Course regulations:

1. „Software Engineering" course ends with a single grade, computed on the basis of lab result (max 50 pts) and exam result (max 50 pts).
2. When attending the lab, students earn points for assignments and tests.
3. To pass the lab, a student has to successfully deliver all assignments and earn at least 25 out of 50 available points.
4. Assignments are done in teams of 3 students (2-member team is also possible). Subjects of particular assignments are given below. Some of assignments are provided with additional instructions available. Exact deadlines and tasks to be done are communicated by lecturers assigned to lab groups.
5. Assignment deadlines should be respected. Delays result in penalty points (-1 for each started week of delay).
6. Assignment is considered as successfully delivered if it is graded for at least 50% of available points, excluding penalty points for delay. Example: an assignment delivered 3 weeks late and graded for 5 points (out of 6) is considered successfully delivered (even though 5 – 3 = 2), but an assignment delivered on time and graded for 2 points requires a correction and resubmission.
7. The date and scope of each test will be announced by the lab lecturer. There are no additional terms or opportunities to write a test again. In case of an unexcused absence, the result of the test is 0 points.
8. Attending the labs is mandatory, the attendance list will be used to verify it. Up to 2 events of unexcused absence have no consequence. In case of 3 and 4 such events, penalty points are used (-3 and -7 respectively). Unexcused absence in 5 lab terms automatically results in failing the course.
9. Sick notes or other absent notes should be delivered to the lab lecturer during the next lab session. If the sick/absence note covers the term of the test, the student is allowed to write the test at another time.
10. When the assignment is graded, the points for it are typically given to all team members. However, when a large disproportion is spotted in contributions of particular team members, the lab lecturer can award them with different numbers of points.
11. When grading a given assignment, the lab lecturer also checks whether his/her remarks to previous assignments were properly addressed. If not – the grade can be lowered.
12. Plagiarism or other cases of academic dishonesty result in immediate failing of the course.
13. Students retaking the lab are supposed to choose a new topic for assignments. The same topic (or a very similar one) is not allowed.
14. The lab is closed at the end of the semester. Afterwards no assignments, absence notes etc. are accepted.

Assignments

The basic descriptions of assignments are given below. More details are given by lab lecturers during introductions to particular assignments.

A1 – Vision document

Aims: Choice of topic, scope definition for further work, identification of customer organization as well as goals, requirements and limitations of the planned IT system.

Tasks:

1. Choose the topic and get its approval from the lab lecturer.
2. Define of the problem scope.
3. Discuss the initial version of the Vision document with the lab lecturer.
4. Prepare the Vision document using the available template (SE_VisionDocument_2019.doc file).
5. Verify of assumptions made in the vision (e.g. scope) through the discussion with the lab lecturer.

Products: Vision document.

A2 – Use Cases

Aims: Identification of system's functional requirements and development of use cases that express them.

Tasks:

1. Identify and describe system actors.
2. Identify functional (system) use cases..
3. Prepare structured descriptions of use cases.
4. Identify relationships between actors and use cases ("comunicate") as well as relationships between different use cases ("include", "extend") and depict them in the use case diagram.

Products: Use case diagram together with use case descriptions included, developed using Enterprise Architect tool.

A3 – Class diagram

Aims: Development of a class diagram covering all essential classes and relationships between them. It usually is an iterative process, as the initial version of the diagram including the set of classes identified at first is being expanded to cover all classes necessary to implement the use cases.

Tasks:

1. Identify all classes necessary for the modelled system.
2. Identify relationships between classes, their type (Association, Generalization etc.) and their properties (e.g. name, multiplicity).
3. For each class identify its attributes and (to the extent it is possible) operations.

4. Describe the classes as well as their attributes/operations when it is necessary for the clarity of the diagram.

Products: Class diagram developed using Enterprise Architect tool.

A4 – Dynamic modeling

Aims: Identification and modeling of the way the objects of classes included in the system interact and collaborate with each other to accomplish the required functionality of the system. Identification and modeling of object's complex behavior.

Tasks:

1. Identify interaction scenarios based on use case scenarios (main and alternative).
2. Identify events (messages), their parameters and sender/receiver objects.
3. Develop sequence, collaboration and activity diagrams.
4. Develop a state diagram for a selected class.

Products: Sequence diagrams (3, for selected use cases), Collaboration diagram (1, for a selected use case), Activity diagram (1, either for a selected business process or for a selected use case), State diagram (1, for a class of the most complex behavior)

A5 – Selected design activities

Aims: Moving from the more generic and technology-independent analytical model to the design model focused on system architecture and use of technologies.

Tasks:

1. Decompose the system into layers and subsystems.
2. Select technological solutions for the system.
3. Define system's physical architecture and prepare deployment diagram.
4. Design user interfaces for the functionality of a selected subsystem.
5. Refine classes and generate code samples for a selected subsystem.
6. Design a database for the system.

Products: A report prepared using the available template (SE_Design_2019.doc) and including the results of the abovementioned tasks.