Course PM

The aim of this course is for students to synthesize and apply the knowledge achieved from previous compulsory courses in the program and learn about and deal with the activities occurring after the initial product release in software development. The purpose of this course is to familiarize the student with situations, activities, and techniques typical for software evolution, such as adding a new feature, refactoring, automating variability or testing, improving performance, re-balancing non-functional requirements, and so on. Furthermore, students learn to plan, implement, and reflect on software evolution scenarios and improvements.

Learning outcomes

After completing the course, the student should be able to:

Knowledge and understanding

- explain the notion of software evolution scenarios as defined by the Architecture Driven Modernization (ADM) task force of the Object Management Group (OMG),
- summarize state of the art in methods and tools for software evolution tasks, such as program comprehension and software refactoring,
- discuss the challenges associated with software evolution,
- explain current research trends in program comprehension, clone detection, and refactoring.

Skills and abilities

- extract a software product's architecture from a given code base and evaluate the quality of the software product,
- implement one software evolution scenario as defined by the OMG, such as platform or language migration,
- implement changes to a software product that lead to an improvement of the product's quality (quality improvement task),
- make use of synergies between different improvements goals for the same product.

Judgement and approach

- detect and judge needs for quality improvement or evolution in an authentic software product,
- plan the use of appropriate methods and techniques for performing a software evolution scenario and a quality improvement task,
- judge needs for improvement of methods and tools to support software evolution,
- plan and evaluate ideas for new or improved tools.

Organization

This course is offered to students from the University of Gothenburg and from Chalmers. The course homepage is located in Canvas.

The course contains project- and problem-based teaching. The course consists of a series of group and individual milestones, and supervision meetings during the semester. Groups and teachers agree on a software evolution scenario to be performed as a project during the course. In addition, lectures and workshops are used to provide the student with methods and knowledge about software evolution.

Teams will meet their supervisors twice a week to discuss progress, reached milestones, next steps, and challenges. If wished or necessary additional meetings with the supervisors can be arranged. Furthermore, rooms for group work are available on Tuesday and Friday mornings (if no lecture is planned).

Components

The course consists of the following components.

Topic Components: Software evolution requires own techniques concerning, e.g. program comprehension, analysis, and migration. In *sessions* throughout the first study period we provide knowledge and practical skills in these areas.

- **Individual Milestones:** By reaching (optional) individual milestones, students can show that they are prepared to make a valuable contribution to the upcoming group work. There will be 3 individual milestones on the topics software comprehension, clone detection, and refactoring.
- **Individual Report:** The report is given in addition to the group report by each individual student. In this report students reflect on their group's proposal for future support for software evolution tasks, such as software comprehension, clone detection and refactoring. Therefore, individual students define metrics for measuring the success of the proposal and select a new open source system in order to demonstrate the proposals strength and weaknesses using their metrics.

Project Components: The project components are structured with milestones. The group will submit a report that clearly outlines the achievements as well as each student's contribution.

- **Group Milestones:** Group milestones allow groups to show continuous progress of their project. To guide the teamwork especially at the starting phase, the first group milestone is predefined targeting comprehension of the software program to be reused. The other 4 group milestones will be planned by the teams themselves in form of sprint planning. Within the supervision sessions together with the supervisor the milestones are planned and assessed. Check on the Canvas course page how to prepare the milestone assessment. The students are asked to prepare a "tour" through the milestone, showing how the milestone was reached. The tour includes demonstrations of the system as well as using code and architectural models to explain progress. Each student should lead and present at least one tour for one of

the 5 milestones. All students are supposed to participate in the discussions to each milestone. Failing to lead at least one of these tours has impact on the grade for the individual contribution.

- **Group Report:** At the end of the course, each team will submit a final team report that consists of two parts. First, the report clearly describes the team's achievements with respect to the quality models and goal that were defined for the project. The teams should describe the original system, the part of the system selected for reuse, the techniques for selecting these parts and for evolving the system towards the customer needs, and the achieved results. In addition, the group report should contain a proposal for future support for software evolution tasks, such as software comprehension, clone detection and refactoring, e.g. an algorithm for clone detection, a concrete idea for a new visualization, or an improvement of an existing software visualization technique.

  Second, the group report should include a documentation of each student's individual contribution to the project. Note, that it is not enough to contribute large amounts of the code within the group in order to gain complete points for the individual contribution. We expect all students in a group to make and document their efforts in order to support underperforming students. In a team with balanced workload all students have the chance to gain full points on the individual contribution. Furthermore, contributing to the code is mandatory in order to pass the individual contribution component.

For details about the schedule see Course Schedule.

A mapping of the components to the learning outcomes can be found in the table below:

Learning Outcomes	Examination				
	Group milestones	Individual optional milestones	Written group report	Documentation of the individual contribution	Individual written final report
Knowledge and understanding					
explain the notion of software evolution scenarios as defined by the Architecture Driven Modernization (ADM) task force of the Object Management Group (OMG),			X		
summarize state of the art in methods and tools for software evolution tasks, such as program comprehension and software refactoring,					X
discuss the challenges associated with software evolution,					X
explain current research trends in program comprehension, clone detection, and refactoring		x			x
Skills and abilities					
extract a software product's architecture from a given code base and evaluate the quality of the software product,	X	(Compr)	X	X	

implement one software evolution scenario as defined by the OMG, such as platform or language migration,	X		X	X	
implement changes to a software product that lead to an improvement of the product's quality (quality improvement task),	X	(R, Cl.)	X	X	
make use of synergies between different improvements goals for the same product,	X		X	X	
Judgement and approach					
detect and judge needs for quality improvement or evolution in an authentic software product,	X	(R, Cl.)	X		
plan the use of appropriate methods and techniques for performing a software evolution scenario and a quality improvement task.	X		X		X
judge needs for improvement of methods and tools to support software evolution,			X		
plan and evaluate ideas for new or improved tools			x		x

Examination (Meetings and Deadlines)

For all dates, please also see the schedule here.

- Group milestones: Bi-weekly sprint planning and evaluation of finished sprint according to previous planning with the supervisor during the supervision meeting. The 5 dates are: 2th of October, 16th of October, 6th of November, 20th of November, and 4th of December 2020
- Individual milestones: There will be 3 individual milestones in the first study period semester. The deadlines will be: 15th of September, 29th of September, and 13th of October 2020
- Fair: 8th of December (the competition will contribute to the grading of the group report component)
- Group report and record of individual contributions: due on 13th of December 2020
- Individual report: due on 20th of December 2020
- Post-project evaluation meeting: 15th of December (will help to assess the individual contribution)

Grading

Grading will be based on individual achievements in the different project activities. Achievements will be assessed continuously during the course and based on the final deliveries. The grading scheme will allow us to give continuous feedback and to do the assessment to a good degree during the project. By collecting data points continuously over the course and providing feedback, we want to achieve a fair assessment, focus of the workload of this course in the term (not in the exam period), and flexibility to account for project specifics in evaluation.

Students can reach up to 30 points:

- Group milestones (mandatory): will contribute up to 5 points to the grade (16% of the maximum points). There will be 5 milestones. For each milestone up to 1 point can be reach (graded in steps of 0, 0.5, or 1 points). To pass this component a minimum of 2 points must be reached.
- Group report (mandatory): will contribute up to 10 points to the grade (33% of the maximum points). To pass this component a minimum of 5 points must be reached.
- Individual Contribution (mandatory): will contribute up to 6 points (20% of the maximum points). To pass this component a minimum of 3 point must be reached.
- Individual report (mandatory): will contribute up to 6 points (20% of the maximum points). To pass this component a minimum of 3 point must be reached.
- Individual milestones (optional): will contribute up to 3 points (10% of the maximum points). For each individual milestone up to 1 point can be reached (graded in steps of 0, 0.5, or 1 points). It is not necessary to reach any of these points in order to pass the course.

To pass the course all four mandatory components (group milestones, group report, individual contribution, and individual report) need to be passed and at least 15 points must be reached (50% of the maximum points).

Thresholds for the grades (hold when all four mandatory components are passed):

GU	**Required points of 30**	**Chalmers**	**Required points of 30**
G (50%)	***15***	***3 (50%)***	***15***
		4 (70%)	***21***
VG (80%)	***24***	***5 (90%)***	***27***

Missed Deadlines and Revisions

Handling of missed deadlines and revisions depends on the grading component.

- Group milestones (mandatory): If a team fails the group milestones component, the project can be extended until January 8th with a milestone assessment on January 8th, 2021.
- Group report (mandatory): If a team fails the group report, a resubmission is possible. The date for the resubmission is February 26th, 2021.
- Individual Contribution (mandatory): If a student fails the individual contribution, the individual work on the project can be extended until January 17th, 2021.
- Individual report (mandatory): The individual report can be resubmitted on three dates: February 26th, 2021, April 13th, 2021, and May 25th, 2021
- Individual milestones (optional): Individual milestones are optional. A resubmission is not possible.