TDA594 / DIT593 Software engineering principles for complex systems

✎ **Edit**

Course-PM

[Click here for the detailed schedule, slides, and resources](#)
(Last Updated: October 18)
[Please read this announcement about Covid-19 safety!](#)

Course is offered by the department of Computer Science and Engineering.

Note: If there are issues with Canvas, course materials are backed-up (with some delay) at **https://greg4cr.github.io/courses/fall21tda594/index.html**

Contact Details

Examiner/Course Responsible
Greg Gay (**ggay@chalmers.se**)

Teaching Assistants
Wardah Mahmood (**wardah@chalmers.se**)
Mazen Mohamad (**mazenm@chalmers.se**)
Mukelabai Mukelabai (**mukelabai.mukelabai@cse.gu.se**)
Shameer Kumar Pradhan (**kumarsh@student.chalmers.se**)

Student Representatives
Please contact **ggay@chalmers.se** if you are interested in volunteering!

Student Office
Contact **student_office.cse@chalmers.se** for questions related to the course administration (e.g., registration, signup, grades in LADOK).

Communication and Course Feedback

We understand that it can be difficult to get answers to your questions, particularly with the speed that things have changed during the pandemic. We recommend the following methods for contacting the teaching staff:
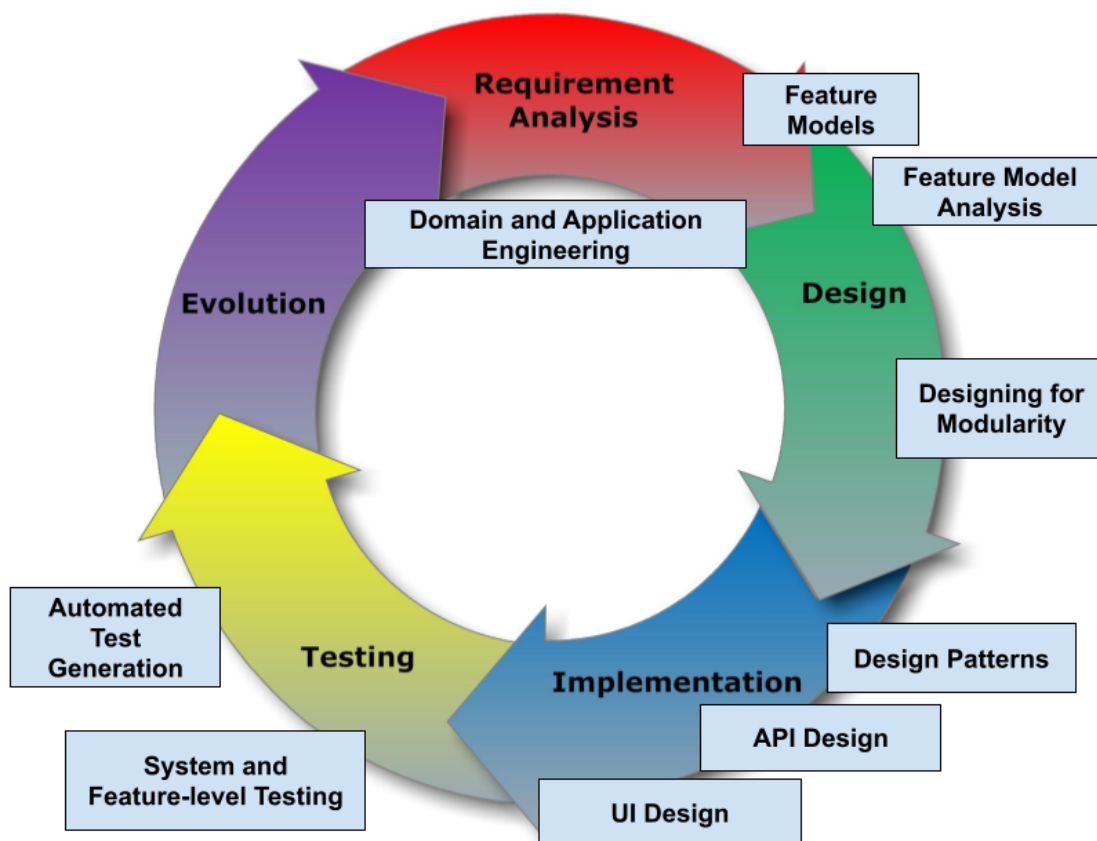
- Any questions related to the lectures or assignments can be asked in the Canvas discussion forum. This forum will be monitored by the teachers and supervisors. You may also e-mail your supervisors directly about assignments and the teachers about lecture content. However, we

recommend posting questions to the forum to help other students who may be having similar issues.

- Private or otherwise sensitive questions may be sent to the supervisors or teachers directly. You are welcome to e-mail Greg Gay (course responsible) directly if the topic is of a highly sensitive nature.
- Course feedback should be presented to the student representatives, who will pass it on to the teachers and supervisors. Please offer feedback to the student representatives first. However, if you have any issues that are time-sensitive or private, you may contact Greg Gay directly.

We welcome course feedback, and are happy to adjust the course to correct issues affecting a majority of the students.

Course purpose

Requirement Analysis

Feature Models

Feature Model Analysis

Domain and Application Engineering

Design

Evolution

Designing for Modularity

Automated Test Generation

Testing

Design Patterns

Implementation

API Design

System and Feature-level Testing

UI Design

Real-world software systems are becoming increasingly complex and pervasive. Consider application domains such as enterprise computing (e.g., data-processing/AI systems), business information systems (e.g., web portals), cyber-physical systems (e.g., automotive software, home automation), systems software (e.g., operating system kernels), or mobile software and software ecosystems (e.g., Android apps). All of these domains boast software systems of unprecedented complexity, many of which are long-living and exist in many different variants. Dedicated planning, modeling, design, realization, and advanced analysis are needed to deliver these systems in a safe, robust, and secure manner.

The goal of this course is to present advanced techniques aimed at creating complex systems with many interacting features and variable configurations. We will examine the design, implementation,

and verification of complex systems, focusing on managing complexity at different stages of the system development life cycle. We will also examine how the development process must be organized to deliver such systems. This class is particularly focused on how to create systems that can change and evolve over time - systems that can be customized for new customers or new market environments.

Much of this course will focus on software product lines (SPLs) - highly configurable families of systems built around a set of common, modularized features. SPLs are developed from a common set of core assets in a planned way, integrating assets into a shared architecture. By developing large software systems in this way, companies can produce a set of products more economically, since the development effort put into the shared assets does not need to be duplicated. SPLs can also help companies to better address product customization tasks to meet specific needs of individual customers. This allows sustaining a high rate of product innovation, while keeping guaranteed levels of overall system performance and quality.

SPLs are seen in many domains, and are highly challenging to develop as they have technical, process, organization, and business-related aspects. We focus on them, as they are designed to be highly evolvable - designed both to reuse components and for components to be reused. Almost all modern systems are designed for one of these two goals - a web app with a REST API is designed to be reused, while a complex web app constructed using third-party libraries is designed to reuse assets. SPLs are designed to deliver reusable components and to reuse those components. Design, implementation, and verification techniques for SPLs are applicable to all evolvable, reuse-driven systems. We will also examine other forms of complex systems and provide insight into how they are delivered.

Schedule

[TimeEdit](#)

[Detailed Schedule and Resources](#)

Course literature

There is no **mandatory** course literature.

Many materials have been used in the creation of slides and assignments. The following **optional** textbooks were used to prepare lecture slides:

- Apel, S., Batory, D., Kästner, C., & Saake, G.. **Feature-oriented software product lines.** (available online via SpringerLink, use a Chalmers IP address to get it for free).
- Van der Linden, F. J., Schmid, K., & Rommes, E.. **Software product lines in action: the best industrial practice in product line engineering.** (available online via SpringerLink, use a Chalmers IP address to get it for free).
- Freeman, E., Robson, E., Bates, B., & Sierra, K.. **Head First Design Patterns.** (available online via **https://www.vlereader.com/Reader?ean=9780596800741** , use a Chalmers IP address to get it for free)

- Pezze, M., Young, M. **Software Testing and Analysis: Process, Principles, and Techniques.** (available for free from **https://ix.cs.uoregon.edu/~michal/book/free.php**)

These books were referenced during the creation of some of the slides, and provide theoretical material relevant to many of the course concepts. In cases where the books are older, we make use of theoretical content that is still relevant to modern systems, while discarding outdated theory and updating technological concepts. Students are recommended to obtain a free copy for use when studying the material in the course.

Additional optional literature will be provided for individual topics.

Course design

The teaching consists of lectures, group and individual assignments, as well as supervision in connection to the group assignments. The course emphasizes problem-based learning. Basic concepts are presented in the lecture, applied in in-class exercises, and then extended in the context of integrated, graded assignments. A series of five group assignments will be developed in teams of 6-7 students. There will also be an individual assignment at the end of the course designed to assess critical thinking about the course content.

Language of instruction: English

Teaching and Learning Activities

Lectures: There will generally be two lectures per week (Tuesday and Thursday from 13:15 - 15:00) (see Schedule). Attendance in lectures is highly recommended, but not mandatory. As the group project and individual assignment will correspond to material taught in the lectures, it is recommended that you attend. Note that any material covered in lectures can be referred to in the project and individual assignment (i.e., not just material written in the slides).

Lectures will be conducted both in-person and will be streamed online. Due to GDPR requirements, we will not record lectures.

Supervision: Formal supervision sessions will be held weekly with your assigned supervisor. These sessions give students the chance to apply class concepts, as well as to see demonstrations. Supervisors will be present to answer questions, as will (at times) the teachers. Attendance in supervision sessions is mandatory. Your assigned supervisor will work with you to schedule a time slot.

Forum: Any questions related to the lecture, exercises, or assignment can be asked in the Canvas discussion forum. If you have any questions or doubts regarding the course material, this is a good place to express them.

Changes made since the last occasion

While the core focus of the course has been preserved, many aspects of the course have been redesigned:

- Lectures topics have been revised. New lectures have been added, while others have been removed. We have created new slides for topics taught be a different teacher previously.
- Assignment content (group and individual) has been substantially revised.

Learning objectives and syllabus

Learning objectives:

1. Knowledge and understanding

- Explain the challenges of engineering complex software systems
- Explain industrial practice and examples of complex software systems engineering
- Explain processes and concepts for engineering complex and variant-rich software systems
- Explain business-, architecture-, process-, and organization-related aspects of engineering complex software systems

2. Skills and abilities

- Model a software system from different perspectives (e.g., using feature models, UML diagrams, architecture description languages)
- Engineer a variant-rich software system (e.g., variant-rich software system, software product line, software ecosystem)
- Analyze and re-engineer a complex software system
- Use and reason about modularization techniques
- Use modern component or service frameworks

3. Judgement and approach

- Analyze existing software systems and discuss potentials for improvement or re-engineering
- Reason about characteristics software modularity concepts
- Recognize in which situations which principles for handling of complex software systems are appropriate
- Read and analyze scientific literature

Syllabus: **Chalmers** (TDA 594), **University of Gothenburg** (DIT 593)

Examination Form

Sub-Courses

1. Project (Group Assignments) (projekt), 6.0 higher education credits
   Grading scale: 3-5 and Fail (U)
2. Written Assignments (Inlämningsuppgifter), 1.5 higher education credits
   Grading scale: 3-5 and Fail (U)

Assessment

The course is examined by an individual written assignment and a project (a series of semi-linked assignments) carried out in groups of 6-7 students.

The project is examined on the basis of solutions to compulsory problems handed in during the course and on the basis of individual contribution to the group work. Each student will be responsible for contributing to each assignment. The project will consist of approximately five assignments. Each assignment is equally weighted. Specific requirements for the assignments will be provided on Canvas. The assignments will correspond to the order of topics in the lectures, covering case studies, domain analysis, modelling, implementation, and testing of complex SPLs.

To give a common SPL platform, we will focus several of the group assignments on Robocode. Robocode is a programming game, where the goal is to develop a robot battle tank to battle against other tanks in Java or .NET. The robot battles are running in real-time and on-screen. A focus in the project will be on re-engineering variants of Robocode bots, which are highly interesting systems, covering different architectural styles and levels of intelligence (for instance, incorporating AI to learn the best strategy to succeed in the Robocode simulator).

For information about Robocode, see its home page (**https://robocode.sourceforge.io/**) and the RoboWiki (**https://robowiki.net/wiki/Main_Page**).

Students are required to complete written self- and peer-assessment forms during the course which will be part of the assessment of the student's individual contribution to the project. The instructor may adjust the individual grades of a student depending on this evaluation.

The written assignment portion of the course will consist of an individual assignment at the end of the course, designed to assess the student's understanding of the core topics of the course. This assignment will be based on the lecture content, and will be similar in scope to a hall exam. However, it will be administered as an online, take-home assignment. The timing and form of this assignment will be further elaborated on soon.

Grading Scales

The two sub-courses (written assignments and project) are graded individually, both of which comprising the grading scale: 3, 4, 5, and Fail (U).

The individual assignment will be graded on a scale of 1-100. This can be translated to a final grade (3-5, U) as follows:

- 84 - 100: 5
- 67 - 83: 4
- 51 - 66: 3
- 0 - 50: U

Other assignments will directly receive a grade (3-5, U) based on assignment-specific rubrics, which will be provided along with the assignment.

The final grade of the course is calculated as a weighted average of the grades for the project and written assignments.

All group and individual assignments must be submitted. The final grade for the project part of the course will be an average of the grade for each of the group assignments. If the average grade

across the group assignments is passing, the students will receive a passing grade for the project part of the course. Failing one of the group assignments does not necessarily mean there will be a failing grade for the project part of the course.

Note that although the project grade is given for a group assignment, this grade is assigned individually to students, and may be adjusted depending on the results of peer evaluation forms. Thus, not all students in the same group are guaranteed to get the same grade.

Late Assignments

Up to One day late: - 20% reduction of final mark or a reduction of one grade level (e.g., 5 -> 4)

Up to Two days late: - 40% reduction of final mark or a reduction of as additional grade level (e.g., 5 -> 3)

Two or more days late: 0% on assignment

Failing Assignments

If the final average grade of all group assignments is a failing grade, all group assignments must be redone and resubmitted. The redone group assignments are handed in again at a date after course completion. Redone group assignments can be done in groups or individually.

Assignment Re-submission

If a group or individual assignment is failed, students have up to two chances to resubmit an improved version. Re-submissions of group or individual assignments will only be accepted until one month after the end of the course. The re-submission must fulfill the original requirements and (for fairness reasons) provide a detailed changelog with an explanation of how/why the first attempt was not sufficient or did not work, and how/why the re-submission works.

If a student, who has failed the same examined component twice, wishes to change examiner before the next examination, a written application shall be sent to the department responsible for the course and shall be granted unless there are special reasons to the contrary (Chapter 6, Section 22 of Higher Education Ordinance).

In cases where a course has been discontinued or has undergone major changes, the student shall normally be guaranteed at least three examination occasions (including the ordinary examination) during a period of at least one year from the last time the course was given.

Additional Information

This section contains some general rules that will be enforced during this course. Please review these guidelines carefully. Violations of conduct guidelines will be taken seriously and will lead to disciplinary action.

Integrity and Ethics

The homework and programs you submit for this class must be entirely your own. If this policy is not absolutely clear, then please contact me. Any other collaboration of any type on any assignment is

not permitted. It is also your responsibility to protect your work from unauthorized access. Any violation of this policy will result - at minimum - in a failing grade on the assignment. Further infractions will result in a failing grade in the course and further disciplinary action. More information on plagiarism will be provided on the Canvas page. We recommend referring to this material for more information.

Classroom Climate

All students are expected to behave as scholars at a leading institute of technology. This includes arriving on time, not talking during lecture (unless addressing the instructor), and not causing disruption in the classroom chat (try to restrict chat to asking and answering questions). Disruptive students will be warned and potentially dismissed from lectures.

Special Needs

It is university policy to provide, on a flexible and individual basis, reasonable accommodations to students that have disabilities that may affect their ability to participate in course activities or to meet course requirements. Students with disabilities are encouraged to contact their instructor early in the semester to discuss their individual needs for accommodations.

Diversity

Someday you will graduate, and in the real world, you will have to work with a wide variety of people. Now is the time to abandon preconceived prejudices about others. Students in this class are expected to respectfully work with all other students, regardless of gender, race, sexuality, religion, or any other protected criteria. There is a zero-tolerance policy for any student that discriminates against other students for any reason.

Course summary:

Date	Details	Due
Thu, 4 Nov 2021	📝 **Assignment 0 - Team Selection**	due by 23:59