2020/2021

Software Design

Code: 102759
ECTS Credits: 6

Degree	Type	Year	Semester
2502441 Computer Engineering	OB	3	1
2502441 Computer Engineering	OT	4	1

The proposed teaching and assessment methodology that appear in the guide may be subject to changes as a result of the restrictions to face-to-face class attendance imposed by the health authorities.

Contact

Name: Joan Serrat Gual

Email: Joan.Serrat@uab.cat

Use of Languages

Principal working language:	catalan (cat)
Some groups entirely in English:	No
Some groups entirely in Catalan:	Yes
Some groups entirely in Spanish:	No

Other comments on languages

La majoria dels materials (transparències, llistes de problemes, enunciat practica, exercicis) són en anglès.

Teachers

David Fernández Mota

Idoia Ruiz López

Lluis Gomez Bigorda

Prerequisites

There are no official prerequisites, but the student must master the notation of UML class diagrams, the object orientation concepts and their programming in some language, preferably Java.

Objectives and Contextualisation

Once the purpose of a software has been determined through the analysis of requirements, software developers design a solution. This goes from the architecture of its components (classes, modules ...) to implementation in some programming language. This subject covers mainly two key aspects of the design: object-oriented design patterns and user interface design. In addition to these, it deals with other subjects such as the design by contract of classes and methods and the style of coding.

The learning method is based on a project (the practicum) or put another way, 'learn to do': all concepts dealt with in theory are necessary in order to design and implement a software application, starting with a statement that plays the role of requirements document. It is about designing and implementing (but not the analysis of requirements, testing, version control or quality assurance, which belong to other courses) of an application.

Competences

Computer Engineering

- Capacity to design, develop, evaluate and ensure the accessibility, ergonomics, usability and security of computer systems, services and applications, as well as of the information that they manage.
- Have the capacity to conceive network technology based systems, applications and services, including Internet, Web, e-commerce, multimedia, interactive services and mobile computers.
- Have the capacity to conceive, develop and maintain computer systems, services and applications employing the methods of software engineering as an instrument to ensure quality.
- Have the capacity to identify and analyse problems and design, develop, implement, verify and document software solutions on the basis of suitable knowledge of current theories, models and techniques.
- Have the right personal attitude.
- Work in teams.

Learning Outcomes

1. Accept and respect the role of the various team members, and its different levels of dependence.
2. Conceive network technology based systems, applications and services, including Internet, Web, e-commerce, multimedia, interactive services and mobile computers.
3. Design a user interface based on customer specifications .
4. Design and document software solutions.
5. Develop curiosity and creativity.
6. Identify the paradigm that best adapts to a design problem in a specific computer software system taking into account the accessibility, ergonomic, usability and security requirements.
7. Identify, manage and resolve conflicts.
8. Know and apply the specific software components for communications based computational problems.
9. Maintain a proactive and dynamic attitude with regard to the development of ones own professional career, personal growth and continuing education. Be able to cope with adversities.
10. Specify the clients needs in a software specification document.
11. Work cooperatively.

Content

1. Principles of object orientation
2. Design patterns: creational, structural, behavioral
3. Design by contract
4. Style of coding and comments. Logging
5. Design of the user interface
- usability
- user research
- process: prototype, user testing
6. Programming interfaces

Methodology

As stated in the objectives, the subject follows a learning methodology based on the development of a project. Its concrete implementation with regard to the use of class hours is as follows:

Theory classes In them, the teacher introduces the theoretical contents and gives useful references (books, articles, web pages) so that the student can then continue their learning, as well as develop the software application proposed. The slides used in these classes should not be taken as a source of study, but rather that they can be used to show the teacher in a more agile way. Instead, you need to study the recommended books, and articles, open access book chapters and others that we will indicate.

Problem classes In them, students present the answers to a list of questions, which they will have previously solved. There will be so much design problems (class diagrams, code, user interface prototypes, etc.) and questions about articles that will, of course, have been read and analyzed, and that will complement the contents of the theory classes or require the your study

Practical sessions. The students will have formed at the beginning of the course small groups (3 per group). In the practical classes students come to 1) show their work to the instructor and pose questions, and 2) given that the practice will be divided into milestones, the weeks that correspond to a milestone, it is evaluated The corresponding part in situ, according to a scale that will be included in the practicum handout. Therefore, the practical sessions are *not* for doing the practicum.

Following last instructions from the Escola d'Enginyeria, teaching of theory, problems and practicum will be "online".

All the lecture notes, slides and more, plus the detailed schedule of the course are in Campus Virtual. Most of it is in English.

Activities

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
practicum sessions	12	0.48	1, 2, 8, 4, 3, 7, 11
problems sessions	12	0.48	2, 8, 4, 3
theory lectures	26	1.04	2, 8, 4, 3, 6
Type: Autonomous			
individual study	29	1.16	8, 4, 3, 10, 6
preparation of practicum	52	2.08	1, 2, 8, 4, 3, 6, 7, 11
solving problems	12	0.48	2, 8, 4, 3, 10, 6

Assessment

The course follows a continued assessment system. This consists of a group assessment part and another individual assessment. The note of the first (PR) is obtained by making the marks of the practical goals (M1, M2, M3). The second (EX) of the average of two tests written during the quarter (EXP1, EXP2), or a retake exam (EXR1, EXR2). The suffixes 1 and 2 refer to the 2 parts of the courset, before and after the first exam EXP1.

The algorithm for calculating the final mark (FM) is as follows:

```
// EXP1, EXP2 partial examinations
// EXR1, EXR2 retake exam
// M1, M2, M3 marks for the 3 practicum milestones
// FM final mark

EX = (max (EXP1, EXR1) + max (EXP2, EXR2)) / 2.

if (M1> 0) and (M2> 0) and (M3> 0) {
PR = (M1 + M2 + M3) / 3.
} else {
PR = 0.
}

if (EX> = 5.) and (PR> = 5.) {
FM = (EX + PR) / 2.
} else {// EX <5. or PR <5.
FM = min (EX, PR)
}
```

The practicum milestones are evaluations equivalent to the written examinations. If a student does not show up, the mark is zero.

The final grade will be 'non-evaluable' if no exam or milestone has been delivered.

There is no retake for the milestones. Depending on the year, we can propose exercises and problems that add points to EXP1, EXP2. These points are maintained for the corresponding retake exams.

One thing that we want to ensure through the different evaluations is that each member of the group has really contributed to the practicum in the same way as the others. Therefore, if, as a result of the questions the instructor makes during the milestones or the written tests about the practicum, we have evidence that this does not happen, then the milestone will be assessed independently for each group member: we will find out which part the current milestone has been made by each member and we will evaluate it individually. In addition, we can split the group from then on.

The dates of continuous assessment and work delivery will be published on the 'campus virtual' of the course web and may be subject to changes in the programming for reasons of adaptation to possible incidents; This will be announced through campus virtual, the mechanism for exchanging information between instructors and students.

For each written exam, a place, date and time of revision will be indicated in which the student will be able to review the activity with the teacher. If the student does not submit to this review, this activity will not be reviewed later.

We do not keep marks for the practicum, exercises or exams from previous years. Students that retake the course are not treated differently that fresh students.

The MH notes will be awarded discretionally to students with a final grade superior to 9.0 and taking into account all kinds of work done (exams, practice and exercises).

Independently of other disciplinary measures considered appropriate, the irregularities committed by the student that can lead to a variation in the rating of an evaluation act will be qualified with a zero. Therefore, copying, plagiarizing,cheating, etc. in any of the assessment activities it will imply fail with a zero mark. Assessment activities qualified in this way and by this procedure will not be recoverable. If it is necessary to pass any of these assessment activities to pass the subject, this subject will be suspended directly, without opportunity to recover it in the same course, with a zero note.

Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
group assessment	50%	4	0.16	2, 8, 5, 4, 3, 10, 6, 9
individual assessment	50%	3	0.12	1, 2, 8, 5, 4, 3, 10, 6, 7, 9, 11

Bibliography

Part 1

- Design patterns: elements of reusable object oriented software. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Addison Wesley, 1994.
- Design patterns explained simply. Alexander Shvets. https://sourcemaking.com/design-patterns-ebook.
- Head first design patterns. E. Freeman, E. Freeman, K. Sierra, B. Bates. O'Reilly, 2004.
- The elements of Java style. Al Vermeulen, Scott W. Ambler, Greg Bumgardner et al. Cambridge University Press, 2000.
- Clean code: a handbook of agile software craftmanship. R.C. Martin. Prentice Hall, 2008.

Part 2

- The user experience team of one. A research and design survival guide. Leah Buley. Rosenfeld, 2013.
- Understanding your users. A practical guide to user research methods. Kathy Baxter, Catherine Courage, Kelly Caine. Morgan Kaufmann, segona edició 2015.
- Killer UX design. Jodie Moule. Sitepoint, 2012.
- Validating product ideas. Tomer Sharon. Rosenfeld, 2016.
- Rocket surgery made easy: the do-it-yourself guide to finding and fixing usability problems. S. Krug. New Riders, 2010.