ACADEMIC PROGRAM

SOFTWARE ENGINEERING

B.F.A. IN **COMPUTER SCIENCE**

MODALITY: ON CAMPUS

ACADEMIC YEAR: 2022-2023

Name of the course:	**Software Engineering**
Degree :	Computer Science
Location:	Centro Universitario de Tecnología y Arte Digital
Area:	Software Engineering
Year:	4º
Teaching period:	1
Type:	OB
ECTS credits:	6
Teaching modality:	On campus
Language:	English
Lecturer / Email	-
Web page:	http://www.u-tad.com/

SUBJECT DESCRIPTION

Area description

This subject establishes the knowledge and techniques necessary for the correct specification, design and implementation of software projects based on good engineering practices and methodologies.

Subject description

The Software Engineering subject is the one in which students learn to plan and develop software projects, going from conceiving this task from the point of view of the programmer to contemplating it from the perspective of the technical project director. Specifically, the student learns to plan and manage software development projects using iterative methodologies, to form work teams to carry out software development projects and to describe software architectures and designs using graphic languages.

COMPETENCIES AND LEARNING OUTCOMES

Competencies

BASIC AND GENERAL COMPETENCIES

CB1: That students have demonstrated possession and understanding of knowledge in an area of study that builds on the foundation of general secondary education, and is usually at a level that, while relying on advanced textbooks, also includes some aspects that involve knowledge from the cutting edge of their field of study.

CB2: That students know how to apply their knowledge to their work or vocation in a professional manner and possess the competencies that are usually demonstrated through the elaboration and defense of arguments and problem solving within their area of study.

CB3: That students have the ability to gather and interpret relevant data (usually within their area of study) to make judgments that include a reflection on relevant social, scientific or ethical issues.

CB4: Students should be able to transmit information, ideas, problems and solutions to both specialized and non-specialized audiences.

CB5: That students have developed those learning skills necessary to undertake further studies with a high degree of autonomy.

CG1 - Ability to understand, plan and solve problems through the development of IT solutions.

GC2 - Development of IT solutions respectful of the environment, social duties and natural resources, in addition to complying with legislation and ethics.

CG3 - Knowledge of the scientific fundamentals applicable to the resolution of computer science problems.

CG6 - Integration, as a software engineer, in multidisciplinary work environments demonstrating teamwork skills, versatility, flexibility, creativity and respect for the work of colleagues from other areas.

CG9 - Ability to learn, modify and produce new computer technologies.

CG10 - Use of creative techniques for the realization of computer projects.

SPECIFIC COMPETENCES

CE10 - Ability to manage a code versioning manager and generate the documentation side of an application automatically.

SC18 - Ability to design the architecture of an object-oriented computer application using the most appropriate design patterns and integrating them into the complete architecture.

SC19 - Ability to conceive, design through graphic languages and implement a computer application using different development methodologies, from the conception of the product to its final development through the definition of its phases and iterations.

CE20 - Ability to test the operation and functionality of a computer application, elaborating test plans and using test-oriented design and programming techniques.

CE22 - Knowledge of the techniques and implications of computer application maintenance including those that use reverse engineering principles to understand and modify a software whose structure is unknown.

Learning outcomes

Upon completion of the degree, the graduate will be able to:

- To know formal specification languages

- To be able to identify and apply design patterns in problem solving

- To use refactoring techniques

- To understand the software life cycle

- To understand and apply waterfall methodologies in software development

- To understand and apply Scrum in software development

CONTENTS

Software life cycle

Agile development methodologies

Iterative development methodologies

SUBJECT SYLLABUS

TOPIC 0. Presentation of the subject

TOPIC 1. Introduction to Software Engineering

1.1. What is software engineering?

1.2. The field of action of Software Engineering

1.3. Risk Management and Software Engineering

1.4. Value orientation and software as a service

TOPIC 2. Requirements management and software design

2.1. Need for requirements management

2.2. Analisys of requirements

23. Non-functional requirements

2.4. Use cases

2.5. Class diagrams

2.6. Sequence and communication diagrams

2.7. Inception and Design Thinking

2.8. Epics and US

2.9. Prioritization and estimation

2.10. Stakeholder management

TOPIC 3. Software development models and methodologies

3.1. Introduction to the software development process

3.2. Software development models

3.3. Methodologies

3.4. Heavy UDP/PUD methodologies

3.5. Light methodologies. Agile

3.6. Scrum

3.7. Kanban

3.8. XP and other ways to apply agility

3.9. Scaling

TRAINING ACTIVITIES AND TEACHING METHODOLOGIES

TRAINING ACTIVITIES

LEARNING ACTIVITIES	Total hours	Hours of presence
Theoretical / Expository classes	25,00	25,00
Practical classes	29,00	29,00
Tutorials	4,00	2,00
Independent study and autonomous work of the student	50,00	0,00
Elaboration of work (group or individual)	32,00	0,00
Evaluation Activities	10,00	10,00
TOTAL	150	66

Teaching methodologies

Expository method or master lesson

Case learning

Learning based on problem solving

Cooperative or collaborative learning

inquiry learning

Flipped classroom methodology

Gamification

Just in time Teaching (JITT) or classroom on time

Expository method or master lesson

Case method

Learning based on problem solving

Cooperative or collaborative learning

inquiry learning

Flipped classroom methodology

Gamification

TEMPORAL DEVELOPMENT

UNIDADES DIDÁCTICAS / TEMAS PERÍODO TEMPORAL

Temas 0 y 1. Introducción a la Ingeniería del Software Semanas 1-3

Traducir por voz

251 / 5.000

Resultados de traducción

Resultado de traducción

DIDACTIC UNITS / TOPICS TIME PERIOD

Topics 0 and 1. Introduction to Software Engineering Weeks 1-3

Topic 2. Requirements management and software design Weeks 4-7

Topic 3. Software development models and methodologies Weeks 8-11

EVALUATION SYSTEM

ASSESSMENT SYSTEM	MINIMUM SCORE RESPECT TO THE FINAL ASSESSMENT (%)	MAXIMUM SCORE RESPECT TO THE FINAL ASSESSMENT (%)
Assessment of participation in class, exercises or projects of the course	0	30
Assessment of assignments, projects, reports, memos	30	80
Objective test	10	60

GRADING CRITERIA

ASSESSMENT SYSTEM	ORDINARY EVALUATION	EXTRAORDINARY EVALUATION
Assessment of participation in class, exercises or projects of the course	10	10
Assessment of assignments, projects, reports, memos	60	60
Objective test	30	30

General comments on the evaluations/assessments

In the case of the project, 25% is attributed to the practical implementation of a prototype, and 75% to the rest (memory, design, exhibition,...).

General considerations about the evaluation:

• The final project (with its intermediate deliveries) will account for 60% of the final grade. It is necessary to pass this section with a 5 to pass the subject, and a 4 to release it in an extraordinary call.

• Work delivered after the deadline will not be evaluated.

• The final exam will be worth 30%. It is necessary to pass the exam with a 5 to pass the subject, and a 4 to be released in an extraordinary session.

• The subject can only be passed with an approved exam and final project.

• In the event that a student has not managed to achieve a 5 in the project and exam, but has at least a 4 in both, he or she will be assigned special additional work to be determined that covers the most deficient aspects of the previous deliveries.

• Those students who fail the final project in the Ordinary Call will have the possibility of repeating it in the Extraordinary Call.

• Any writing that the student presents (problems, exams, comments on the programs, etc.) must be well presented, correctly written (with commas, periods and full stops in their proper place) and without spelling mistakes. The grade of the writing may drop up to 20% otherwise, since a university student is required to have maximum quality in their written expression.

• Exam and project grades are not saved between successive academic years.

• It is not possible to obtain Honor Registration (MH) in the Extraordinary Call.

• The use of notes or calculators of any kind is not allowed in the exams, for which the student must refer to the teacher's specific instructions on this topic.

• The ENTIRE subject will be failed if it is discovered that a student has copied from another (both will be failed) or has copied from a book or from the Internet. In addition, the university will open disciplinary files against both students, which may even lead to their expulsion.

• At least 80% attendance is required to pass the subject.

• "Active participation" does not mean coming to class or "winning" group exercises. Voluntary resolution of exercises and presentations is valued. Likewise, as part of this section, the presentation of ideas, participation in debates, presentation of proposals or additional exercises and, in general, everything that demonstrates involvement in the subject, and not mere passive attendance, will be valued.

LIST OF REFERENCES (BOOKS, PUBLICATIONS, WEBSITES):

Basic bibliography

• Vliet, H. (2007); "SOFTWARE ENGINEERING: PRINCIPLES AND PRACTICES"; Publisher: Wiley.

• Pressman, R. S. (2010); "Software engineering: A practical approach" 7th Ed.: McGraw-Hill

Recommended bibliography

• Fowler, M. and Scott, K. (1999); "UML Distilled: A Brief Guide To The Standard Object Modeling Language"; Publisher: Addison Wesley;

• Beck, K. (1999); "eXtreme Programming explained"; Addison Wesley

REQUIRED MATERIALS, SOFTWARE AND TOOLS

Type of classroom
Theory classroom

Board and projection system

Materials:
Personal computer.

Notebook or tablet for taking notes

Software:
-