Software verification techniques

Academic Year: (2023 / 2024) **Review date: 09/02/2024 13:22:40**

Department assigned to the subject: Computer Science and Engineering Department

Coordinating teacher: GENOVA FUSTER, GONZALO

Type: Compulsory ECTS Credits : 6.0

Year : 3 Semester : 2

REQUIREMENTS (SUBJECTS THAT ARE ASSUMED TO BE KNOWN)

Programming (1st year / 1st semester)
Programming Techniques (1st year / 2nd semester)
Algorithms and Data Structures (2nd year / 2nd semester)

LEARNING OUTCOMES

CB1. 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. Students are able to apply their knowledge to their work or vocation in a professional manner and possess the competences usually demonstrated through the development and defence of arguments and problem solving within their field of study.
CB3. Students have the ability to gather and interpret relevant data (usually within their field of study) in order to make judgements which include reflection on relevant social, scientific or ethical issues.
CB4. Students should be able to communicate information, ideas, problems and solutions to both specialist and non-specialist audiences.
CB5. Students will have developed the learning skills necessary to undertake further study with a high degree of autonomy.
CG1. Students are able to demonstrate knowledge and understanding of concepts in mathematics, statistics and computation and to apply them to solve problems in science and engineering with an ability for analysis and synthesis.
CG3. Students can solve computationally with the help of the most advanced computing tools mathematical models coming from applications in science, engineering, economy and other social sciences.
CG4. Students are able to show that they can analyze and interpret, with help of computer science, the solutions obtained from problems associated to real world mathematical models, discriminating the most relevant behaviours for each application.
CG6. Students can search and use bibliographic resources, in physical or digital support, as they are needed to state and solve mathematically and computationally applied problems arising in new or unknown environments or with insufficient information.
CE17. Students know how to apply software verification techniques to determine if a software component fulfills its specifications, and that they are able to detect faults in those components.

RA2. Through sustained and well prepared argument and procedures, students will be able to apply their knowledge, their understanding and the capabilities to resolve problems in complex specialized professional and work areas requiring the use of creative and innovative ideas.
RA3. Students must have the capacity to gather and interpret data and information on which they base their conclusions, including where relevant and necessary, reflections on matters of a social, scientific, and ethical nature in their field of study.
RA5. Students must know how to communication with all types of audiences (specialized or not) their knowledge, methodology, ideas, problems and solutions in the area of their field of study in a clear and precise way.
RA6. Students must be capable of identifying their own education and training needs in their field of study and the work or professional environment and organize their own learning with a high degree of autonomy in all types of contexts (structured or not).

OBJECTIVES

- To devise, design, build and verify programs that can be executed in a computer knowing the impact of different alternatives in software performance and maintainability.
- To know software verification strategies and techniques, and to be able to define tests for a software component within different software development processes.

DESCRIPTION OF CONTENTS: PROGRAMME

1.- Fundamentals of software verification.
2.- Testing throughout the software lifecycle development models.
3.- Structured based techniques.
4.- Analytic techniques.
5.- Code and design verification techniques.
6.- Agile testing methods

LEARNING ACTIVITIES AND METHODOLOGY

THEORETICAL-PRACTICAL CLASSES. [44 hours with 100% classroom instruction, 1.67 ECTS] Knowledge and concepts students must acquire. Students receive course notes and will have basic reference texts to facilitate following the classes and carrying out follow up work. Students partake in exercises to resolve practical problems and participate in workshops and evaluation tests, all geared towards acquiring the necessary capabilities.

TUTORING SESSIONS. [4 hours of tutoring with 100% on-site attendance, 0.15 ECTS] Individualized attendance (individual tutoring) or in-group (group tutoring) for students with a teacher.

STUDENT INDIVIDUAL WORK OR GROUP WORK [98 hours with 0 % on-site, 3.72 ECTS]

WORKSHOPS AND LABORATORY SESSIONS [8 hours with 100% on site, 0.3 ECTS]

FINAL EXAM. [4 hours with 100% on site, 0.15 ECTS] Global assessment of knowledge, skills and capacities acquired throughout the course.

METHODOLOGIES

THEORY CLASS. Classroom presentations by the teacher with IT and audiovisual support in which the subject's main concepts are developed, while providing material and bibliography to complement student learning.

PRACTICAL CLASS. Resolution of practical cases and problem, posed by the teacher, and carried out individually or in a group.

TUTORING SESSIONS. Individualized attendance (individual tutoring sessions) or in-group (group tutoring sessions) for students with a teacher as tutor.

LABORATORY PRACTICAL SESSIONS. Applied/experimental learning/teaching in workshops and laboratories under the tutor's supervision.

ASSESSMENT SYSTEM

% end-of-term-examination/test:	30
% of continuous assessment (assigments, laboratory, practicals…):	70

SE1 - FINAL EXAM. [30 %]
Global assessment of knowledge, skills and capacities acquired throughout the course.

SE2 - CONTINUOUS EVALUATION. [70 %]
Assesses papers, projects, class presentations, debates, exercises, internships and workshops throughout the course.

BASIC BIBLIOGRAPHY

- Black, R. Agile Testing Foundations, BCS Learning & Development Ltd: Swindon UK, 2017

- Gregory, J. and Crispin, L. More Agile Testing, Pearson Education: Boston MA, 2015

- Myers, G. The Art of Software Testing, (3e), John Wiley & Sons: New York NY, 2011