School of Computer Science

Contact

School of Computer Science

Campus Universitario s/n

02071 Albacete

España (SPAIN)

Web: www.esiiab.uclm.es

Phone: +34 967 599 200, ext. 2461

Fax: +34 967 599 296

Ph.D. Víctor López-Jaquero

ESII Mobility Coordinator

E-mail: VictorManuel.Lopez@uclm.es

About University of Castilla-La Mancha @ Albacete

The University of Castilla-La Mancha, Spain, is ready to face the challenges in the road ahead. The campus of Albacete is tied to the city's urban development and its consolidation in trade and industry. Albacete is located in the southeast, connected by train and motorway to Madrid and Valencia, and by plane to Barcelona. The region offers a wide range of spectacular contrasts in the land known as the native home of Don Quixote.

About the School of Computer Science

Our Faculty started its activity in the academic year 1985/86. In the academic year 2005/06, the European Higher Education Area (EHEA) was adapted with the deployment of an academic calendar in accordance with the rest of Europe, as well as the implementation of a curriculum based on ECTS credits and new teaching methodologies. Our experience for more than 30 years has been very beneficial for the implementation in the 2010/11 academic year of the current **Degree in Computer Science**.









One of the main pillars of our Faculty is our young and experienced teaching staff. Almost ninety percent have a PhD Thesis, mainly in Computer Science, but also in other fields such as Maths and Business. The quality of the research carried out our professors and researchers helps to maintain our teaching in the avant-garde of the profession, preparing our students for the new challenges of the Information and Communication Technology.

It is worth noting that, since 2016 our Degree has obtained the Euro-Inf accreditation issued by the European Quality Assurance Network for Informatics Education (EQANIE). This accreditation is internationally recognized and facilitates both academic and professional mobility, as it allows the identification of high-quality studies in Europe and beyond its borders. The Euro-Inf Framework Standards defines the learning outcomes as well as the criteria to be met by all programmes statements for accreditation.

About our Degree in Computer Science

Our Degree consists of 4 academic years with 60 ECTS each, 240 in total, including the following 4 ACM-IEEE profiles:

- Computer Engineering
- Computer Science
- Information Technology
- Software Engineering

Foreign students coming to Albacete will be able to take the courses in computing depicted in the following table. All of them are **taught in the English language**, in both autumn and spring semesters, as part of our Degree in Computer Science adapted to the European Space for Higher Education:

Semester	**Code**	**Courses**	**Year**	**ECTS credits**
"Autumn" semester	42300	Calculus and Numerical Methods	1	6
	42301	Physics for Computer Science Engineering	1	6
	42302	Programming Fundamentals I	1	6
	42312	Data Structures	2	6
	42314	Software Engineering I	2	6
	42311	Computer Organization	2	6
	42313	Operating Systems I	2	6
	42323	Computer Architecture	3	6
	42324	Software Engineering II	3	6
	42320	Human-Computer Interaction I	3	6
	42321	Intelligent Systems	3	6
	42367	Web and Services Engineering	4	6
"Spring" semester	42306	Programming Fundamentals II	1	6
	42308	Computer Networks I	1	6
	42309	Information Systems	1	6
	42315	Statistics	2	6
	42316	Programming Methodology	2	6
	42317	Concurrent and Real Time Programming	2	6
	42361	Autonomous Robotics	4	6
	42377	Graphic Design and Animation	4	6
	42378	Artificial Intelligence in Videogames	4	6
	42379	Videogames and Virtual Reality	4	6

Both	42399	Professional Training (*)	4	12
	42358	Project Dissertation	4	12

Moreover, in the following table the available **English-Friendly subjects** are listed. This means that the teaching is in Spanish but the foreign students are provided with student's guide, materials, tutorials and exams in English, at the request of the student.

Semester	**Code**	**Courses**	**Year**	**ECTS credits**
"Autumn" semester	42304	Principles of Business Management	1	6
	42349	Language Processors	4	6
	42356	Multimedia	4	6
	310608	Audit and Security Management (Master)	5	6
"Spring" semester	42319	Databases	2	6
	42342	Theory of Automata & Computation	3	6
	42327	Software Design	3	6

In the following sections, a description of these courses is provided in order to describe its curricula and compentences.

English Subjects
Autumn Semester

Calculus and Numerical Methods

Objective

A computer science engineer needs some mathematical tools to comprehend the proper techniques which are useful for his professional life. Our course provides some of them.

Another important point is that mathematics helps students to develop the abstraction capacity, the scientific rigour and the analysis and synthesis skills that are helpful for future engineers.

In this subject, we include some mathematical background useful for other subjects, such as Physics for Computer Science, Statistics and Programming methodology.

Contents

Unit 1. Numbers, sequences and series

1.1 Numbers. Different sets of numbers and their properties. Principle of mathematical induction.

1.2 Sequences of real numbers.

1.3 Series. An introduction.

Unit 2. Differential calculus.

2.1 Basic concepts: functions, limits and continuity.

2.2 Differentiation. Definition of derivative. Tangent line to the graph of a function. Basic properties. The chain rule. Derivative of the inverse of a function.

2.3 Applications of the derivative. Maximum and minimum points. Increasing and decreasing functions. Rolle and the mean value theorems. L'Hôpital's rule. Convexity and concavity.

2.4 Approximation by polynomial functions. Taylor polynomial and Lagrange remainder theorem.

2.5 Approximating the solution of an equation: bisection, Newton and fix points methods.

2.6 Polynomial interpolation.

Unit 3. Integral calculus.

3.1 Riemann integral. Definite integral and its properties. Fundamental theorem of calculus.

3.2 Indefinite integrals. Applications of integrals.

3.3 Improper integrals. Convergence. Different types of improper integrals.

3.4 Approximating a definite integral: trapezoid and Simpson rules.

Teaching dispositions

In the course the following methods of instruction are used: lectures, practical sessions and teamwork projects. Lectures are supported by problem sheets and tutorial hours, practical sessions allow students to use appropriate software and teamwork projects also give students the opportunity to develop their communication skills.

Evaluation

Assessment Criteria in the regular exam session. A minimum mark of 4 out of 10 in each progress test is compulsory to pass the subject. Below this minimum, the student will have to take the corresponding parts in the extra exam session.

Readings and other material

- Apostol, T.M., *Calculus (2nd Ed, Vol.1)*, Wiley India Pvt. Limited, 2007

- Burden, R.L. and Faires, J.D., *Numerical Analysis*, Thomson Learning, 2003
- Mathews, J.H., *Numerical Methods Using Matlab*, Pearson/Prentice Hall, 2007

Physics for Computer Science Engineering

Objective

In this course we present the basic concepts on Scalar and Vector Fields and the main ideas about the Electromagnetism and Electric Circuits that a computer science student needs to understand how a computer works today. In addition, we present some interesting topics that will be covered in this course: Lightning, Pacemakers, Electric Shock Treatment, Electrocardiograms, Metal Detectors, Magnetic Levitation, Superconductivity and any others topics.

Contents

Unit 1. Physical quantities

Unit 2. Error calculus

Unit 3. Vector analysis

Unit 4. Electric field

Unit 5. Electric potential

Unit 6. Capacitors and dielectrics

Unit 7. Direct current

Unit 8. Magnetic interaction

Unit 9. Sources of magnetic fields

Unit 10. Electromagnetic induction

Unit 11. Magnetic properties of matter

Unit 12. Alternating current

Unit 13. Electrical network theory

Teaching dispositions

In this course we will use as main methods of instruction: lectures, seminars, problem resolution and lab practices. Physics are basically an observational science; we observe phenomena that occur in nature and we try to find the rules, patterns, principles and the concepts that control these phenomena. Understanding the patterns, the rules and concepts requires a high level of creative and critical thinking. Then in this course we want you to experience these levels of thinking and develop a high degree of conceptual understanding. Learning to solve problems with a conceptual understanding is essential. Then we aim to teach you not only the general principles but how to apply them in specific real situations.

Evaluation

There will be 4 exams during regular lecture hours and a final exam. Students who do not pass the written examination will be offered a second opportunity in September.

Readings and other material

- Giancoli, D. C. *Physics for Scientists and Engineers for Modern Physics*. Vol. 2. Pearson Ed., 2008
- Cutnell, J.D. and Johnson, K. W. *Introduction to Physics*, Wiley, 2013

Programming Fundamentals I

Objective

Programming Fundamentals I is the first course of the subject of Programming. Thus, it is a cornerstone to provide the foundations for all the subsequent programming courses. Although this course is an introduction to programming in C, it is possible to apply the acquired knowledge to many other programming languages.

Contents

Unit 1. Introduction to Programming

Unit 2. Representing simple data in memory

Unit 3. Data input/output

Unit 4. Control statements

Unit 5. Subprograms

Unit 6. Vectors and Matrices

Unit 7. User defined datatypes

Unit 8. Data input/output: Files

Teaching dispositions

This course will use different methods of instruction, namely lectures, exercises in class, practical sessions in the lab and teamwork for the development of a final project. Students will have slides available in English for each of the units. The exercises in class will serve to consolidate the knowledge acquired during the lessons, while the laboratory sessions will be used to carry out specific practical exercises. Finally, students will develop a final project in group in which they will apply the different techniques learnt during the course.

Evaluation

There will be 3 exams during regular lecture hours, which will be divided in a written test and a lab test (corresponding to the 80% of the final mark), and a final project (15% of the final mark). The remaining 5% will correspond to class participation.

Students not passing the tests will have to take the corresponding part (written or lab) in the final exams.

Readings and other material

- Schildt, H., C: The Complete Reference, 4th Ed. McGraw-Hill, 2000
- Gottfried, B.S., Programming with C, McGraw-Hill, 1996

Data Structures

Objective

Data Structure belongs to the Programming block and it is included into the common module of informatics within de degree programme. The subject can be seen as the natural continuation of Programming Foundations II, but here we study new data structures under several points of view which include, apart from its use, how they are formally defined and implemented, allowing too the possibility for being incorporated as a library into a programming environment.

The subject is also strongly connected with Programming Methodology, where several algorithmic patterns admit an immediate application on the new data structures (for instance, sorting linear ADTs, greedy/dynamic traversing of trees, optimal path searching on graphs, etc) as well as with Declarative Programming (speciality on Computation), where it is mandatory to have acquired some basic notions on lists, recursion, etc., in order to reinforce some data structures after introducing new expressive resources like higher-order, infinite data structures, and so on.

During the rest of the degree (and even furthermore, on their professional careers), students will be concerned with the implementation and manipulation of software applications using complex data structures. Modern languages usually provide some of them by default (lists, queues, etc.) but they must be studied in detail for being correctly used. Besides, other more intricate data structures (e.g. trees, graphs), which are not directly available on typical programming environments, must be designed and incorporated into such tools. Furthermore, the use of data structures, from the point of view of their design and analysis, provide a good level of abstraction and programming skills (e.g. recursion, modularity), very useful in most tasks related to the development of software applications.

Contents

Unit 1. Introduction.

- 1.1. Presentation and preliminary concepts.
- 1.2. Data structures and efficiency.

Unit 2. Formal description of ADTs.

- 2.1. Formal specification with Haskell.
- 2.2. Example: sets.
- 2.3. Implementing an ADT.

Unit 3. Linear ADTs.

- 3.1. Lists. Specification and examples.
- 3.2. Implementing lists.
- 3.3. Stacks and Queues. Specification and implementation.

Unit 4. Non linear ADTs. Trees.

- 4.1. General trees. Specification.
- 4.2. Binary trees. Specification and implementation.
- 4.3. Search trees. Implementation.
- 4.4. Balance and efficiency. AVL trees.

Unit 5. Graphs.

- 5.1. Introduction.
- 5.2. Specification of undirected graphs.
- 5.3. Static implementation of graphs.

Teaching dispositions

In this subject we are concerned with both theoretical and practical issues, that is, formal specification as well as implementation/application of data structures, which are mainly instructed via lectures in the classroom and practical sessions in the laboratory, respectively. Students acquire additional skills by performing small presentations.

Evaluation

There is not a final term exam. Each student's final grade in the regular assessment will be based on the results from her/his work developed throughout the course (tests + submitted tasks + participation). In order to achieve a pass grade in this subject, the total score of the sum (weighted sum according to the previous table) of all the assignments to be assessed cannot be less than 50% of the possible maximum score.

Readings and other material

- Ehrig, H., Mahr, B., *Fundamentals of Algebraic Specifications (vol. 2)*, Springer-Verlag, 1990
- Goodrich, Michael T., *Data structures and algorithms in Java*, Wiley & Sons, 2011
- Lafore, R., *Data Structures and algorithms in Java*, Sams Publishing, 2002
- Dale, N., and Walker, H.M., *Abstract Data Types: Specifications, Implementations and Applications*, D. C. Heath and Company, 1996
- Heffelfinger, D.R., *Java EE 7 Development with NetBeans 8*, Packt Publishing open source, 2015
- Hutton, G, *Programming in Haskell, 2nd. Edition*, Cambridge University Press, 2016

Software Engineering I

Objective

This subject provides a general view of the discipline called Software Engineering and the role it plays within the context of computer science engineering. In order to do so, the subject includes the basic concepts and the main features and techniques to be applied during the stages of software development, that is, requirements, analysis, design and implementation, taking into account the Object-Oriented Paradigm.

Students will acquire the basic skills and knowledge needed to work as a software engineer. With the foundations acquired in this subject, students will be able to deepen in more specific software engineering-related subjects offered throughout the degree.

Contents

Unit 1. Introduction to Software Engineering

- 1.1. Background and definitions
- 1.2. Fundamentals

Unit 2. Requirements Engineering

- 2.1. Introduction
- 2.2. Requirement. Definition, types and features
- 2.3. Requirements Engineering Process Models
- 2.4. Requirements Elicitation Stage
- 2.5. Requirements Analysis Stage
- 2.6. Requirements Validation and Verification Stage

Unit 3. Requirements modelling with UML 2.0. Use Case Diagrams

- 3.1. Introduction
- 3.2. Use Cases Specification
- 3.3. Graphical Notation
- 3.4. Relationships within a Use Case Diagram
- 3.5. Examples and practical cases

Unit 4. Object Oriented Analysis with UML 2.0

- 4.1. Introduction to Software Modelling with UML 2.0
- 4.2. Domain Class Diagrams
- 4.3. Analysis Class Diagrams
- 4.4. Activity Diagrams

Unit 5. Object Oriented Design with UML 2.0

- 5.1. State Diagrams
- 5.2. Sequence Diagrams
- 5.3. Communication Diagrams
- 5.4. Timing Diagrams

5.5. Component Diagrams

5.6. Deployment Diagrams

Unit 6. Object Oriented Software Development. Unified Process (RUP)

6.1. Introduction

6.2. Basic Principles of RUP

6.3. RUP Stages

Teaching dispositions

This subject is taught using different teaching methods such as lectures, seminars, working groups for problem solving, and practice sessions where students must tackle the development of a project applying the knowledge acquired in the theoretical sessions.

Evaluation

To pass the subject, it is compulsory to pass both the written exam corresponding to the theoretical Units and the practice sessions performed in the lab. Besides, it is required a minimum mark of 4 out of 10 in each one of them (theory and practice). The final mark is obtained by applying the following formula:

Final Mark = Theory Mark * 0.55 + Practice Mark * 0.25 + Working Groups * 0.20

Readings and other material

- Ambler, S.W., *The elements of UML 2.0 style*, Cambridge University Press, 2007
- Jacobson, I., *The Unified Software Development Process*, Addison-Wesley, 2005
- Leszek A. Maciszek., *Requirements analysis and system design: developing information systems with UML*, Addison Wesley, 2001
- Miles, R., *Learning UML 2.0.*, O'Reilly., 2006
- Pilone, Dan, UML 2.0 in a nutshell, O'Reilly, 2005
- Sommerville, I., *Software Engineering, 8ª Edition*, Addison Wesley, 2007

Computer Organization

Objective

What is a computer? How does a computer work? How is it designed? How is it programmed? There are many questions whose answers can be found in the field of Computer Engineering (CE). CE is a field of knowledge with unique characteristics, resulting from the combination of purely technological aspects, as well as problems on organization, structure and optimization, and finally the implementation of the controlling software and its integration with other systems. Hence, in order to provide the required instruction in CE, a set of courses have been included in the Degree on Computer Science that fulfil the formative needs of new graduates.

In the first year of the degree the courses "Tecnología de Computadores" and "Estructura de Computadores" introduce the basic components of computers to the students . These components will be used as essential elements of the designs addressed by the courses of the next years. Thus, in these two courses the "bricks" to build more complex structures, in further courses of the area, are studied.

During the second year, the students have to study the course "Computer Organization, where the different alternatives to build a data-path in multicycle systems (with or without pipelining) are studied, as well as the memory system of a computer, focusing mainly on the cache and virtual memory hierarchy. These are fundamental aspects in a computer; indeed their configuration has a significant impact on computers performance.

During the third year of the degree the students study the course "Arquitectura de Computadores". This course continues directly from the knowledge acquired in the previous course. Specifically, in this course the concepts of pipelining, introduced in the previous course, are expanded. For instance, new techniques to take advantage of the instruction level parallelism are introduced. Moreover, new architectures with the same purposes are introduced, such as superscalar processors. Finally, a broad vision of current processors is given.

With this background the student reaches the fourth year of the degree, to study the course "Computadores Avanzados". In this course the parallel computing systems based on multiple computing nodes, such as multi-computers or multiprocessors, are introduced to the students. The aspects of their structure that differentiate it from the systems with only one processing node are analysed. As an example, special attention is devoted to the networks interconnecting multiple nodes.

Note that the course of the second year is mandatory for those students whose intention is to work designing computing systems because understanding and controlling the basic concepts covered by this course is crucial to design even the simplest systems.

However, this course is also fundamental even for those students whose intention is not to work on the design of systems, in order to break the image of the computer as a black box that magically executes the programs. Indeed, without a deep insight into the processes carried out under the surface, the future graduate will not be able to develop or understand the mechanisms of optimization that allow, for instance, to analyse and understand the performance problems of a system. All these skills are added value, and in fact are increasingly appreciated in almost whichever professional activity related to this area.

Contents

Unit 1. Designing the datapath processor

Unit 2. Pipelining the datapath processor

Unit 3. Cache memory

Unit 4. Virtual memory

Teaching dispositions

In this course we will use as main methods of instruction both lectures and practical sessions to apply all the acquired skills. During the lectures, explanations are interleaved with exercises and similar

activities in order to strengthen the assimilation of the main concepts. At the end of each unit, a complete lecture is devoted to solving different types of exercises related to that unit.

Evaluation

Assessment criteria in the regular exam session: it is obligatory to pass both the practices and the combination of the final exam and the progress tests. Besides, the student must have completed at least 80% of the deliverables (practices reports and questionnaires, critical incidences reports, and reports about the working group) to pass the course. In the case of students of the English group, all the deliverables must be written in English.

Readings and other material

- Patterson, D.A. and Hennessy, J.L., *Computer Organization and Design The Hardware/Software Interface, 5th Edition*, Morgan Kaufman Publishers, 2014

Operating Systems I

Objective

An operating system is a program that manages the hardware of a computer and facilitates the interaction between it and the user. It is, therefore, a low-level software element that acts as an interface between high-level software and hardware.

This subject addresses the question of how a program can finally run on a physical system (hardware), and therefore its knowledge is indispensable for a computer engineer. It can be considered as one of the fundamental subjects of the degree and appears in all similar study plans, both national and international.

This subject covers the specific competencies common to the branch of computing [CO5] Knowledge, administration and maintenance of systems, services and computer applications, and [CO10] Knowledge of the features, functionalities, and structure of the Operating Systems and design and implements applications based on their services.

It is, therefore, a compulsory subject in the curriculum, regardless of the specific technology that the student wants to follow. The knowledge acquired in this subject is complemented with those of other contemporary subjects in the curriculum (Organization of Computers, Real-Time and Concurrent Programming) as well as those of higher-level subjects (Distributed Systems, Operating Systems II).

Contents

Unit 1. Computer Systems Overview

Unit 2. Operating systems: A brief introduction

Unit 3. Processes

Unit 4. Deadlocks

Unit 5. Scheduling

Unit 6. Memory management

Unit 7. Files and directories management

Teaching dispositions

In this subject, a teaching methodology focused on student learning is used. For this, the student must perform a series of individual and group activities, both inside and outside the classroom. These activities aim to get the student to learn the contents of the subject as well as acquire a series of transversal skills such as, among others, autonomous learning and teamwork.

On average, the student must dedicate 8 hours per week to the subject, considering within this time the attendance to theory and laboratory lectures as well as the autonomous or teamwork did outside the classroom.

In the virtual platform, a work plan is set out with all the details of the activities to carry out, when they must be carried out and when the product of these activities (deliverables) must be delivered. It is especially important to meet the delivery deadlines for each deliverable.

Evaluation

The course is divided into 4 parts: theory, exercises, project, and practices. For the ordinary and extraordinary, calls complete parts approved previously (in the continuous assessment or the ordinary call respectively) will be saved. The global mark shall be calculated considering the weight of each part (shown in parentheses).

- Theory (20%) and Exercises (20%): [ESC] There will be 2 controls (progress tests) of theory and exercises throughout the course. A final exam will be scheduled to recover the parts not passed during the course.

- Laboratory (30%): [LAB] There will be 3 controls (progress tests) of practices throughout the course. A final exam will be scheduled for those students who have not passed the practices in the progress tests.
- Project (20%): [INF] Realization of a programming project and a report on the assigned topic (in a group).
- Deliverables (10%) related to the topics of theory and exercises. Some of these deliverables will be made through the virtual campus, others only in class. The mark given depends on the percentage of deliveries as follows: < 80%: 0%; 80-90%: 50%; >90%: 100% point.

To pass the subject it is necessary to pass the block "Theory and Exercises" + "Practices", considering the indicated weighting. In addition, the global grade of the subject must be greater than or equal to 5. That is, it must be fulfilled simultaneously:

- "Theory and Exercises" * 0.4 + "Practices" * 0.3> = 3.5 (out of 7 points)
- "Theory and Exercises" * 0.4 + "Practices" * 0.3 + "Project" * 0.2 + "Deliverables" * 0.1> = 5 (out of 10 points)

Readings and other material

- Stallings, William. "Operating systems: internals and design principles", 7th ed. Upper Saddle River (New Jersey): Pearson education, 2018.
- Tanenbaum, Andrew S. "Modern operating systems", 2nd. ed. Upper Saddle Rivert, New Jersey: Prentice Hall, 2001.

Computer Architecture

Objective

This course is part of the 'Computing Engineering' subject of the degree programme. The course revises the architectural concepts present in most of modern computers, from a laptop to a big internet server, and underlies the next courses: 'Advanced Computers,' 'Operating Systems II', 'Design of Microprocessor-based Systems', and 'Computing Systems Integration.'

With regards to your profession, the knowledge acquired in the course will ease the task of selecting the most suitable computer system for a client. Also, you will acquire basic knowledge to work in the computer design industry.

Contents

Unit 1. Introduction

Unit 2. Instruction-Level Parallelism

Unit 3. Code Scheduling

Unit 4. Branch Handling

Unit 5. Speculation

Unit 6. Multiple-issue Processors

Unit 7. Current Processors

Unit 8. Introduction to Parallel Computers

Teaching dispositions

Face-to-face learning activities include theoretical lectures, problem solving sessions, lab sessions, and the development and presentation of a course final project.

Evaluation

Final course marks consider the quality of both lab reports and the final project, class participation, online progress tests performed at the end of each unit (10%), and the mark obtained in a final written exam.

Readings and other material

- Patterson, D.A. and Hennessy, J.L., *Computer Organization and Design The Hardware/Software Interface, 5th Edition*, Morgan Kaufman Publishers, 2014
- Hennessy, J.L., *Computer architecture: a quantitative approach (5th Edition)*, Morgan Kaufmann, 2012

Software Engineering II

Objective

This subject aims to offer a big picture of the Software Engineering processes and their relations to create work teams that develop information systems industrializing such processes, highlighting that this development is always cooperative and multidisciplinary.

For this aim, the subject introduces the concept of Software Lifecycle and describes a map of the processes necessary to develop software with a work team, explaining its enactment when different software development methodologies are used. Moreover, different important process are presented such as configuration management, quality management, testing management and maintenance management.

As result, it is expected that the student will be able to achieve the abilities and knowledge necessary to work as a Software Engineer.

Moreover, it is important to highlight that this subject belongs to the Software Engineering module, Information Systems and Intelligent Systems of the curriculum and provides the basis for the subjects of the speciality of Software Engineering:

- Requirements Engineering
- Software Design
- Process of Software Engineering
- Quality of Software Engineering
- Software Project Management
- Databases Development
- Enterprise Information Systems
- Security of Software Systems

Contents

Unit 1. Software Configuration Management

Unit 2. Software Testing: Introduction and Principles

Unit 3. Software Testing: Strategies and Processes

Unit 4. Software Quality

Unit 5. Software Maintenance

Teaching dispositions

In this course we will use as main methods of instruction lectures, seminars, problem resolution, and practices of laboratory. Understanding the patterns, the rules and concepts require a high level of creative and critical thinking. In this course we want you to experience these levels of thinking and develop a high degree of conceptual understanding. Learning to solve problems with a conceptual understanding is essential and so we aim to teach you not only the general principles but how to apply them in specific situations.

Evaluation

In order to pass this subject, the student will be required a minimum pass grade of 5, satisfying the following criteria:

- Theory [ESC]: 2,25 out of 4,5. The grade of theory will be calculated as: (Final term exam * 0. 225) + (test *0.225) if the student has passed both the test and the final term exam. If the student has failed the test but has passed the final term exam, then the grade of theory will be calculated as: (Final term exam * 0.45). Otherwise, the grade of theory will be 0.
- Laboratory [INF][LAB]: 2,25 out of 4,5. The grade of Laboratory will be kept for future assessments if the student has a grade of Laboratory higher or equal than 2.25 (out of 4.5).
- Class activities up to 1 point.

Readings and other material

- Brown, W.J., Mccormick, H.W., Thomas, S.W., *AntiPatterns: refactoring software, architectures and projects in crisis*, John Wiley and Sons, 1998
- Collard, J.F. Burnstein, I., *Practical Software Testing: A Process-Oriented Approach*, Springer, 2003
- Galin, D., *Software Quality Assurance: From theory to implementation*, Addison-Wesley, 2003
- Kan, S.H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1995
- Kaner, C., Nguyen, H.Q., Falk, J., *Testing Computer Software*, John Wiley & Sons, 1999
- Leon, A., *Software Configuration Management Handbook*, Artech House, 2005
- Pigoski, T.M., *Practical Software Maintenance*, John Wiley & Sons, 1997

Human-Computer Interaction I

Objective

A user interface is the visible part of an application. In the discipline of human-computer interaction, designing a proper user interface is understood as a must. The user should perform the tasks easily. Obviously, the interface must be aesthetically pleasant, but always having in mind that the main objective is to ease the user interaction with the application. Throughout this course, students will develop applications considering fundamentals regarding the person, the mechanisms of interaction or some design rules.

Contents

Unit 1. Introduction to HCI

Unit 2. The Human

Unit 3. The Computer

Unit 4. The Interaction

Unit 5. Design Rules

Unit 6. HCI within the SE Process

Teaching dispositions

In this course we will use as main methods of instruction: lectures, assignments, oral expositions and practical sessions in group to apply all the acquired skills. Therefore, the students will develop a software application applying the concepts learnt in theory activities, as well as the ones required by the own nature of their application.

Evaluation

To pass the subject, it is required a minimum mark of 50% in every part (Class activities, Lab, Theory Exam)

Depending on the quality of the work made by the student during the course, teachers might rise the mark up to 1 extra point in the subject.

Readings and other material

- Dix, A., Finlay, J., Abowd, G.D., Beale, R., *Human-Computer Interaction*, Prentice Hall, 2004
- Steve, K., *Don't Make Me Think: A Common Sense Approach to Web Usability*, Pearson Prentice Hall, 2006

Intelligent Systems

Objective

This course introduces the basic techniques for Artificial Intelligence in the degree. Such techniques are often required nowadays for the solution of complex problems: decision making, diagnose systems, control and monitoring, web search, semantic web, recommended systems, machine learning, data analysis and mining, vision, robotics, etc.

The course certainly requires some other previous ones in the program - discrete maths, logic, programming- and is a pre-requisite of some other subsequent courses such as data mining, knowledge based systems, multi agent systems, artificial intelligence, or robotics.

It is also a co-requisite which allows defining a software project with some other courses such as information systems, data bases or software engineering.

Contents

Unit 1. Introduction

Unit 2. Search in the state-space

Unit 3. Heuristic search

Unit 4. Adversarial search

Unit 5. Learning agents

Unit 6. Combinatorial optimization problems.

Unit 7. Metaheuristics. Local search

Unit 8. Metaheuristics. Genetic algorithms

Unit 9. Rule-based systems

Unit 10. Machine learning

Unit 11. Classification, rules and trees

Unit 12. Classification, naive Bayes

Teaching dispositions

This course is divided into class and practical sessions. In the classroom, besides lectures, we will do and solve different problems and exercises. As for the practical assignments, during the course we will have to develop three different projects where we will apply the studied concepts and techniques.

Evaluation

- It is compulsory to pass the theory part in order to pass the subject (mark >= 5).
- It is compulsory to pass all the assignments (mark >= 5) in order to pass the subject.
- Doing the problem sets and/or questionnaires is not compulsory, but highly recommendable.
- The mark for the theory part will be maintained for the extra exam session.
- The mark for each laboratory assignment will be maintained for the extra exam session.

Readings and other material

- Nilsson, N.J., *Artificial Intelligence: A New Synthesis*, McGraw Hill, 2000,
- Russell, S., Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2010

Web and Services Engineering

Objective

This course for senior students is in the seventh semester. The aim is to bring students with the skills to design and develop proper web systems by applying an engineering point of view. They will be able to use the latest techniques for web design and development.

Nowadays, our society is suffering a structural change towards an interconnected paradigm where everybody can access to a wide range of resources via the Internet, but what about the companies themselves? Can they use or share the resources on the Internet? Most resources are available on-line. Thus, it is possible not only to share data resources but the same applications. That is, companies or different department in huge companies can remotely access resources (data or programs). However, it implies a big effort of standardization. Some of the current trends are web services and cloud computing. This last one is an extended version of the previous that solves some problems as for example stateless and where the services are executed. In this course we will see how these new technologies can help software engineers to build systems that can be used as bricks and foundations to bigger systems. These new systems will be platform-independent since the interaction between each construction block is standardized.

The point of view to tackle this course is from an engineering one. In this matter, we will use what it is known as Service Oriented Architectures. This feature will make possible to interconnect this course with the previous Software Engineering I and II courses and furthermore with the Web Technologies course taught in the same semester.

Contents

The information access is a key issue in globalized societies. It is becoming essential to carry out everyday tasks (i.e. from food delivery to ordering products from one company to another automatically). To carry out these tasks, we need to connect computers that provide services to those that consume them.

The Web has become the preferred platform to carry out this communication. Initially, it was employed to communicate users. Later on, it allowed users to access different resources and nowadays, it allows them to share software infrastructure or applications.

The Web as a communication platform requires protocols to exchange information. These protocols should follow standards in order to improve the software reuse, maintenance and scalability. Nowadays, the concepts that support these communication issues are the Service Oriented Architecture and Cloud Computing.

This course focuses on the development of systems capable of using these technologies to develop applications that use other systems to fulfil their goals. To successfully develop this type of applications, developers should maximize platform independence, applications' decoupling and communication standardization.

This course is highly related to subjects such as Software Engineering I and II. In addition, there are commonalities with courses of the Software Engineering specialization (i.e. Software Design and Software Engineering Processes) and the Information Technology specialization (i.e. Technology and Web Systems and Integration of Information Systems).

Teaching dispositions

In this course we will use as main methods of instruction lectures, seminars, problems resolution and practices of laboratory. Software development is basically an engineering science. We construct prototypes by using different techniques during the analysis, design and deployment phases. Furthermore, the use of e-learning platforms like Moodle allows students to work from their homes and follow auto-learning easily.

Evaluation

The evaluation process not only considers exams but the whole student's improvement in all the intended skills. Therefore, the evaluation will be carried out by a daily check of the proposed activities, i.e., by grading all the theoretical and practical assignments, plus some quizzes.

Readings and other material

- Hasan, J., Duran, M., *Expert Service-Oriented Architecture in C#,* Apress, 2005
- Iyengar, A., Jessani, V., Chilanti, M., *WebSphere Business Integration Primer: Process Server*, BPEL, SCA, and SOA, IBM press, 2007
- Kumar, B.V., Narayan, P. and Ng, T., *Implementing SOA Using Java EE*, Prentice Hall, 2010
- Linthicum, D.S., *Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide*, Addison-Wesley Professional, 2009
- Hewitt, E., *Java SOA Cookbook*, O'Relley media, 2009
- Burke, B. *RESTful Java with JAX-RS*, O'Relley media, 2009
- Ganci, J., Flugrath, M., Manekar, A., Trasti. M., *Best Practices for SOA Management*, IBM red books, 2007 (http://www.redbooks.ibm.com/redpapers/pdfs/redp4233.pdf)
- Keen, M., Ackerman, G., Azaz, I., Haas, M., Johnson, R., Kim, J., Robertson, P., *Patterns: SOA Foundation - Business Process Management Scenarios*, IBM red books, 2006 (http://www.redbooks.ibm.com/redbooks/pdfs/sg247234.pdf)
- SOA Patterns, http://www.soapatterns.org/
- Zirn, M., Gaur, H., Eds. *BPEL Cookbook: Best Practices for SOA-based integration and composite applications development*, Packt Publishing open source, 2006

English Subjects
Spring Semester

Programming Fundamentals II

Objective

As its name indicates, it will cover the fundamental concepts and skills of programming, in this particular case object-oriented programming (OOP). This methodology is basic for the academic training of our students, in a two-fold way: in order to acquire the necessary basis for studying more advanced courses within the degree, but also aiming at their professional life. In this subject, event-driven programming is also introduced, and the students learn how to apply it for a particular set of defined problems. Students are also taught the possibility of creating by themselves computer applications from the perspective of OO paradigm.

Contents

Unit 1. Introduction: Fundamental concepts of Object-Oriented Programming (OOP)

Unit 2. Object-Oriented Programming

Unit 3. Exception handling

Unit 4. Introduction to event-driven programming

Teaching dispositions

In this course we will use as main methods of instruction lectures and practical sessions to apply all the acquired skills. Class sessions would include some general explanation from the lecturer, using as the basis the slides corresponding to the current lesson/unit. To illustrate the concepts, the lecturer will also include some examples in Java code, and some further exercises will be proposed, so that the students can write their code individually. With respect to laboratory sessions, they could also include some general explanation from the lecturer (time depends on the topic) and specific practical exercises. At every week the student will have to solve a particular lab assignment whose guide and extra material, if necessary, will be available. In the laboratory, students will consequently develop specific Java projects exemplifying the concepts learnt in theory activities by means of the weekly lab assignment. Notice that laboratory tests will be done with the computer, and will consist on modifications of the original solutions done by the student.

Evaluation

There is not a final term exam. Each student's final grade in the regular assessment will be based on the results from her/his work developed throughout the course (tests + submitted tasks + participation). Some activities/tasks can be compulsory or require a minimum pass grade, which will be notified. The following must be satisfied in order to achieve a pass grade in this subject: the minimal requirements have to be reached and the total score of the sum (weighted sum according to the previous table) of all the tasks to be assessed cannot be less than 50% of the maximum possible score.

Readings and other material

- Eckel, B., *Thinking in Java*, Prentice Hall, 2006
- Deitel, P. *Java How To Program (Early Objects) (10th Edition)*, Pearson - Prentice Hall, 2014
- Liang, Y.D., *Introduction to Java Programming, Comprehensive Version (10th Edition)*, Pearson/Prentice Hall, 2014

Computer Networks I

Objective

Computer networks I is a compulsory course in the Computer Engineering programme. Its contents are essential to the practice of the computer enginnering profession. It falls into the area of Operating Systems, Distributed Systems and Networks" of the curricula. It sets the principles of the following subjects:

- Computer Networks II
- Network Management and Design
- Design of Network Infrastructures
- Information Systems Security
- Network Security
- Network Management and Administration
- Planning and Integration of Systems and Services

Contents

Unit 1 Introduction to computer networks

1.1 Basic Concepts

1.2 Terminology

1.3 Network Architecture

1.4 Network Examples

Unit 2 Internet and applications

Unit 2.1 Motivation

Unit 2.2 Evolution

Unit 2.3 Examples of applications: Web, messenger, ftp, telnet, mail, etc.

Unit 3 Network Layer and IP Addressing

3.1 Principles of internetworking

3.2 IP addressing: subnets and VLANs

3.3 IP datagram

3.4 Fragmentation and reassembly

3.5 Other network protocols

Unit 4 Data link layer and local area networks

4.1 Medium access control

4.2 Flow control

4.3 Error control

4.4 Interconnection devices

4.5 Ethernet

4.6 Token Ring

4.7 Wireless LANs

Unit 5 Physical Layer

5.1 Terminology

5.2 Media transmission

5.3 Digital communication principles

Teaching dispositions

This course introduces the student into the design and operation principles of a computer network. The course has an important lab component comprising nine labs. Students make use of a simulation tool, protocol analyser, and state-of-the-art network lab. The use of such facilities allows student to go through the design and operation analysis of a computer network. Students get to learn the importance of working in teams.

Evaluation

To be eligible to pass the regular final exams, the student must previously complete all lab sessions. Previous to each lab session, the student must hand in a preliminary work report. Failure to do so will result in the ineligibility to participate in the corresponding lab session. If for a justified reason a student misses to participate in one or two lab sessions, he/she will be able to attend a lab catch-up session scheduled by the end of the semester. The final regular exam will consist of a written exam divided into a theory part (50%), and a lab part (20%). The students having obtained a passing mark in each one of the two partial theory exams scheduled during the semester will be exempted from taking the final theory exam. In case of having failed any of the two partial exams, the student will have to take the corresponding part of the final theory exam. Note: the passing mark requires to get at least 5/10 in each exam.

Readings and other material

- Behrouz, A.F., *Data Communications and Networking*, McGraw-Hill, 2007
- Cisco, *Cisco Packet Tracer* 6.2, 2015, (https://supportforums.cisco.com/document/12739396/cisco-packet-tracer-62-free-download-link)
- Stallings, W., *Data and Computer Communications*, Prentice Hall, 2008

Information Systems

Objective

The Information Systems course aims to provide a transversal and integrating vision of the Computer Science context, relating the business and organisational aspects with the specific objectives of IT (Information Technology) systems. To this end, the basic concepts managed in IT (engineering, abstraction, model, system, project, process, information) are introduced, to then come up with the idea of a computer system (hardware + software + data). From this basis, the more global vision of an information system is developed as a special type of socio-technical system (with technological, human, social, and organizational elements) whose objective is to satisfy the information needs of an organization.

Contents

Unit1: Introduction to information systems

1.1: What is an information system?

1.2: Information system model

1.3: Classification of information systems

Unit 2: Module II: Information management

2.1: What is a database?

2.2: Database management systems

2.3: Relational databases: The relational model

2.4: Data models: The entity/relationship model

2.5: Relational languages: Introduction to SQL

Unit 3: Module III: Development of computer systems

3.1: Introduction: Fundamental concepts

3.2: Software process models

Unit 4: Module IV: Security in information systems

4.1: Introduction to computer security

4.2: Types of encryption

Teaching dispositions

In the course the following methods of instruction are used: lectures, lab sessions, and teamwork projects. Lectures are supported by problem sheets and tutorial hours. Lab sessions allow students to use appropriate software and solve problems on information systems. Teamwork projects also give students the opportunity to develop their collaboration and communication skills.

Evaluation

An exam is scheduled for each main unit of the course. Lab activities are also assessed, and the students have to take an assignment about the first Unit 1 (presentation).

Readings and other material

- McLeod, R., and Schell, G., *Management Information Systems (10th edition)*, Prentice Hall, 2007
- Stephens, R., *Beginning Software Engineering*, Wrox, 2015
- Nield. T., *Getting Started with SQL: A Hands-On Approach for Beginners*, O'Reilly, 2008
- Olivé, A., *Conceptual Modeling of Information Systems*, Springer, 2007

Statistics

Objective

The statistics course is the only course where students learn statistical techniques in the degree. The student must learn to make decisions based on data and how to represent them.

This course aims to:

- Describe and represent large amounts of data through the main measures of location and dispersion and be able to use graphs.
- To help students acquire the necessary skills for modelling situations with "Variability" techniques.
- Basing the decision-making process in general situations on the basis of incomplete information.
- To familiarize the future with computer techniques that directly reflect key statistics related to computer systems situations, and to use in the exercise of their profession.

In addition you will learn to use SPSS software and a very powerful language such as R, available for free download specific packages and allows for a multitude of task statistical program.

Relationship to other courses:

- This is a course of such vital importance that students acquire a working method and a way of thinking and dealing with the difficulties of logic and rigorous manner. The course will take an interdisciplinary sense connecting problems and proposed materials and examples with other subjects of the curriculum. The studied concepts are used in almost all courses of enhanced smart systems as well as in matters related to the study of large amounts of data.
- The student will describe tools for models with uncertainty and make decisions in the presence of this uncertainty.

Relationship with the profession:

- Statistics is a transversal field in a wide variety of disciplines, from physics, chemistry to social sciences. In recent decades, the quality control has approached statistical virtually in all businesses and is used for decision making in almost all areas.
- In computing, it is common use for reporting and is also frequently used in areas such as data mining, where there is an increasing number of computer professionals. Any consultant should have basic knowledge of statistics, like any computer analyst must know based inference techniques.

Contents

Unit 1. Descriptive statistics

1.1 Measures of central Lower

1.2 Measures of Central tendency not

1.3 Measures of dispersion

1.4 Graphing

Unit 2. Probability

2.1 Conditional probability Theme

2.2 Subject Rule Bayes

Unit 3. Random Variables

3.1 Continuous Random Variables

3.2 Discrete Random Variables

Unit 4. Statistical sampling

4.1 Sampling in normal populations

Unit 5. Statistical Inference

5.1 Estimation timely

5.2 Interval estimation

Unit 6. Hypothesis testing

6.1 Parametric methods

6.2 Non-Parametric methods

Unit 7. Analysis of variance

Unit 8. Regression and Correlation

Teaching dispositions

The methods of instruction of this course are mainly lectures, practical sessions, where we apply the concepts learnt in lectures by means of a statistical software, and project-oriented learning where the students develop acquired skills.

Evaluation

In the case of not passing the theoretical part, an examination must be faced with a minimum grade of 4 out of 10. It contributes with 50% to the final grade.

The practices may be retaken with an examination of the practices in class, representing 25% of the grade.

Problem solving could be retaken with the theory exam, considering 75% of this note.

Readings and other material

- Diez, D.M., Barr, C.D. Cetinkaya-Rundel, M., *OpenIntro Statistics*, 3rd Ed. 2015, (http://www.openintro.org/stat/textbook.php)
- Devore, J.L., *Probability and Statistics for Engineering and the Sciences*, International Thomson, 2001
- Montgomery, D.C., *Applied statistics and probability for engineers*, Limusa Wiley, 2007
- Walpole, R.E., *Probability and Statistics for Engineers and Scientists*, Prentice-Hall Hispanoamericana, 1999

Programming Methodology

Objective

It provides appropriate methodology for solving complex / real problems that require more abstract approaches than those provided by the courses of Programming Fundamentals.

It follows the learning program developed in both "Programming Fundamentals" and "Data Structures", and will be continued by both "Design of algorithms" and "Software Engineering" courses.

Contents

Unit 1. Algorithmic complexity

1.1. Definition. Temporal complexity

1.2. Asymptotic complexity orders

1.3. Basic calculations

1.4. Real examples and Recursive Equations. Characteristic Equation. Non-homogenous equations. Variable changes. Domain changes.

Unit 2. Greedy Algorithms

2.1. Overall technique

2.2. Basic features

2.3. Examples. Coins, the knapsack problem, scheduling, minimum spanning tree, single-course shortest paths problem

Unit 3. Dynamic Programming

3.1. Overall technique

3.2. Basic features

3.3. Examples. Coins, the knapsack problem, banks, optimal binary search trees, all-pairs shortest path problem, optimal binary search trees, disk space, ...

Unit 4. Backtracking

4.1. Overall technique

4.2. Basic features

4.3. Examples. Generation of combinatorial objects, chess, graph colourings, cliques, Hamiltonian cycles, Sudoku, ...

Teaching dispositions

In this course we will use as main methods of instruction: lectures focused on firstly discussing the way to deal with exercises the students are supposed not to know how to solve. The professor acts as a chairman so rationally analysing the appropriateness of the proposed solutions. If the students do not find the good way to design the solution, the professor would manage to lead the dissertation to the more promising way to solve it, and in the end, present an informal solution for the exercise to be solved. Afterward the formal theory supporting it is presented. There are lab sessions to fully apply each new algorithm/strategy they are supposed to be able to choose and define. This is the main skill to be acquired in the subject.

Evaluation

Optional written test with 30% weight in the final grade (free for evaluation provided that passed)

Practice: 30% (3 x 10%)

Final test:

- If you pass the optional test described in the first line of this space you are allowed to only take the final test referring to:
  - Additional outcome 2: Choosing and implementing the computationally cheapest methodology which solves any given problem. It contributes with 40% to the final grade.
- If desired or failed the test explained in the first line of this space you must take an additional one that contributes 70% to the final grade. This test will cover:
  - Additional outcome 1. Ordering algorithms according to their computational complexity
  - Additional outcome 2. Choosing and implementing the computationally cheapest methodology which solves any given problem

Final Rating: Assuming hereinafter "Test", "final test" and "Practice" ranked out of 10.

- If [test exceeded 50% and exceeded 50% Final Test (only examining additional 2 Final test result)] GRADE = 0.3* Test + 0.4*final test + 0.3*Practice
- If [Final Test (examining additional outcomes 1 and 2) exceeded 50%] GRADE = 0.7 * final test + 0.4 * Practice

Readings and other material

- Aho, A.V., Hopcroft, J.E., Ullman, J.D., *The Design and Analysis of Algorithms*, Addison-Wesley, 1974
- Brassard, G., *Fundamentals of Algorithmics*, Prentice-Hall, , 2006
- Horowitz, E., *Fundamentals of computer algorithms*, Computer Science Press, 1978
- Parberry, I., *Problems on algorithms*, Prentice-Hall, 1995
- Sedgewick, R., *An introduction to the analysis of algorithms*, Addison-Wesley, 2013

Concurrent and Real Time Programming

Objective

The Real Time and Concurrent Programming course is a compulsory subject of the Degree in Computer Engineering and its justification is directly linked to the evolution of operating systems, particularly the concepts of multithreading and multiprocessor. In this context, the student must know and master the basic techniques of management of racing conditions, such as synchronization and communication mechanisms between processes. Also, programming for real-time systems is another key aspect to obtain the skills necessary to design and implement this type of critical systems that are part of our daily lives.

In the professional market, a graduate must ensure some knowledge on concurrent and real-time scheduling, taking particular account of market developments processors and processing systems in general, which show a clear trend of multiprocessing .

Contents

Unit 1. Introduction

1.1 Basic Concepts I

1.2 Basic Concepts II

Unit 2. Busy Wait Synchronization

Unit 3. Shared-Memory Communication

3.1 Semaphores

3.2 Conditional Critical Regions

3.3 Monitors

Unit 4. Message-Passing Communication

Unit 5. Real Time

Teaching dispositions

This is a very practical course so, in addition to lectures and programming lab assignments, students will perform several group presentations in which they present the solution to a concurrency-related problem. The CLIL context is taken into account in the design of lectures and group tasks in order to provide an appropriate scaffolding to the learners along the course.

Evaluation

The score of the regular exam session is the total score. The mark of the final exam could be replaced by midterm exams in lecturing (theory) time.

The lab assignments are to be presented to the lecturer in the lab time, always before the indicated deadlines.

To pass the course you need to pass both parts (final exam mark and lab sessions mark) getting a 5 on each part out of a total of 10.

Failing mid-term exams counts as a fail in the regular exam session.

Readings and other material

- Ben-Ari, M., *Principles of concurrent and distributed programming*, Addison-Wesley, 2006
- Burns, A., *Concurrent programming*, Addison-Wesley, 1993
- Hartley, S.J., *Concurrent programming: the Java programming language*, Oxford University Press,1998
- Lea, Doug, *Concurrent Programming in Java*, Addison Wesley, 2000

Autonomous Robotics

Objective

This course will introduce students to the fundamental constraints, technologies, and algorithms of autonomous robotics. The focus will be on computational aspects of autonomous wheeled mobile robots. The most important themes will be mobility, perception, and navigation. Assignments will require the implementation of controllers for the LEGO Mindstorms NXT robot kits.

Content

Unit 1. Introduction

Unit 2. Mobility

2.1 Methods of Locomotion

2.2 Kinematics of Wheeled Robots

Unit 3. Perception

3.1 Sensors and Ranging

3.2 Feature Extraction

Unit 4. Localization

4.1 Maps, Beliefs, and Probability Review

4.2 Introduction to Markov Localization

4.3 Motion and Measurement Models + Grid Localization

4.4 Monte Carlo Localization

4.5 The Kalman Filter and Extended Kalman Filter

Unit 5. Advanced Topics

5.1 SLAM, Planning, Probabilistic Planning

Teaching dispositions

The activities and methodologies used to conduct learning process fall into three groups:

- Guided Activities: Lectures, laboratory exercises, and seminars on problems and cases.
- Self-employment activities: study, problem solving and case preparation, and preparation of laboratory practice.
- Evaluation Activities: Assessment of written evidence and/or oral assessment of laboratory work and/or cases, delivery and evaluation of reports, problems, etc.

Below are guided activities:

- Lectures (face to all): consist of the explanation by the professor of the relevant contents of the topic and practical examples, and the approach to follow to solve exercises and problems.
- Practical laboratory (work in small groups): explanation and presentation by the professor to develop the case study; development and implementation of work by the group; individual monitoring of each group and resolution of general and specific questions.
- Seminars and case problems: presentation of the problems and cases to solve; proposed solutions to such problems in small groups; resolution of these problems and solution explanation by the professor.

As for self-employment activities are proposed:

- Study: it consists of the study and assimilation of the theoretical contents explained in each unit, both for the subsequent resolution of the exercises and problems proposed for the preparation of the written or oral presentations.

- Troubleshooting and case preparation: completion and delivery of the exercises in problem sets.
- Preparation of laboratory practice: preparation of laboratory practice sessions by the assigned work within the group; bibliography search, preparation of reports to present tasks of implementation, evaluation of the carried out work, etc.

And evaluation activities will include:

- Evaluation of written evidence and/or oral (individually); evaluation of the resolution of exercises and/or problems similar to those proposed and solved in the guided activities.
- Evaluation of laboratory work and/or cases (by group); evaluation of practices.
- Evaluation of deliverables, problems, etc. (individually); evaluation of the resolution of the problem sets.

Evaluation

The evaluation of the course will be carried out by a continuous and formative assessment during the course. This will include evaluation of each and every one of the proposed activities. The final mark for the course will be averaged according to the weight associated to each activity.

To successfully pass the course, the student must obtain a mark equal to or greater than 4 (40% of valuation) in each of the separate activities and to obtain a final score (with a weighted average) equal to 5 points.

Readings and other material

Textbook:

- Thrun, S., Burgard, W., Fox, D., *Probabilistic Robotics*, MIT Press, 2005

References:

- Siegwart, R., Nourbakhsh, I.R., *Introduction to Autonomous Mobile Robots*, MIT Press, 2004
- Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., and Thrun, S., *Principles of Robot Motion*, MIT Press, 2005
- Dudek, G., Jenkin, M., *Computational Principles of Mobile Robotics*, Cambridge University Press, 2000
- Trucco, V., Verri. A., *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998

Graphic Design and Animation

Objective

This course is part of a group devoted to providing the students with the basic skills needed to develop a computer videogame. The other courses are "Graphic Processing Units", "Videogames and Virtual Reality" and "Artificial Intelligence for Videogames". Every one of those approaches this topic from a different point of view but in such a way that those students that follow all these courses can have a complete vision of the main techniques required to develop a videogame. In any case, each subject is an independent unit, which means that a student can take full advantage of the topics covered in one course without taking the other ones. In the specific case of "Graphic Design and Animation", it is focused on the creation and processing of 3D graphic models and animation to be used to create pictures or movies as well as to be employed as part of a computer videogame.

Contents

Unit 1. Introduction and basic concepts in computer graphics and animation

Unit 2. Introduction to Blender.

Unit 3. Polygon modelling techniques

Unit 4. Other modelling techniques.

Unit 5. Material and textures

Unit 6. Lighting and cameras

Unit 7. Animation I. Basic animation in Blender

Unit 8. Animation II. ShapeKeys and armatures

Unit 9. Animation III. Character creation and animation.

Unit 10. Introduction to Unity and Game Asset Pipeline

Unit 11. Particle systems

Unit 12. Physics

Teaching dispositions

In this course, lectures and lab sessions are tightly integrated so that the students can better assimilate it. Besides, the learned skills are applied by developing several project involving real case scenarios that has to be addressed with the techniques learnt throughout the course

This course will use different methods of instruction, namely lectures, exercises in class, practical sessions in the lab and teamwork for the development of a final project. Students will have slides available in English for each of the units. The exercises in class will serve to consolidate the knowledge acquired during the lessons, while the laboratory sessions will be used to carry out specific practical exercises. Finally, students will develop a final project in group in which they will apply the different techniques learnt during the course.

Evaluation

The core of the evaluation lies on the development on several project where the different techniques taught in the course will be applied to solve a real task related to the subject scope. Besides, some specific and small tasks will be proposed during the course. Solving these tasks appropriately will improve the student's grades.

There will be three main deliverables: one about modelling techniques, another about animation techniques, and a final one that will include all the knowledge learnt during the course.

Readings and other material

- Blain, J.M., The complete guide to Blender graphics. 2016

- Chroniste, J., Blender Basics, Classroom Tutorial book, 2011.
- Blain, J.M., An introduction to BLENDER 3D - A book for beginners, 2011

Artificial Intelligence in Videogames

Objective

The world of video games has an important place in today's society. So much that their level of business is even greater than other types of entertainment such as films. There is therefore a great need for experts in this domain. In the development process of a video game, the component associated with the behaviour of its elements, requires knowledge of artificial intelligence techniques. Numerous videogame programmers deal with this type of resource. This course contributes to the formation of this profile of professionals.

Contents

Unit 1. Introduction

Unit 2. Video Games classification

Unit 3. Basic concepts

Unit 4. Learning

Unit 5. Basic behaviour

Unit 6. Search techniques

Unit 7. Rule-based systems

Unit 8. Case based reasoning

Unit 9. Connectionism

Unit 10. Evolutionary computation

Unit 11. Agents

Unit 12. Evolutionary behaviour

Unit 13. Future possibilities

Teaching dispositions

In this course we will use as main methods of instruction lectures, seminars, practical sessions and mainly project-oriented learning to develop the skills associated with the subject. Therefore, the students will develop several projects applying the concepts learnt in theory activities. A joint project is also developed during practice sessions

Evaluation

The student's assessment is done mainly by the development of course work. We will also consider the exam marks related to the theoretical concepts presented in the classroom, as well as student participation in debates and exhibitions; and the contribution of concepts to work, both for its workgroup, and the rest of the work that the other groups perform during the course.

Readings and other material

- Russell, S., Norvig, P., Artificial Intelligence: A Modern Approach, Prentice Hall, 2010

Videogames and Virtual Reality

Objective

The aim of this course is to provide a comprehensive introduction to videogame and virtual reality technology and development, with special attention to input and output devices.

Contents

Unit 1. From videogames to virtual reality

Unit 2. The development process. Modelling and programming

Unit 3. Mathematics and physics for videogames

Unit 4. Input devices

Unit 5. Display devices. Sight

Unit 6. Sound devices. Hearing

Unit 7. Haptic devices. Touch

Unit 8. On-line multi-user worlds and games

Teaching dispositions

The main methods of instruction will be lectures and problem resolution at the classroom, and seminars and work assignments in the laboratory.

Evaluation

Final grades will be based on laboratory assignments and written exams. Attendance to lectures and seminars is mandatory.

Readings and other material

- Burdea, G.C., Coiffet, P., *Virtual Reality Technology (2nd Edition)*, Wiley-InterScience, 2003
- Hawkins, K., Astle, D., *OpenGL Game Programming*, Prima Publishing, 2004

Both Semesters

Professional Training

Objective

Sensitive to the problems of labour integration of graduates in their duties as future practitioners, the University of Castilla-La Mancha, and this School in particular, offers its students a program of professional training that enables them to carry out this training by working in companies and institutions in accordance with their interests.

Contents

No applicable for this subject.

Teaching dispositions

This will be customized according to the specific needs of the company.

Evaluation

Both an academic supervisor and a professional supervisor will issue a report evaluating the student's commitment, attitude, development, etc. The student will also present a report describing the activities carried out during their professional training, the trained skills, etc. All these reports will be evaluated by the academic committee of the centre to score the student's work.

Readings and other material

They will be provided by the company.

Final Project

Objective

The final project (hereinafter FP) is the first professional work carried out by the students during their formative stage in our School. The FP, carried out in the final phase of their studies, must put into practice the knowledge acquired in the courses taken, constituting a synthesis and colophon of their studies.

Its development is an essential requirement to finish the degree in Computer Science. The purpose of the FP is for the student to develop a personal work where he applies their knowledge, experiences, skills and creativity and originality for the solution of real problems. The FP will provide solutions in different fields of computer science. The student will specially focus on the engineering or methodological approach in its implementation. It may contain prototypes, theoretical models, algorithms, specifications, analysis, and design of both software and hardware components, but in all cases will have a remarkable part related to the competences of the Specific Technology he is taken.

Contents

The FP will be done on a subject related to the competences of the Specific Technology that the student takes, which are:

- Computer Engineering
- Computer Science
- Information Technology
- Software Engineering

Teaching dispositions

This will be established by the supervisor of the FP, according to the needs of the FP.

Evaluation

Once the student finishes the assigned work, he will present the results to a committee related to the Specific Technology of the FP. This committee will consist of three members who will examine, at the oral examination of the FP, the presented thesis, the quality and complexity of the work, as well as the student's presentation.

Readings and other material

The material will be provided by the supervisor of each FP.

English Friendly Subjects

Databases

Objective

The goals of this subject are:

- To know the characteristics and structures of databases, the capabilities of the database management systems as well as the data manipulation and data management languages.
- To develop software applications that use databases.

Contents

1 Unit 1. Database Management Systems
   1.1 Origin and evolution of databases
   1.2 Database concept: Purpose of Databases
   1.3 Data independence. Database architecture
   1.4 Database Management Systems
   1.5 Database Management
2 Introducing database design
   2.1 Database design challenge
   2.2 Design steps
   2.3 Extended Entity-Relationship Model
3 Relational model concepts
   3.1 Structure of relational databases
   3.2 Integrity constraints
   3.3 The theory of functional dependencies
   3.4 Relational languages. Relational algebra
4 relational database design
   4.1 Introduction
   4.2 normalization based on the notion of functional dependencies
   4.3 Logical design. Mapping E/R models to relational models.
5 Introduction to active databases
   5.1 Concepts
   5.2 Active rules
   5.3 Triggers. Applications.

Teaching dispositions

In the course the following methods of instruction are used: lectures and practical sessions.

Foreign students will have the class notes and tutorials in English and Spanish version. They can interact in English with their professors.

Evaluation

The student must pass the theory and laboratory exams, as well as a work of the subject. In this work, the student must demonstrate their capabilities to design a postgres database based on user requirements. This work is a group activity. Foreign students can do this work together other foreign students or Spanish students of the bilingual group.

Exams will be written in English.

Readings and other material

- Silberschatz A., Korth H. F., Sudarshan S., "Database System Concepts", McGraw-Hill, 2010.
- The official site for PostgreSQL, https://www.postgresql.org/.

Theory of Automata & Computation

Objective

The goal of this subject is to introduce the theory of computation through a set of abstract machines that serve as models for computation - finite automata, pushdown automata, and Turing machines - and examines the relationship between these automata and formal languages by means of the corresponding grammars and regular expressions when they apply

Contents

1. Introduction to Computability
2. Formal Languages
3. Formal Grammars
4. Finite Automata
5. Regular Expressions
6. Context-free Grammars
7. Push-down Automata
8. Turing Machines

Teaching dispositions

In the course the following methods of instruction are used: lectures, practical sessions and lab work supported by JFLAP tool.

English spoken students can use English for their questions, therefore they will be answered in English. In addition, they can use also English at office

Evaluation

Tests for English spoken students can be fully performed in English if they want.

The course is assessed according to continuous evaluation policy which is mainly supported by a couple of lab. job assignments and a partial test, in addition to the regular exam at the end of the semester.

Readings and other material

- Brookshear, J. G. "Theory of Computation: Formal Languages, Automata and Complexity", Benjamin/Cummings Series in Computer Science, 1989.

Multimedia

Objective

The goals of this subject are:

- Identify which are the content and composition of multimedia data
- Study the characteristics of the international standards involved in Multimedia content.
- Develop Multimedia applications and systems

Contents

Unit 1. Introduction to Multimedia

Unit 2. Acquisition, Formats and Metrics of Audiovisual Signals.

Unit 3. Image, Video, Audio Coding Techniques

Unit 4. Image Compression Standards

Unit 5. Video Compression Standards

Unit 6. Advanced Video Compression Standards

Unit 7. Audio Compression Standards

Unit 8. Multimedia Systems

Teaching dispositions

In the course the following methods of instruction are used: lectures, practical sessions, case of studies and problems resolution. Equivalent material in English is provided.

Evaluation

Exam (50%), Lab Sessions (30%), Problems Resolution in Class (20%)

Exam, Lab Session and Problem Resolution is provided in English.

Readings and other material

- Ohm J-R. *Multimedia Signal Coding and Transmission*, Springer. 2014
- Shi Y.Q., Sun H. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*, 2nd. Edition. CRC Press, 2008
- Richardson I. *The H.264 Advanced Video Compression Standard*, 2nd. Edition. John Wiley & Sons, 2010
- Sze V., Budagavi M., Sullivan G.J. *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Springer. 2014.

•

Language Processors

Objective

This course provides an overview of the principles of operation of compilers and interpreters, and the role it plays within the context of computer science engineering. The concepts of language processors also find application outside the field of compilation. Some of these fields are design of programming languages and programming aid tools, processing of text files with structured information, symbolic calculation, etc.

Currently, compilation is considered a classical area of informatics, which has a deep theoretical base and a systematized design process. The future of language processors is linked to the development of their multiple application areas.

Therefore, a deep knowledge of the principles and techniques of language processors is fundamental in the computer science curricula.

Contents

Unit 1. Structure of compilers and interpreters

- 1.1 Introduction
- 1.2 Steps in the translation process
- 1.3 Interpretation
- 1.4 Real structure of compilers and interpreters

Unit 2. Lexical Analysis

- 2.1 Introduction
- 2.2 Lexical categories
- 2.3 Lexical categories specification
- 2.4 Finite state automata
- 2.5 Lexical analyser implementation
- 2.6 Applications of lexical analyser

Unit 3. Syntax Analysis

- 3.1 Introduction
- 3.2 Context-free grammars
- 3.3 Some constructions of the programming languages
- 3.4 Extensions of context-free grammars
- 3.5 Top-down parsing
- 3.6 Bottom-up parsing
- 3.7 Syntax analyser implementation

Unit 4. Semantic Analysis

- 4.1 Introduction
- 4.2 Syntax driven translation
- 4.3 abstract syntax tree

4.4 Semantic checks

4.5 Interpretation

Unit 5. Intermediate Code Generation

5.1 Introduction

5.2 Intermediate code

5.3 Real memory organization and management

5.4 Code generation for expressions

5.5 Code generation for control structures

5.6 Code generation for function calls

5.7 Machine code generation

Teaching dispositions

This subject is taught using different teaching methods such as lectures, seminars, working groups for problem solving, and practice sessions where students must tackle the development of a project applying the knowledge acquired in the theoretical sessions.

Evaluation

The course is divided into 3 parts: theory, exercise and a project. For the ordinary and extraordinary, calls complete parts approved previously (in the continuous assessment or the ordinary call respectively) will be saved.
The global mark shall be calculated considering the weight of each part (shown in parentheses).

- Theory (35%) There will be 2 controls (progress tests) throughout the course. For the extraordinary call, a final exam will be scheduled to recover the parts not passed during the course.
- Exercise (20%) Students will be divided in groups to do and send some exercise of each unit.
- project (45%). Students, divided in the same group as exercise task, will implement a compiler for a small programming language.

Readings and other material

- Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman: Compilers. Principles, Techniques and Tools. Addison-Wesley,(2nd. Edition), 2006
- Jean Paul Tremblay and Paul G. Sorenson: The theory and practice of compiler writing. McGraw-Hill, 1985.
- Allen I. Holub: Compiler Design in C. Prentice-Hall, 1990
- Keith D. Cooper and Linda Torczon: Engineering a Compiler. Morgan Kaufmann, 2004.
- Thomas Pittman and James Peter. The Art of Compiler Design. Prentice-Hall, 1992.

Audit and Security Management

Objective

The goal of this subject is to understand key concepts about audit and security of information systems from the governance point of view. Then, this subject is focused on those standards that allows to manage the information system in a secure manner emphasizing on ISO27000 family and other related norms belonging to the National Institute of Standards and Technology.

Contents

Theoretical Contents

Unit 1. Introduction to audit and security

Unit 2. Governance, Risk Management and Compliance: COBIT, COSO, etc.

Unit 3. Introduction to ISO27000

3.1. Identifying information system assets and security requirements

3.2. Risk appetite. Risk matrix.

3.3. Selecting controls

Unit 4. Business Continuity Plan and Disaster Recovery Plan

Unit 5. Communication skills and elaborating an awareness campaign

Practical Contents

Practice 1. Cybersecurity

Practice 2. Introduction to Magerit and Pilar

Practice 3. Establishing an Information security management systems

Practice 4. Elaborating a Disaster Recovery Plan for the School of Computing Engineering in Albacete

Teaching dispositions

In the course the following methods of instruction are used: lectures, practical sessions and additional resources.

The slides will be in English as well as the practices bulletins. Most of the additional resources are in English in nature.

Evaluation

The evaluation consists of three parts:

- Theoretical part: an exam in English will be carried out for foreigner students in virtual platform.
- Practical part: the students are evaluated during the practices in live.
- A final report and presentation: The foreigner students will develop a report of the whole practices (Establishing an Information security management systems and DRP) in English as well as the presentation of this project.

Readings and other material

- ISO/IEC 27000:2016. Information technology -- Security techniques -- Information security management systems -- Overview and vocabulary
- ISO/IEC 27001:2013. Information technology -- Security techniques -- Information security management systems – Requirements.

- ISO/IEC 27002:2013. Information technology -- Security techniques -- Code of practice for information security controls.
- ISO/IEC 27005:2011 Preview
- Information technology -- Security techniques -- Information security risk management
- ISO 31000 - Risk management.
- IEC 31010:2009. Risk management -- Risk assessment techniques.
- Magerit tutorial in English.
- NIST Special Publication 800-100. Information Security Handbook: A Guide for Managers.
- NIST Special Publication 800-30 Risk Management Guide for Information Technology Systems.
- NIST Special Publication 800-34 Rev. 1 Contingency Planning Guide for Federal Information Systems.
- NIST Special Publication 800-39 Managing Information Security Risk Organization, Mission, and Information System View.
- NIST Special Publication 800-53A Guide for Assessing the SecurityRevision 1 Controls in Federal Information Systems and Organizations Building Effective Security Assessment Plans.
- Developing the IT Audit Plan. K. Rehage, S. Hunt and F. Nikitin. The Institute of Internal Auditors.
- Auditing Information Systems. J. J. Champlain. John Wiley and Sons.

escuela superior de ingeniería informática

UCLM

UNIVERSIDAD DE CASTILLA-LA MANCHA