Teaching Guide				
Identifying Data				2019/20
Subject (*)	Design of Information Systems		Code	614502007
Study programme	Mestrado Universitario en Enxeñaría Informática (plan 2012)			
Descriptors				
Cycle	Period	Year	Type	Credits
Official Master's Degree	1st four-month period	First	Obligatory	6
Language	GalicianEnglish			
Teaching method	Face-to-face			
Prerequisites				
Department	Ciencias da Computación e Tecnoloxías da InformaciónComputación			
Coordinador	Sanchez Penas, Juan Jose	E-mail	juan.jose.sanchez.penas@udc.es	
Lecturers	Sanchez Penas, Juan Jose	E-mail	juan.jose.sanchez.penas@udc.es	
Web	https://moodle.udc.es/course/view.php?id=32116			
General description	We will review advanced concepts related to all the aspects of software design, including design and architectural patterns, component-based design, design quality, software evolution, metrics and software complexity or software accessibility. We will focus on consolidating those concepts by studying complex real world projects from a professional perspective.			

Study programme competences / results	
Code	Study programme competences / results
A4	Capacidade para modelar, deseñar, definir a arquitectura, implantar, xestionar, operar, administrar e manter aplicacións, redes, sistemas, servizos e contidos informáticos.
A14	Capacidade para conceptualizar, deseñar, desenvolver e avaliar a interacción persoa?ordenador de produtos, sistemas, aplicacións e servizos informáticos.
B1	Capacidade de resolución de problemas.
B2	Traballo en equipo.
B3	Capacidade de análise e síntese.
B4	Capacidade para organizar e planificar.
B5	Habilidades de xestión da información.
B6	Toma de decisións.
B7	Preocupación pola calidade.
B8	Capacidade de traballar nun equipo interdisciplinar.
B9	Capacidade para xerar novas ideas (creatividade).
B10	Capacidade para proxectar, calcular e deseñar produtos, procesos e instalacións en todos os ámbitos da enxeñaría informática
B13	Capacidade para o modelado matemático, cálculo e simulación en centros tecnolóxicos e de enxeñaría de empresa, particularmente en tarefas de investigación, desenvolvemento e innovación en todos os ámbitos relacionados coa Enxeñaría en Informática
B14	Capacidade para a elaboración, planificación estratéxica, dirección, coordinación e xestión técnica e económica de proxectos en todos os ámbitos da Enxeñaría en Informática seguindo criterios de calidade e ambientais
B17	Capacidade para a aplicación dos coñecementos adquiridos e de resolver problemas en contornas novas ou pouco coñecidos dentro de contextos máis amplos e multidisciplinares, sendo capaces de integrar estes coñecementos
B21	Posuír e comprender coñecementos que acheguen unha base ou oportunidade de ser orixinais no desenvolvemento e/ou aplicación de ideas, a miúdo nun contexto de investigación
B22	Que os estudantes saiban aplicar os coñecementos adquiridos e a súa capacidade de resolución de problemas en contornas novas ou pouco coñecidos dentro de contextos máis amplos (ou multidisciplinares) relacionados coa súa área de estudo
B23	Que os estudantes sexan capaces de integrar coñecementos e enfrontarse á complexidade de formular xuízos a partir dunha información que, sendo incompleta ou limitada, inclúa reflexións sobre as responsabilidades sociais e éticas vinculadas á aplicación dos seus coñecementos e xuízos
B24	Que os estudantes saiban comunicar as súas conclusións, e os coñecementos e razóns últimas que as sustentan, a públicos especializados e non especializados dun modo claro e sen ambigüidades

B25	Que os estudantes posúan as habilidades de aprendizaxe que lles permitan continuar estudando dun modo que haberá de ser en gran medida autodirixido ou autónomo
C1	Expresarse correctamente, tanto de forma oral coma escrita, nas linguas oficiais da comunidade autónoma.
C6	Valorar criticamente o coñecemento, a tecnoloxía e a información dispoñible para resolver os problemas cos que deben enfrontarse.

Learning outcomes			
Learning outcomes	Study programme competences / results		
Understand and know how to design information systems using patterns and following quality principles	AJ4 AJ14	BJ1 BJ2 BJ3 BJ4 BJ5 BJ6 BJ7 BJ8 BJ9 BJ10 BJ13 BJ14 BJ17 BC1 BC2 BC3 BC4 BC5	CJ1 CJ6

Contents	
Topic	Sub-topic
Introduction to advanced software design	Importance of software design Software design and software development processes and methodologies Design and architectural patterns, component-based design Software evolution, design quality, metrics and software complexity Software accessibility Real world examples of complex software design
Advanced concepts of software design	Languages and tools used for software design Design patterns Architectural patterns User interface and User experience patterns Introduction to software refactoring and evolution
Advanced concepts of quality in software design	Software and design quality Metrics and software complexity Evaluation and verification of software systems
Advanced concepts of software accessibility	Importance of software accessibility Software accessibility and software design Software accessibility standards Tools and technologies for software accessibility Case studies of software accessibility

Real world case studies	Overview of some well known, complex software systems Software design in industry-used open source projects In depth analysis of the design, tools, quality and accessibility in several open source projects (e.g. WebKit, GNOME&KDE, Linux, MeeGo/Tizen, etc.)

Planning				
Methodologies / tests	Competencies / Results	Teaching hours (in-person & virtual)	Student?s personal work hours	Total hours
Guest lecture / keynote speech	A4 B7 B10 B14 B17	10	15	25
Case study	A14 B2 B5 B6 B13	10	20	30
Objective test	B1 B3	5	0	5
Workshop	B21 C6	10	20	30
Workbook	B24 B25	0	10	10
Laboratory practice	B4 B8 B9	10	20	30
Events academic / information	B23	0	8	8
Online forum	B22 C1	0	10	10
Personalized attention		2	0	2
(*)The information in the planning table is for guidance only and does not take into account the heterogeneity of the students.				

Methodologies	
Methodologies	Description
Guest lecture / keynote speech	We will invite relevant engineers and managers from the IT industry, in order to give the students guest lectures aligned with the contents of the course.
Case study	We will review real world projects and discuss how the theoretical content that we have studied in the lectures is applied there. We plan to focus mostly on open source projects, as we have full access to the design material and source code.
Objective test	Written exam, where the student will need to show both the theoretical knowledge acquired, and the capacity to resolve practical problems.
Workshop	Practical discussion, analysis and design sessions, with the students organized in groups, supervised by the teacher.
Workbook	The teacher will provide the student with relevant book chapters and articles, related to the theoretical content of the lectures, and the student will need to do a critical read of them, and prepare an executive summary that will be either reviewed by the teacher or by the whole classroom, depending on the case.
Laboratory practice	Practical design and coding exercises, with the students organized in groups, supervised by the teacher.
Events academic / information	We will complement the theoretical and practical lectures with attendance (either onsite or virtually) to conferences related to software design and development.
Online forum	All the topics discussed in lectures, workshops and practical lab time will have a follow up in the virtual forums. We will try to stimulate the discussion there and open new topics proposing links to extra content so that the students can complement their knowledge in the topics they have personal interest in.

Personalized attention	
Methodologies	Description
Online forum Workbook Laboratory practice	The personal attention to the students includes, in this case, not only the classical supervision time (tutorship), or the virtual help using the online resources, but also the following actions: - We will follow constantly the work of the student in all the supervised tasks that will be proposed along the duration of the subject. - Assessment of the results obtained in the practical assignments developed by the student. - Constant communication with the goal of solving the problems found by the student to understand the contents discussed in the lectures or the difficulties of the tasks proposed by the teacher.

Assessment			
Methodologies	Competencies / Results	Description	Qualification
Objective test	B1 B3	Written exam with 3 parts: short theoretical questions, more practical questions where the students can elaborate a bit more the answers to the problems explained, and a specific complete design challenge.	50
Workshop	B21 C6	The assesment of practical tasks in workshops will be continuous along the course, and will be based on a final presentation to the teacher. We will consider as part of the assesment the following aspects: - Capacity to work as part of a group. - Personal capacity to carry out work and explain it. - Capacity to cover all the goals of the task. - Capacity to apply the knowledge acquired during the theoretical lessons. - Critical thinking and capacity to innovate and find solutions to problems. - Capacity to deliver the tasks on time.	50

Assessment comments
The summary of the qualification distribution is that we will obtain 50% of the marks from the written exam and 50% from a collection of workshops and practical activities that will be carried out during the course. The student is required to pass both the written exam and the workshops and practical activities in order to pass the subject. Those students who are attending the course part time, or those who have any circumstance that prevents them from attending all the lectures, should contact the teacher to discuss alternatives for the assessment.

Sources of information	
Basic	Gamma, E.; Helm, R.; Johnson, R. y Vlissides J. (1996). Design Patterns: Elements of Reusable Object-oriented Software. Addison Wesley Martin Fowler with contributions by Kent Beck, John Brant, William Opdyke and Don Roberts. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999. Michael Jackson. Problem Analysis and Structure. In Proceedings of NATO Summer School, Marktoberdorf, August 2000 (in publication). Available here. Michael Jackson. Problem Frames: Analyzing and Structuring Software Development Problems. Addison Wesley, 2001. G. Polya. How to Solve It. 2nd ed., Princeton University Press, 1957. Diomidis Spinellis. Code Quality: The Open Source Perspective. Addison Wesley, Boston, MA, 2006. Stephen H. Kan. Metrics and Models in Software Quality Engineering. Addison-Wesley, Boston, MA, second edition, 2002. Henry, Shawn Lawton. Integrating Accessibility Throughout Design. Lulu.com. February 2007Gamma, E.; Helm, R.; Johnson, R. y Vlissides J. (1996). Design Patterns: Elements of Reusable Object-oriented Software. Addison WesleyMartin Fowler with contributions by Kent Beck, John Brant, William Opdyke and Don Roberts. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999.Michael Jackson. Problem Analysis and Structure. In Proceedings of NATO Summer School, Marktoberdorf, August 2000 (in publication). Available here.Michael Jackson. Problem Frames: Analyzing and Structuring Software Development Problems. Addison Wesley, 2001.G. Polya. How to Solve It. 2nd ed., Princeton University Press, 1957.Diomidis Spinellis. Code Quality: The Open Source Perspective. Addison Wesley, Boston, MA, 2006.Stephen H. Kan. Metrics and Models in Software Quality Engineering. Addison-Wesley, Boston, MA, second edition, 2002.Henry, Shawn Lawton. Integrating Accessibility Throughout Design. Lulu.com. February 2007

Complementary	Rumbaugh, J.; Jacobson, I. y Booch, J. (2004). The Unified Modeling Language Reference Manual. Addison Wesley Booch J.; Rumbaugh J. y Jacobson I. (2005). The Unified Modeling Language User Guide. Addison Wesley Page-Jones, M. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall PTR Cooper J. (2000). Java Design Patterns: A Tutorial. Addison Wesley Stevens, P. y Pooley, R. (1999). Using UML. Software Engineering with Objects and Components. Addison Wesley Ho-Won Jung, Seung-Gweon Kim, and Chang-Sin Chung. Measuring software product quality: A survey of ISO/IEC 9126. IEEE Software, 21(5):10?13, September/October 2004. Omar Alshathry, Helge Janicke, "Optimizing Software Quality Assurance," compsacw, pp. 87?92, 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010. Robert L. Glass. Building Quality Software. Prentice Hall, Upper Saddle River, NJ, 1992. Roland Petrasch, "The Definition of? Software Quality?: A Practical Approach", ISSRE, 1999Rumbaugh, J.; Jacobson, I. y Booch, J. (2004). The Unified Modeling Language Reference Manual. Addison WesleyBooch J.; Rumbaugh J. y Jacobson I. (2005). The Unified Modeling Language User Guide. Addison WesleyPage-Jones, M. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall PTRCooper J. (2000). Java Design Patterns: A Tutorial. Addison WesleyStevens, P. y Pooley, R. (1999). Using UML. Software Engineering with Objects and Components. Addison WesleyHo-Won Jung, Seung-Gweon Kim, and Chang-Sin Chung. Measuring software product quality: A survey of ISO/IEC 9126. IEEE Software, 21(5):10?13, September/October 2004.Omar Alshathry, Helge Janicke, "Optimizing Software Quality Assurance," compsacw, pp. 87?92, 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010.Robert L. Glass. Building Quality Software. Prentice Hall, Upper Saddle River, NJ, 1992.Roland Petrasch, "The Definition of? Software Quality?: A Practical Approach", ISSRE, 1999

Recommendations
Subjects that it is recommended to have taken before
Subjects that are recommended to be taken simultaneously
Information Systems Analysis/614502006
Subjects that continue the syllabus
Project Management/614502002 Quality, Information Security and Computing Audit/614502003 Architectures and Mobile Platforms/614502005 Business Practice/614502011 Final Project/614502012
Other comments

(*)The teaching guide is the document in which the URV publishes the information about all its courses. It is a public document and cannot be modified. Only in exceptional cases can it be revised by the competent agent or duly revised so that it is in line with current legislation.