



Guía Docente				
Datos Identificativos				2020/21
Asignatura (*)	Deseño de sistemas de información	Código	614502007	
Titulación				
Descritores				
Ciclo	Período	Curso	Tipo	Créditos
Mestrado Oficial	1º cuatrimestre	Primeiro	Obrigatoria	6
Idioma	GalegoInglés			
Modalidade docente	Híbrida			
Prerrequisitos				
Departamento	Ciencias da Computación e Tecnoloxías da InformaciónComputación			
Coordinación	Sanchez Penas, Juan Jose	Correo electrónico	juan.jose.sanchez.penas@udc.es	
Profesorado	Sanchez Penas, Juan Jose	Correo electrónico	juan.jose.sanchez.penas@udc.es	
Web	https://moodle.udc.es/course/view.php?id=32116			
Descrición xeral	<p>Revisaremos conceptos avanzados relacionados con todos os aspectos do deseño software, incluíndo patróns de deseño e arquitectura, deseño orientado a componentes, calidade no deseño, evolución do software, métricas e complexidade software, ou accesibilidade. O obxectivo será consolidar eses conceptos estudando proxectos do mundo real dende unha perspectiva profesional. O idioma principal da asignatura será o inglés.</p> <p>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.</p>			



Plan de continxencia	<p>1. Modificacións nos contidos</p> <p>Non se realizarán cambios nos contidos.</p> <p>2. Metodoloxías</p> <p>*Metodoloxías docentes que se manteñen</p> <p>Todas. As metodoloxías utilizadas son compatibles coa docencia non presencial.</p> <p>*Metodoloxías docentes que se modifican</p> <p>Ningunha.</p> <p>3. Mecanismos de atención personalizada ao alumnado</p> <p>- Moodle: Aloxamento dos contidos do curso, xestión das tarefas, e foros de debate en grupo.</p> <p>- Teams, correo electrónico: Comunicación permanente e directa profesor-alumno.</p> <p>4. Modificacións na avaliación</p> <p>Non se considera necesario ningún cambio significativo nos criterios de avaliación utilizados.</p> <p>*Observacións de avaliación:</p> <p>O único que se introduce é a posibilidade de eliminar a presencialidade da proba obxectiva e da presentación traballos prácticos, usando alternativas online.</p> <p>5. Modificacións da bibliografía ou webgrafía</p> <p>Ningunha. Todas referencias máis relevantes están disponibles online.</p>
-----------------------------	---

Competencias / Resultados do título	
Código	Competencias / Resultados do título

Resultados da aprendizaxe	
Resultados de aprendizaxe	Competencias / Resultados do título



Comprender e saber deseñar sistemas de información mediante patróns e seguindo pautas de calidade.	AP4 AP14	BP1 BP2 BP3 BP4 BP5 BP6 BP7 BP8 BP9 BP10 BP13 BP14 BP17 BM1 BM2 BM3 BM4 BM5	CP1 CP6
--	-------------	--	------------

Contidos	
Temas	Subtemas
Introducción ao deseño de software avanzado	<ul style="list-style-type: none"> Importancia do deseño software Metodoloxías e procesos de deseño e desenvolvemento software Patróns de deseño e arquitectura, deseño orientado a componentes Evolución do software, calidade do deseño, métricas e complexidade do software Accesibilidade do software Exemplos do mundo real de deseño software complexo
Conceptos avanzados de deseño software	<ul style="list-style-type: none"> Linguaxes e ferramentas usadas para o deseño software Patróns de deseño Patróns de arquitectura Patróns de interfaz de usuario e experiencia de usuario Introducción á refactorización e a evolución do software
Conceptos avanzados de calidade no deseño software	<ul style="list-style-type: none"> Software e calidade no deseño Métricas e complexidade do software Evaluación e verificación de sistemas software
Conceptos avanzados de accesibilidade do software	<ul style="list-style-type: none"> Importancia da accesibilidade do software Accesibilidade do software e deseño software Standards de accesibilidade no software Ferramentas e tecnoloxías para a accesibilidade do software Casos de estudo de accesibilidade do software
Casos de estudo do mundo real	<ul style="list-style-type: none"> Revisión de algúns sistemas software populares e complexos Deseño software en proxectos de software libre utilizados na industria Análise en profundidade do deseño, as ferramentas, a calidade e a accesibilidade en varios proxectos de software libre (por exemplo WebKit, GNOME&KDE, Linux, MeeGo/Tizen, etc.)

Planificación



Metodoloxías / probas	Competencias / Resultados	Horas lectivas (presenciais e virtuais)	Horas traballo autónomo	Horas totais
Sesión maxistral	A4 B7 B10 B14 B17	10	15	25
Estudo de casos	A14 B2 B5 B6 B13	10	20	30
Proba obxectiva	B1 B3	5	0	5
Obradoiro	B21 C6	10	20	30
Lecturas	B24 B25	0	10	10
Prácticas de laboratorio	B4 B8 B9	10	20	30
Eventos científicos e/ou divulgativos	B23	0	8	8
Foro virtual	B22 C1	0	10	10
Atención personalizada		2	0	2

*Os datos que aparecen na táboa de planificación son de carácter orientativo, considerando a heteroxeneidade do alumnado

Metodoloxías	
Metodoloxías	Descrición
Sesión maxistral	Convidaremos enxeñeiros e managers relevantes da industria das TIC, co obxectivo de impartir sesións maxistras que complementen os contidos formativos da asignatura.
Estudo de casos	Revisaremos proxectos reais e discutiremos o xeito no que o contido teórico estudado na asignatura é aplicado neles. Enfocáremosnos principalmente en proxectos de software libre, xa que temos acceso a todo o código fonte e material de deseño.
Proba obxectiva	Exame escrito, no que o estudante terá que amosar tanto os coñecementos teóricos adquiridos como a capacidade para resolver problemas prácticos
Obradoiro	Sesións de análise, deseño e discusión práctica, cos estudantes organizados en grupos, supervisados polo profesor.
Lecturas	O profesor proporcionará aos estudantes artigos e capítulos de libros relevantes, relacionados co contido teórico do curso, e o estudante terá que facer unha lectura crítica dos mesmos e preparar un resumo que será revisado polo profesor ou por toda a clase, dependendo do caso.
Prácticas de laboratorio	Exercicios prácticos de deseño e desenvolvemento, cos estudantes organizados en grupos, supervisados polo profesor.
Eventos científicos e/ou divulgativos	Como complemento das clases teóricas e prácticas, recomendarase aos alumnos a asistencia (en persoa ou en remoto) a conferencias relacionadas co deseño e desenvolvemento de software.
Foro virtual	Todos os temas estudados nas clases, obradoiros e tempo práctico de laboratorio terán a súa continuidade nos foros online. Trataráse de estimular a conversa neles, e de abrir novos temas de conversa propoñendo ligazóns extra que complementen o coñecemento dos alumnos en temas colaterais que podan ser do seu interese.

Atención personalizada	
Metodoloxías	Descrición
Foro virtual Lecturas Prácticas de laboratorio	A atención persoal ao estudante inclúe, neste caso, non só o clásico tempo de titorías, ou o apoio virtual usando os recursos online, senón as seguintes accións: - Seguirase constantemente o traballo do estudante nas tarefas supervisadas que serán propostas ao longo da duración da materia. - Avaliación crítica dos resultados obtidos nos traballos prácticos desenvolvidos polo estudante. - Comunicación constante co obxectivo de resolver os problemas atopados polo estudante para comprender os contidos expostos nas clases ou as dificultades das tarefas propostas polo profesor.

Avaliación			
Metodoloxías	Competencias / Resultados	Descrición	Cualificación



Proba obxectiva	B1 B3	Exame por escrito con 3 partes: preguntas teóricas curtas, preguntas máis prácticas na que os estudantes podan elaborar con máis detenimento as respostas ás cuestións prantexadas, e un problema real específico de deseño de software.	50
Obradoiro	B21 C6	A avaliación das tarefas prácticas en obradoiros será continua ao longo do curso, e basearase nunha presentación final ao profesor. Consideraranse na avaliación os seguintes aspectos: <ul style="list-style-type: none"> - Capacidade para traballar en grupo. - Capacidade persoal para facer o traballo e explicalo. - Capacidade para axustarse aos obxectivos das tarefas. - Capacidade para aplicar coñecemento adquirido durante as clases teóricas. - Pensamento crítico e capacidade para innovar e atopar solucións a problemas. - Capacidade para entregar as tarefas a tempo. 	50

Observacións avaliación

O resumo da distribución de pesos nas avaliacións é o seguinte: o 50% da nota derivará do exame escrito, e o outro 50% dun conxunto de traballos prácticos que serán realizados ao longo do curso. É necesario ter unha nota mínima de aprobado tanto no exame escrito como no conxunto de traballos prácticos. Aqueles estudantes con matrícula a tempo parcial ou calquer circunstancia que impida a asistencia as clases, deber an contactar cos docentes para determinar alternativas ao seguimento e a avaliación da materia.

Fontes de información

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



Bibliografía complementaria	<p>Rumbaugh, J.; Jacobson, I. y Booch, J. (2004). The Unified Modeling Language Reference Manual. Addison Wesley</p> <p>Booch J.; Rumbaugh J. y Jacobson I. (2005). The Unified Modeling Language User Guide. Addison Wesley</p> <p>Page-Jones, M. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall PTR</p> <p>Cooper J. (2000). Java Design Patterns: A Tutorial. Addison Wesley</p> <p>Stevens, P. y Pooley, R. (1999). Using UML. Software Engineering with Objects and Components. Addison Wesley</p> <p>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.</p> <p>Omar Alshathry, Helge Janicke, "Optimizing Software Quality Assurance," compsocw, pp. 87?92, 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010.</p> <p>Robert L. Glass. Building Quality Software. Prentice Hall, Upper Saddle River, NJ, 1992.</p> <p>Roland Petrasch, "The Definition of? Software Quality?: A Practical Approach", ISSRE, 1999</p> <p>Rumbaugh, J.; Jacobson, I. y Booch, J. (2004). The Unified Modeling Language Reference Manual. Addison Wesley</p> <p>Booch J.; Rumbaugh J. y Jacobson I. (2005). The Unified Modeling Language User Guide. Addison Wesley</p> <p>Page-Jones, M. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall PTR</p> <p>Cooper J. (2000). Java Design Patterns: A Tutorial. Addison Wesley</p> <p>Stevens, P. y Pooley, R. (1999). Using UML. Software Engineering with Objects and Components. Addison Wesley</p> <p>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.</p> <p>Omar Alshathry, Helge Janicke, "Optimizing Software Quality Assurance," compsocw, pp. 87?92, 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010.</p> <p>Robert L. Glass. Building Quality Software. Prentice Hall, Upper Saddle River, NJ, 1992.</p> <p>Roland Petrasch, "The Definition of? Software Quality?: A Practical Approach", ISSRE, 1999</p>
------------------------------------	---

Recomendacións

Materias que se recomenda ter cursado previamente

Materias que se recomenda cursar simultaneamente

Análise de sistemas de información/614502006

Materias que continúan o temario

Dirección de proxectos/614502002

Calidade, seguridade e auditoría informática/614502003

Arquitecturas e plataformas móbiles/614502005

Prácticas en empresa/614502011

Traballo fin de mestrado/614502012

Observacións

(*)A Guía docente é o documento onde se visualiza a proposta académica da UDC. Este documento é público e non se pode modificar, salvo casos excepcionais baixo a revisión do órgano competente dacordo coa normativa vixente que establece o proceso de elaboración de guías