Facultad de Ciencias

GUÍA DOCENTE DE LA ASIGNATURA

G668 - Métodos de Desarrollo

Grado en Ingeniería Informática
Optativa. Curso 4

Curso Académico 2023-2024

1. DATOS IDENTIFICATIVOS

Título/s	Grado en Ingeniería Informática			Tipología y Curso	Optativa. Curso 4
Centro	Facultad de Ciencias				
Módulo / materia	MATERIA INGENIERÍA DEL SOFTWARE MENCIÓN EN INGENIERÍA DEL SOFTWARE				
Código y denominación	G668 - Métodos de Desarrollo				
Créditos ECTS	6	Cuatrimestre		Cuatrimestral (1)	
Web	http://moodle.unican.es				
Idioma de impartición	Español	English friendly	No	Forma de impartición	Presencial

Departamento	DPTO. INGENIERÍA INFORMÁTICA Y ELECTRÓNICA
Profesor responsable	PABLO SANCHEZ BARREIRO
E-mail	p.sanchez@unican.es
Número despacho	Facultad de Ciencias. Planta: + 1. DESPACHO PROFESOR (1069)
Otros profesores	JULIO LUIS MEDINA PASAJE JUAN MARIA RIVAS CONCEPCION

2. CONOCIMIENTOS PREVIOS

El alumno deberá haber cursado las asignaturas de Ingeniería del Software II e Ingeniería de Requisitos.

Concretamente, el alumno deberá tener conocimientos básicos de UML y ser capaz de: (1) gestionar versiones de un proyecto software utilizando una herramienta como Git; y, (2) ser capaz de especificar requisitos mediante Historias de Usuario.

3. COMPETENCIAS GENÉRICAS Y ESPECÍFICAS DEL PLAN DE ESTUDIOS TRABAJADAS
Competencias Genéricas
Capacidad de organización y planificación.
Capacidad para argumentar y justificar lógicamente las decisiones tomadas y las opiniones.
Capacidad de trabajo en equipo.
Capacidad de relación interpersonal.
Aprendizaje autónomo.
Capacidad de liderazgo.
Tener motivación por la calidad.
Capacidad de análisis, síntesis y evaluación.
Capacidad de gestión de la información.
Capacidad de resolución de problemas aplicando técnicas de ingeniería.
Adaptación a nuevas situaciones.
Creatividad.
Competencias Específicas
Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.
Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la Ingeniería del Software.
Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.
Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.
Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos.
Competencias Básicas
Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.
Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.

3.1 RESULTADOS DE APRENDIZAJE
- Saber desarrollar software siguiendo alguna metodología.
- Conocer y aplicar alguno de los principales paradigmas del desarrollo de software: guiado por modelos, componentes, aspectos, eventos, etc
- Aplicar las principales técnicas y normas para hacer el plan de un proyecto de desarrollo o mantenimiento de software.
- Comprender los procesos software, y saber modelarlos y especificarlos
- Emplear técnicas y herramientas de gestión de la configuración.
- Utilizar metodologías y técnicas para llevar a cabo proyectos o servicios de mantenimiento o modernización del software.
- Aplicar los principales métodos, técnicas y herramientas para pruebas, verificación y validación del software.
- Utilizar métodos y técnicas para la gestión, control y seguimiento de un proyecto de desarrollo de software.
- Saber realizar una estimación del tamaño, esfuerzo y costes de un proyecto software.
- Realizar una adecuada planificación, análisis y control de los riesgos en un proyecto de desarrollo de software y de los riesgos en un sistema de información en operación.

4. OBJETIVOS
Conseguir que el alumnado sea capaz gestionar configuraciones de productos software utilizando una herramienta adecuada para ello como Git.
Conseguir que el alumnado comprenda el funcionamiento de los procesos de la gestión de las configuración.
Conseguir que el alumnado comprenda con claridad la diferencia entre metodologías rígidas y ágiles.
Conseguir que el alumnado comprenda los fundamentos de las metodologías ágiles.
Conseguir que el alumnado sea capaz de aplicar las principales técnicas utilizadas por las metodologías ágiles.
Conseguir que el alumnado comprenda las similitudes y diferencias entre las principales metodologías ágiles.
Conseguir que el alumnado sea capaz de desarrollar un proyecto sw de mediana escala utilizando una metodología ágil como Scrum.
Conseguir que el alumnado sea capaz de modelar procesos de desarrollo software utilizando un lenguaje específico para tal propósito como SPEM.
Conseguir que el alumnado conozcael funcionamiento de los procesos de mantenimiento.
Conseguir que el alumnado conozca el funcionamiento de la metodología Métrica v3.
Conseguir que el alumnado sea capaz de utilizar esquemas de ramificación profesionales como GitFlow.
Conseguir que el alumnado sea capaz de utilizar esquemas de integración continua.
Conseguir que el alumnado conozca el concepto de DevOps.

5. MODALIDADES ORGANIZATIVAS Y MÉTODOS DOCENTES	
ACTIVIDADES	HORAS DE LA ASIGNATURA
ACTIVIDADES PRESENCIALES	
HORAS DE CLASE (A)	
- Teoría (TE)	20
- Prácticas en Aula (PA)	
- Prácticas de Laboratorio Experimental(PLE)	
- Prácticas de Laboratorio en Ordenador (PLO)	40
- Prácticas Clínicas (CL)	
Subtotal horas de clase	60
ACTIVIDADES DE SEGUIMIENTO (B)	
- Tutorías (TU)	4
- Evaluación (EV)	8
Subtotal actividades de seguimiento	12
Total actividades presenciales (A+B)	72
ACTIVIDADES NO PRESENCIALES	
Trabajo en grupo (TG)	53
Trabajo autónomo (TA)	25
Tutorías No Presenciales (TU-NP)	
Evaluación No Presencial (EV-NP)	
Total actividades no presenciales	78
HORAS TOTALES	150

6. ORGANIZACIÓN DOCENTE

	CONTENIDOS	TE	PA	PLE	PLO	CL	TU	EV	TG	TA	TU-NP	EV-NP	Semana
1	Tema 1. Gestión de la Configuración. Introducción. Terminología. Procesos de Gestión de la Configuración. Integración y Entrega Continua. DevOps. Gestión Avanzada de Versiones con Git. Esquemas de Ramificación: GitFlow.	4,00	0,00	0,00	6,00	0,00	0,00	1,00	0,00	8,00	0,00	0,00	1-2
2	Tema 2. Metodologías Ágiles Metodologías Rígidas y Ágiles. Manifiesto Ágil. Principios Lean. Técnicas Básicas de las Metodologías Ágiles: Historias de Usuario, Desarrollo Dirigido Por Pruebas, Planning Poker, Programación por Pares. Scrum. Otras Metodologías Ágiles: Lean, Kanban, XP.	7,00	0,00	0,00	28,00	0,00	4,00	3,00	60,00	0,00	0,00	0,00	3-10
3	Tema 3. Modelado de Procesos Software Concepto de Proceso Sw. Modelado de Procesos Software: SPEM.	6,00	0,00	0,00	4,00	0,00	0,00	1,00	0,00	7,00	0,00	0,00	11-13
4	Tema 4. Procesos de Desarrollos Estandarizados. Métrica v3. Procesos de Mantenimiento en Métrica v3.	3,00	0,00	0,00	2,00	0,00	0,00	1,00	0,00	5,00	0,00	0,00	14-15
TOTAL DE HORAS		20,00	0,00	0,00	40,00	0,00	4,00	6,00	60,00	20,00	0,00	0,00	

Esta organización tiene carácter orientativo.

TE	Horas de teoría
PA	Horas de prácticas en aula
PLE	Horas de prácticas de laboratorio experimental
PLO	Horas de prácticas de laboratorio en ordenador
CL	Horas de prácticas clínicas
TU	Horas de tutoría
EV	Horas de evaluación
TG	Horas de trabajo en grupo
TA	Horas de trabajo autónomo
TU-NP	Tutorías No Presenciales
EV-NP	Evaluación No Presencial

7. MÉTODOS DE LA EVALUACIÓN

Descripción	Tipología	Eval. Final	Recuper.	%
Actividades Prácticas Evaluables	Evaluación en laboratorio	No	Sí	30,00

Calif. mínima	3,00
Duración	Variable según cada prueba
Fecha realización	Tras finalizar el proyecto integrado (Semana 12 ó 13) y en el periodo oficial de exámenes.
Condiciones recuperación	Repetición, en convocatoria extraordinaria, de aquellas pruebas que no se hayan superado.
Observaciones	Durante el curso se realizarán varias entregas de trabajos y tests prácticos para verificar el grado de adquisición de competencias de los alumnos con respecto a diversas cuestiones concretas. El número de trabajos y tests a realizar, así como sus pesos dentro de la calificación de este apartado, se definirá al comienzo del curso.

Desarrollo de un Proyecto Sw utilizando SCRUM	Trabajo	No	Sí	70,00

Calif. mínima	5,00
Duración	6 semanas
Fecha realización	Semanas 6 a 11 del cuatrimestre.
Condiciones recuperación	Realización en convocatoria extraordinaria de un proyecto similar donde se demuestren las habilidades relacionadas con Scrum.
Observaciones	Los alumnos deberán desarrollar un pequeño proyecto software en Scrum. Los proyectos se desarrollarán en grupos de entre 4-6 alumnos. Se valorará especialmente que el alumno se adhiera a los principios metodológicos propuestos por Scrum. Durante las semanas que dure el desarrollo del proyecto todas las clases serán de laboratorio, no impartiéndose clases ni teóricas ni de problemas en aula. El proyecto se desarrollará de manera integrada con las asignaturas de Calidad y Auditoría , y Procesos de Ingeniería del Software. Dicha evaluación se complementará con una prueba escrita la cual contendrá una serie de preguntas cortas destinadas a verificar que: (1) el alumno conoce y comprende los fundamentos teóricos y metodológicos sobre los cuales se sustenta la práctica; y, (2) el alumno ha particicipado activamente en el desarrollo del proyecto y, por tanto, lo entregado es tanto trabajo suyo como de sus compañeros de equipo.

TOTAL	100,00

Observaciones

En caso de que no se alcance la nota mínima exigida en alguno de los elementos evaluables y la media ponderada de dichos elementos sea superior o igual a 5, la calificación que constará en las actas de la asignatura será de 4.9, Suspenso

En cualquier momento el profesorado podrá aplicar los mecanismos que considere adecuados para verificar que el alumno es realmente el autor del material del que asegura ser autor. La detección de un plagio supondrá el suspenso automático de la asignatura y su notificación a instancias superiores para que éstas adopten las medidas disciplinarias que estimen oportunas.

Criterios de evaluación para estudiantes a tiempo parcial

Un objetivo importante de la asignatura es que los alumnos aprendan a desarrollar proyectos software en equipo utilizando para ello una metodología de desarrollo concreta. Es por tanto necesaria en muchas ocasiones la interacción de un alumno con sus compañeros, siendo esta interacción, además, un elemento evaluable de la asignatura. Por tanto, no existe la posibilidad de realizar el proyecto de manera individualizada, debiéndose integrar los alumnos a tiempo parcial en equipos de desarrollo software. La presencia de los alumnos a tiempo parcial se requiere principalmente entre las semanas 6 a 11 del cuatrimestre. Por tanto, sería deseable, aunque no obligatorio, que los alumnos a tiempo parcial pudiesen asistir durante estas semanas a clase. Si por el motivo que fuese, algún alumno a tiempo parcial no pudiese asistir a clases de manera regular durante esas semanas, se le permitirá trabajar de manera remota y asíncrona con sus compañeros. No obstante, todo alumno deberá acudir a un número mínimo de sesiones presenciales de manera que se puedan realizar de manera adecuada diversas actividades de evaluación previstas en la asignatura. Dado que el calendario de la asignatura está definido con gran precisión, es posible informar al alumno de las sesiones a las que es imprescindible que asista desde el primer día del cuatrimestre, facilitándole así la gestión de su agenda.

8. BIBLIOGRAFÍA Y MATERIALES DIDÁCTICOS

BÁSICA

Mike Cohn. "User Stories Applied:For Agile Software Development". Addison-Wesley. Marzo 2010.

Complementaria

Ian Sommerville, 2005. “Ingeniería del Software”. 9ª Edición, Addison-Wesley. Julio 2010.

Mary Poppendieck and Tom Poppendieck. "Lean Software Development:An Agile Toolkit". Addison-Wesley. Mayo 2013.

Alistair Cockburn. "Agile Software Development: The Cooperative Game". 2ª Edición. Addison-Wesley Professional, 2006

David J. Anderson. "Kanban". Blue Hole Press. Abril 2010.

Object Management Group. "Software & Systems Process Engineering Meta-Model Specification". Estandar formal/2008-04-01 Abril 2008.

Scott Chacon, Ben Straub "Pro Git". 2ª Edición. Apress. Noviembre 2014.

Jeff Sutherland. "Scrum: The Art of Doing Twice the Work in Half the Time". Crown Business. 2014

9. SOFTWARE

PROGRAMA / APLICACIÓN	CENTRO	PLANTA	SALA	HORARIO
EPF Composer	Facultad de Ciencias			
Git	Facultad de Ciencias			
SourceTree	Facultad de Ciencias			
Scrumdesk	Facultad de Ciencias			

10. COMPETENCIAS LINGÜÍSTICAS

- ☑ Comprensión escrita
- ☐ Comprensión oral
- ☐ Expresión escrita
- ☐ Expresión oral
- ☐ Asignatura íntegramente desarrollada en inglés

Observaciones

La asignatura hará uso de abundante material escrito y bibliografía sólo disponible en inglés.