TecnoCampus Escola Superior Politècnica
Centre adscrit a la upf. Universitat Pompeu Fabra Barcelona

DOBLE GRADO EN INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN/GRADO EN DISEÑO Y PRODUCCIÓN DE VIDEOJUEGOS

107331 - INGENIERÍA DEL SOFTWARE III

Información general

- Tipo de asignatura : Obligatoria
- Coordinador : Adso Fernández Baena
- Curso: Tercero
- Trimestre: Tercero
- Créditos: 4
- Profesorado:
  - David Ródenas Picó <drodenas@tecnocampus.cat>
  - Josep Roure Alcobé <roure@tecnocampus.cat>

Idiomas de impartición

- Catalán

El idioma de impartición principal en clase es el Catalán.

Sin embargo:

- todo el material de la asignatura disponible está en inglés,
- las herramientas usadas y sus documentaciones está en inglés,
- todos los trabajos y todas las entregas que se hacen deben ser en inglés.
(Esto incluye código, comentarios, documentación, interfaces, etc.)

Competencias que se trabajan

Básica

- B2_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
- B4_Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
- B5_Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

Común

- CIN8_Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
- CIN16_Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

Específica

-

EIS1_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.

- EIS2_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.
- EIS4_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.
- EIS5_Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.
- EIS6_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.
- ESI3_Capacidad para participar activamente en la especificación, diseño, implementación y mantenimiento de los sistemas de información y comunicación.

Transversal

- T1_Que los estudiantes conozcan un tercer idioma, que será preferentemente inglés, con un nivel adecuado de forma oral y por escrito y de acuerdo con las necesidades que tendrán las graduadas y los graduados en cada titulación

Descripción

La asignatura de Ingeniería del Software III del tercer trimestre de tercer curso, es la última de las tres asignaturas denominadas Ingeniería del Software. Su impartición está pensada en dedicar 3 ECTS en la parte de teoría y 1 ECTS a practicar los conceptos expuestos en teoría.

En esta asignatura se explicará nuevos Patrones de Diseño de Software y temas relacionados con la Ingeniería de Requisitos, poniendo el énfasis en la modelización con Diagramas de Actividad y especificaciones formales con OCL.

Otro de los temas abordados en esta asignatura será el Testing, haciendo una extensión de las técnicas ya vistas en Ingeniería del Software 1, introduciendo diferentes estrategias y la depuración. El uso del testing para mejorar la calidad de codi i su mantenibilidad es clave.

Esta asignatura dispone de recursos metodológicos y digitales para hacer posible su continuidad en modalidad no presencial en el caso de ser necesario por motivos relacionados con el Covid-19. De esta forma se asegurará la consecución de los mismos conocimientos y competencias que se especifican en este plan docente.

Resultados de aprendizaje

En nivel general, esta asignatura contribuye a los siguientes resultados del aprendizaje especificades en la materia a la que pertenece (Ingeniería del Software)

- Utilizar de forma apropiada teorías, procedimientos y herramientas en el desarrollo profesional de la ingeniería informática en todos sus ámbitos (especificación, diseño, implementación, despliegue -implantación- y evaluación de productos) de manera que se demuestre la comprensión de los compromisos adoptados en las decisiones de diseño.
- Demostrar conocimiento de la dimensión ética en la empresa: la responsabilidad social y corporativa en general y, en particular, las responsabilidades civiles y profesionales del ingeniero en informática.
- Usar las herramientas de un entorno de desarrollo de software para crear y desarrollar aplicaciones.
- Demostrar conocimiento y saber aplicar las técnicas apropiadas para modelar y analizar los diferentes tipos de decisiones.
- Gestionar y resolver los problemas y conflictos gracias a la capacidad de generar alternativas o escenarios de futuro convenientemente analizados, integrando los aspectos de incertidumbre y los múltiples objetivos a considerar.
- Controlar versiones y configuraciones del proyecto.
- Especificar, diseñar, implementar, gestionar y mantener sistemas y servicios software complejos y/o críticos.
- Controlar la calidad y diseñar pruebas en la producción de software.
- Identificar tecnologías actuales y emergentes y evaluar si son aplicables, y en qué medida, para satisfacer las necesidades de los usuarios.
- Diseñar soluciones que integren tecnologías de hardware, software y comunicaciones (y capacidad de desarrollar soluciones específicas de software de sistemas) para sistemas distribuidos y dispositivos de computación ubicua.

En un nivel más concreto, al acabar esta asignatura el estudiante tendrá que ser capaz de:

RA1: Reconocer los diversos participantes en la adquisición de requisitos y definir estrategias Lean de captación de requisitos.
RA2: Reconocer y saber leer diagramas de actividad y restricciones OCL.
RA3: Conocer el Agile Software Development y como liga con el concepto de Refactor y Lean.
RA4: Hacer pruebas del software y aplicar metodologías TDD, con propiedades FIRST y AAA.
RA5: Leer y escribir diagramas UML y más concretamente saber interpretar las direcciones del acoblamiento con las flechas.
RA6: Entender y aplicar los patrones SOLID, la ley de Demeter y la Inyección de Dependencias.
RA7: Entender y aplicar los patrones Command y Observador, y el patrón arquitectónico MVC.
RA8: Entender el concepto de datos derivados, materializados y computados, y como se relacionan con MVC.
RA9: Crear una aplicación MVC.

Metodología de trabajo

Todos los conceptos teóricos de la materia se tratarán en las clases de teoría (grupos grandes) de la asignatura. En estas clases se introducen los conceptos básicos propuestos en el temario, mostrando su aplicación con ejercicios resueltos por el docente. Se recomienda que antes de cada sesión los estudiantes se lean el material publicado en la plataforma virtual. En las clases se pedirá la participación de los estudiantes de manera individual o en grupo, para resolver diferentes problemas propuestos con o sin anticipación. Estas actividades, que por su naturaleza de optatividad y brevedad no aparecen reflejadas en este documento, servirán al estudiante como instrumento de autoevaluación de la consecución de los contenidos de la materia y podrán ser utilizados por parte del docente para tomar decisiones sobre la calificación final del estudiante, pero nunca en detrimento de la calificación numérica calculada según el sistema de calificación antes indicado.

Los conceptos de carácter más práctico serán trabajados en grupos pequeños (de laboratorio) donde se presentan trabajos de complejidad media, que requieren la aplicación de los conocimientos adquiridos en las clases más teóricas. En estas sesiones se darán las herramientas adecuadas para resolver las actividades programadas pero se espera que estas alarguen desde el punto de vista temporal, más allá de las horas de laboratorio y que, en consecuencia, los estudiantes deban finalizar durante el tiempo de aprendizaje autónomo.

Los estudiantes deberán asistir a todas las clases con un ordenador portátil con la capacidad de ejecutar el software apropiado para la asignatura. Los docentes impartidores informarán de qué es este software y cómo se puede obtener.

Se pondrá a disposición de los estudiantes actividades de tipo totalmente opcional que le ayuden a preparar y a prepararse para las de carácter obligatorio.

Normas de realización de las actividades:
Para cada actividad, los docentes informarán de las normas y condiciones particulares que las rijan. Las actividades unipersonales presuponen el compromiso del estudiante de realizarlas de manera individual. Se considerarán suspendidas todas aquellas actividades en que el estudiante no se ajuste a este compromiso, independientemente de su papel (origen o destino). Igualmente, las actividades que se deban realizar en grupos presuponen el compromiso por parte de los estudiantes que lo integran de realizarlas en el seno del grupo. Se considerarán suspendidas todas aquellas actividades en las que el grupo no haya respetado este compromiso con independencia de su papel (origen o destino). En las actividades realizadas en grupo el docente puede, basándose en la información de que disponga, personalizar la calificación para cada integrante del grupo. Cualquier actividad no entregada se considerará puntuada con cero puntos. Es potestativo de los docentes aceptar o no entregas fuera de los plazos que se indiquen. En caso de que estas entregas fuera de plazo se acepten, es potestativo del docente decidir si aplica alguna penalización y la cuantía de la misma.

Contenidos

1. **Ingeniería de requisitos**

1.1. Introducción.

1.2. Método de captura de requisitos: Lean

1.3. Agile, la búsqueda de la confianza.

1.4. Modelización (Diagramas de Actividad).

1.5. Especificación (introducción al OCL).

2. **Pruebas del Software**

2.1. Estrategias de pruebas y técnicas.

2.2. Test Driven Development

2.3. Code Coverage

2.4. Law of Demeter

2.5. Dependency Injection

2.6. Lower 's' singleton pattern

2.7. Revisión de código

2.8. Professionalism as emergence of Testing + Agile

3. **Patrones de diseño de software**

3.1. High Cohesion / Low Coupling

3.2. Plugin Architecture

3.3. UML drawings, relations, and dependency directions

3.4. Patrón S.O.L.I.D.

3.5. Patrón Command.

3.6. Patrón Observer.

3.7. Patrón Event Target.

3.8. Event Bus.

3.9. Patrón Modelo Vista Controlador.

3.10. State.

3.11. Patrones funcionales puros.

Actividades de aprendizaje

Se pone a disposición de los estudiantes todo un seguido de actividades de caracter claramente práctico (ejercicios cortos, problemas, ...) que son la base de las actividades de aprendizaje de la asignatura. Los estudiantes tendrán que resolver estas actividades, amenudo de forma no presencial, siguiendo las indicaciones de los docentes y tambien seran trabajadas en clase, ja sea como ejemplos en las sesiones de teoria, o en las sesiones de laboratorio. Si bien estas actividades tendrán carácter optativo (los docentes no verificaran de manera individualizada la realización por parte de los estudiantes), seran imprescindibles para asumir les conocimientos teorico-practicos de la asignatura.

Con objetivo de recoger les evidencias del logro de los resultados de aprendizaje esperades se realizaran las siguientes actividades de caracter evaluativo: hasta cinco practicas y hasta dos pruebas escritas (examenes).

Practica 1 (Evidencia de los resultados de aprendizaje RA1)

Se trata de captar los requisitos de un software mediante metodologia Lean.

Ponderación: 8% de la nota final.

Objectius específicos: En finalizar los estudiantes seran capaces de:

· Modelar los requisitos del software.

· Escribir y entender los requisitos así como los actores implicados.

Pràctica 2 i 3 (Evidencia de los resultados de aprendizaje RA3 i RA4)

Se desarrollará un pequeño software mediante Test Driven Development mediante la aplicación del bloque 2.

Ponderación: 16% (8% cada una) de la nota final.

Objetivos específicos: Al finalizar la actividad los estudiantes deben ser capaces de:

· Conocer estrategias y aplicar técnicas de pruebas del software.

· Ser capaces de hacer un dessenvolupament mediante TDD

· Conocer las fases de un proyecto software y el agile software development.

Práctica 4 y 5 (Evidencia de los resultados de aprendizaje RA7, RA8, y RA9)

Se trata de rediseñar un pequeño software aplicando uno o varios de los siguientes patrones: MVC, Observer o Command.

Ponderación: 16% (8% cada una) de la nota final.

Objetivos específicos: Al finalizar la actividad los estudiantes deben ser capaces de:

· Modelar aplicaciones software.

· Aplicar los principios de análisis y diseño orientado a objetos.

· Aplicar y analizar el uso de patrones de diseño de software.

Prueba 1 (Evidencia de los resultados de aprendizaje RA1, RA2, RA3, RA4, RA5 y RA6)

Prueba individual de los conceptos teóricos y procedimientos prácticos del bloque 1 y 2.

Esta prueba representa el 30% de la calificación final de la asignatura.

El objetivo de esta actividad es evaluar si el estudiante sabe:

· Modelar los requisitos del software.

. Escribir y entender especificaciones formales.

· Conocer estrategias y aplicar técnicas de pruebas del software

Prueba 2 (Evidencia de los resultados de aprendizaje RA5, RA6, RA7, RA8, y RA9)

Prueba individual de los conceptos teóricos y procedimientos prácticos del bloque 2 y 3.

Esta prueba representa el 30% de la calificación final de la asignatura.

El objetivo de esta actividad es evaluar si el estudiante sabe:

· Aplicar los principios de análisis y diseño orientado a objetos

· Aplicar y analizar el uso de patrones de diseño de software

Nota: la competencia transversal asociada a la asignatura (conocimiento tercera lengua) se trabaja a partir de las fuentes documentales que los estudiantes deben consultar, dado que todas ellas se encuentran en inglés. En algunas ocasiones también se suministran en inglés los enunciados de algunas de las prácticas y materiales de apoyo de la asignatura

Sistema de evaluación

La nota final se calculará con las calificaciones de las actividades ponderadas de la forma siguiente:

- Prueba 1: 30%
- Prueba 2: 30%
- Prácticas de la 1 a la 5: 40% (8% cada una de las prácticas)

Con las ponderaciones anteriores, las prácticas de laboratorio tienen un peso del 40% y las pruebas un 60%.

Sólo podrán recuperarse las pruebas 1 y 2 en una única prueba de toda la asignatura (las prácticas no se podrán recuperar). El 60% de la nota final de la asignatura será la mayor entre la prueba de recuperación y la obtenida en las pruebas 1 y 2.

Para poder realizar la prueba de recuperación del estudiante deberá cumplir las tres condiciones siguientes:

- La nota de la asignatura es inferior a cinco.
- Al menos tiene un 3 de las pruebas.
- Al menos tiene un 3 de prácticas.

Recursos

Básicos

Bibliografías

- Patrones de diseño : elementos de software orientado a objetos reutilizable.

  Gamma, Eric et all. Addison-Wesley, cop. 2003. ISBN 9788478290598.

- Pruebas de software y JUnit: un análisis en profundidad y ejemplos prácticos. Bolaños Alonso, Daniel; Sierra Alonso, Almudena; Alarcón Rodríguez, Miren Idoia. 1a. Prentice-Hall España, 2008. ISBN 9788483223543.

- Warmer, Jos B; Kleppe, Anneke G. The Object constraint language : getting your models ready for MDA. 2nd ed. Reading 2003: Addison-Wesley. ISBN 0321179366.

Complementarios

Bibliografías

- Patrones de diseño aplicados a Java. Stelting Stephen, Maassen Olav. Prentice-Hall 2003. ISBN 9788420538396.

- UML y patrones : una introducción al análisis y diseño orientado a objetos y al proceso unificado.

  Larman, Crai Prentice Hall, cop. 2003. ISBN 978 8420534382.

Enlaces web

- Robert C. Martin: Principis del Disseny Orientat a Objectes http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
- Web del professor: http://david-rodenas.com/tc/es3
- Writing Testable Code. Miško Hevery. http://misko.hevery.com/code-reviewers-guide/