Guía docente
270080 - AS - Arquitectura del Software

Última modificación: 30/01/2024

Unidad responsable: Facultad de Informática de Barcelona
Unidad que imparte: 747 - ESSI - Departamento de Ingenieria de Servicios y Sistemas de Información.

Titulación: GRADO EN INGENIERÍA INFORMÁTICA (Plan 2010). (Asignatura optativa).

Curso: 2023 **Créditos ECTS:** 6.0 **Idiomas:** Catalán

PROFESORADO

Profesorado responsable: CRISTINA GÓMEZ SEOANE

Otros:

Primer quadrimestre:
CLAUDIA PATRICIA AYALA MARTINEZ - 12, 13, 21, 22
CRISTINA GÓMEZ SEOANE - 11, 12, 13, 14, 22

Segon quadrimestre:
PAU DURAN MANZANO - 11
CRISTINA GÓMEZ SEOANE - 11, 12, 13

CAPACIDADES PREVIAS

Saber qué es una especificación (formal) en UML/OCL de los requisitos funcionales y no funcionales de un sistema.
Saber el contexto en el qué se puede realizar el diseño de un sistema y, en particular, cuáles son las actividades previas y siguientes.
Conocer la programación orientada a objetos y (como mínimo) un lenguaje de programación orientada a objetos.

REQUISITOS

- Pre-requisito IES

COMPETENCIAS DE LA TITULACIÓN A LAS QUE CONTRIBUYE LA ASIGNATURA

Específicas:
CES1.1. Desarrollar mantener y evaluar sistemas y servicios software complejos y/o críticos.
CES1.4. Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de red.
CES2.2. Diseñar soluciones apropiadas en uno o más dominios de aplicación, utilizando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos.
CT2.1. Demostrar conocimiento y capacidad de aplicación de los principios, metodologías y ciclos de vida de ingeniería de software.
CT2.3. Diseñar, desarrollar, seleccionar y evaluar aplicaciones, sistemas y servicios informáticos, y al mismo tiempo asegurar su fiabilidad, su seguridad y su calidad, conforme a principios éticos y a la legislación y la normativa vigente.

Genéricas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

METODOLOGÍAS DOCENTES

La docencia de la asignatura está estructurada en clases de teoría, clases de problemas y clases de laboratorio.

En las clases de teoría el profesorado presenta los contenidos esenciales de la asignatura. El profesorado usa transparencias que los estudiantes tienen disponibles antes de las clases. Otros contenidos de la asignatura se tendrán que estudiar de forma autónoma.

En las clases de problemas se practican los contenidos de la asignatura presentados o adquiridos autónomamente previamente a la clase. Durante la clase se resuelven los problemas propuestos previamente a la clase y se discutirán las diferentes soluciones. En algunos casos, el alumnado entrega los problemas resueltos. También, en algunas sesiones, al inicio de la clase el estudiantado contesta un cuestionario sobre los contenidos de la asignatura presentados y practicados previamente a la clase.

En las clases de laboratorio los estudiantesa trabajan en equipos de 2 personas (pair programming). Durante la clase el estudiantado realiza ejercicios de laboratorio propuestos por el profesor. Al inicio de algunas clases contestan un cuestionario sobre los contenidos de la asignatura presentados y practicados y/o sobre los problemas propuestos previamente a la clase.

El estudiantado tienen que asistir a las clases del GRUPO DONDE ESTAN MATRICULADOS.

OBJETIVOS DE APRENDIZAJE DE LA ASIGNATURA

1.Explicar qué es la arquitectura lógica y física de un sistema software y las relaciones entre ellas.
2.Explicar las características de una arquitectura en capas y los principios de diseño que rigen la construcción de sistemas software con este tipo de arquitectura.
3.Explicar las características de una arquitectura orientada a objetos y los principios de diseño que rigen la construcción de sistemas software con este tipo de arquitectura.
4.Diseñar sistemas software utilizando una arquitectura en capas y orientada a objetos y usando servicios software, partiendo de la especificación de sus requisitos (funcionales y no funcionales) utilizando metodologías clásicas y agiles.
5.Representar el diseño de los sistemas software utilizando el lenguaje UML.
6.Aplicar correctamente los principios de diseño al realizar el diseño de sistemas software.
7.Diseñar la capa de dominio (negocio) de un sistema software utilizando metodologías clásicas y ágiles.
8.Diseñar la capa de presentación de un sistema software identificando los patrones a aplicar i aplicarlos correctamente.
9.Explicar las características de las diferentes estrategias de persistencia.
10.Diseñar la capa de gestión de datos de un sistema software aplicando los patrones de acceso o de traducción de datos.
11.Identificar y usar correctamente servicios software existentes al realizar el diseño de sistemas software.
12.Evaluar y criticar los diseños de los sistemas software y proponer diseños alternativos que mejoren su calidad.
13.Explicar qué es la arquitectura y el diseño del software y en qué consisten las diferentes vistas de la arquitectura del software.

HORAS TOTALES DE DEDICACIÓN DEL ESTUDIANTADO

Tipo	Horas	Porcentaje
Horas aprendizaje autónomo	84,0	56.00
Horas grupo grande	30,0	20.00
Horas actividades dirigidas	6,0	4.00
Horas grupo pequeño	30,0	20.00

Dedicación total: 150 h

CONTENIDOS

Introducción a la arquitectura y diseño del software

Descripción:
Metodologías de desarrollo de software. Arquitectura y diseño de software. Arquitectura lógica y física de un sistema software. Rol de los patrones de diseño en el diseño del software. Arquitectura y diseño del software en metodologías clásicas y ágiles.

Arquitectura en capas y orientada a objetos

Descripción:
Patrón arquitectónico en capas. Aplicación del patrón arquitectónico en capas a los sistemas y servicios software. Principios de diseño de la arquitectura en capas. Patrón arquitectónico orientación a objetos. Aplicación del patrón arquitectónico orientación a objetos a los sistemas software. Principios de diseño de la arquitectura orientada o objetos.

Diseño del software utilizando metodologías clásicas

Descripción:
Diseño del software utilizando metodologías clásicas. Diseño de la capa de dominio. Patrones Domain Model y Transaction Script. Patrones de diseño de la capa de dominio. Uso de servicios software. Diseño de la capa de presentación. Diseño externo de un sistema software. Diseño interno de la capa de presentación. Diseño de la capa de datos. Tecnología de bases de datos relacionales. Estrategias de gestión de la persistencia. Generación automática de la persistencia. Diseño directo de la persistencia. Otras arquitecturas. Arquitectura de microservicios

Diseño de software utilizando metodologías ágiles

Descripción:
Disseny del software utilizando metodologías ágiles. Extreme Programming. TDD. Tests en TDD.
Coding en TDD. Pair Programming. Simple Design and Incremental design and architecture. Code Smells and Refactorings.

ACTIVIDADES

C2

Descripción:
Control 2 de teoría y problemas donde se evaluan parte de los objetivos del tema 3.

Objetivos específicos:
4, 6, 8, 9, 10, 11

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 12h
Actividades dirigidas: 2h
Aprendizaje autónomo: 10h

C1

Descripción:
Control 1 de teoría y problemas donde se evaluan todos los objetivos de los temas 1 hasta parte del tema 3

Objetivos específicos:
1, 2, 3, 4, 5, 6, 7, 13

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 12h
Actividades dirigidas: 2h
Aprendizaje autónomo: 10h

Introducción a la arquitectura del software

Descripción:
Escucha con atención, coge los apuntes que considera necesarios y pregunta las dudas que tenga. Además, estudia autónomamente los contenidos que el profesor indica

Objetivos específicos:
1, 13

Dedicación: 3h
Grupo grande/Teoría: 2h
Aprendizaje autónomo: 1h

Estudio de la arquitectura en capas

Descripción:
Escucha con atención, coge los apuntes que considera necesarios y pregunta las dudas que tenga. Además, estudia autónomamente los contenidos que el profesor indica

Objetivos específicos:
2

Dedicación: 3h
Grupo pequeño/Laboratorio: 1h
Aprendizaje autónomo: 2h

Estudio de la arquitectura orientada a objetos

Descripción:
Escucha con atención, coge los apuntes que considera necesarios y pregunta las dudas que tenga. Además, estudia autónomamente los contenidos y hace los ejercicios que el profesor indica

Objetivos específicos:
3, 5, 12

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 14h
Grupo pequeño/Laboratorio: 6h
Aprendizaje autónomo: 8h

Estudio del diseño del software utilizando metodologías clásicas

Descripción:
Escucha con atención, coge los apuntes que considera necesarios y pregunta las dudas que tenga. Además, estudia autónomamente los contenidos y hace los ejercicios que el profesor indica

Objetivos específicos:
4, 5, 6, 7, 8, 9, 10, 11

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 56h
Grupo grande/Teoría: 14h
Grupo pequeño/Laboratorio: 14h
Aprendizaje autónomo: 28h

Estudio del diseño del software utilizando metodologías ágiles

Descripción:
Escucha con atención, coge los apuntes que considera necesarios y pregunta las dudas que tenga. Además, estudia autónomamente los contenidos y hace los ejercicios que el profesor indica

Objetivos específicos:
4, 6, 7, 12

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 32h
Grupo grande/Teoría: 10h
Grupo pequeño/Laboratorio: 7h
Aprendizaje autónomo: 15h

C3

Descripción:
Control 3 de laboratorio donde se evaluan todos los objectivos del tema 4

Objetivos específicos:
6, 7, 12

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 12h
Actividades dirigidas: 2h
Aprendizaje autónomo: 10h

Revisión de C1 (Control 1) con feedback formativo (entrega solución con comentarios grupales)

Descripción:
Revisión de C1 (Control 1) con feedback formativo (entrega solución) en grupos de 10 estudiantes (vía meet o con aula asignada, dependiendo de la disponibilidad de aulas).

Objetivos específicos:
1, 2, 3, 4, 5, 6, 7

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 2h
Actividades dirigidas: 2h

Revisión de C2 (Control 2) con feedback formativo (entrega solución con comentarios grupales)

Descripción:
Revisión de C1 (Control 1) con feedback formativo (entrega solución) en grupos de 10 estudiantes (vía meet o con aula asignada, dependiendo de la disponibilidad de aulas).

Objetivos específicos:
4, 6, 8, 9, 10, 11

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 2h
Actividades dirigidas: 2h

Revisión de C3 (Control 3) con feedback formativo (entrega solución con comentarios grupales)

Descripción:
Revisión de C3 (Control 3) con feedback formativo (entrega solución) en grupos de 10 estudiantes (vía meet o con aula asignada, dependiendo de la disponibilidad de aulas).

Objetivos específicos:
6, 7, 12

Competencias relacionadas:
G9. RAZONAMIENTO: Capacidad de razonamiento crítico, lógico y matemático. Capacidad para resolver problemas dentro de su área de estudio. Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales. Capacidad de diseñar y realizar experimentos sencillos, y analizar e interpretar sus resultados. Capacidad de análisis, síntesis y evaluación.

Dedicación: 2h
Actividades dirigidas: 2h

SISTEMA DE CALIFICACIÓN

La Nota Final (NF) de las competencias técnicas de la asignatura se obtiene con la siguiente fórmula:

NF= 0,2*C1 + 0,25*C2 + 0,4*C3 + 0,1*NT + NPP

C1 es la nota del primer control
C2 es la nota del segundo control
C3 es la nota del tercer control
NT es la nota de los tareas realizadas (entrega de problemas antes de la classe y cuestionarios realitzados al inicio de las clases de problemas y laboratorio.

Cálculo de NT:
- Durante el curso el estudiantado podrá realizar 5 tareas.
- Si entrega 4 o menos tareas, NT será igual al sumatorio de las notas de todas las tareas entregadas dividido por 4.
- Si entrega 5 tareas, NT será igual al sumatorio de las 4 mejores notas de las tareas entregadas dividido por 4.

NPP es la nota de problemas hechos y entregados en clase y de la participación en clase. Esta nota tendrá como máximo un valor de 0,5. Para obtener esta nota se considerarán los siguientes aspectos:
- La entrega de un 75% o más dels problemas propuestos para entregar.
- La participación activa en clase.

El estudiantado SÓLO SERÁ EVALUADOS SI ASISTE a las clases del GRUPO DONDE ESTÁ MATRICULADO.

La competencia transversal se evalúa mediante una pregunta en los controles, donde el alumnado deben demostrar su capacidad crítica y de evaluación de los diseños propuestos. La Nota Final de la competencia transversal se obtiene haciendo la media de las notas obtenidas en estas preguntas y tiene una calificación de:
A si la nota está entre 8.5 y 10
B si la nota está entre 7 y 8.4
C si la nota está entre 5 y 6.9
D si la nota está entre 0 y 4.9

BIBLIOGRAFÍA

Básica:
- Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. Design patterns: elements of reusable object-oriented software. Addison-Wesley, 1995. ISBN 0201633612.
- Larman, C. Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. 3rd ed. Prentice Hall PTR, 2005. ISBN 0131489062.
- Fowler, M. Patterns of enterprise application architecture. Addison-Wesley, 2003. ISBN 0321127420.
- Rumbaugh, J.; Jacobson, I.; Booch, G. The unified modeling language reference manual. 2nd ed. Addison-Wesley, 2005. ISBN 0321245628.
- Shore, J.; Warden, S. The art of agile development. O'Reilly Media, 2008. ISBN 9780596527679.
- Christensen, H.B. Flexible, reliable software: using patterns and agile development. CRC Press, 2010. ISBN 9781420093629.
- Meszaros, G. XUnit test patterns: refactoring test code. Addison-Wesley, 2007. ISBN 9780131495050.

Complementaria:
- Pressman, R.S.; Maxim, B.R. Software engineering: a practitioner's approach. 9th ed. New York: McGraw Hill Higher Education, 2020. ISBN 9781260548006.
- Sommerville, I. Software engineering. 10th ed., global ed. Pearson Education Limited, 2016. ISBN 9781292096148.
- Martin, R.C. Agile software development: principles, patterns, and practices. First edition. Pearson Education Limited, 2014. ISBN 9781292038360.
- Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M. Pattern-oriented software architecture: Vol. 1: a system of patterns. Wiley, 1996. ISBN 0471958697.
- Beck, K. Test-driven development: by example. Addison-Wesley, 2003. ISBN 0321146530.
- Professorat de l'assignatura. Material d'elaboració pròpia de l'assignatura.

RECURSOS

Enlace web:
- http://msdn.microsoft.com/en-us/library/ff650706.aspx- http://martinfowler.com/eaaCatalog/- https://industriallogic.com/xp/refactoring/catalog.html- http://www.uml.org/- http://refactoring.com/catalog/