Guía docente
230719 - ARQSOFT - Arquitectura de Software

Última modificación: 25/05/2023

Unidad responsable: Escuela Técnica Superior de Ingeniería de Telecomunicación de Barcelona
Unidad que imparte: 701 - DAC - Departamento de Arquitectura de Computadores.

Titulación: MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN (Plan 2013). (Asignatura optativa).
MÁSTER UNIVERSITARIO EN INGENIERÍA AERONÁUTICA (Plan 2014). (Asignatura optativa).
MÁSTER UNIVERSITARIO EN TECNOLOGÍAS AVANZADAS DE TELECOMUNICACIÓN (Plan 2019). (Asignatura obligatoria).

Curso: 2023 **Créditos ECTS:** 5.0 **Idiomas:** Inglés

PROFESORADO

Profesorado responsable: Consultar aquí / See here: https://telecos.upc.edu/ca/estudis/curs-actual/professorat-responsables-coordinadors/responsables-assignatura

Otros: Consultar aquí / See here: https://telecos.upc.edu/ca/estudis/curs-actual/professorat-responsables-coordinadors/professorat-assignat-idioma

CAPACIDADES PREVIAS

Bases conceptuales sólidas de programación orientada a objetos: objeto, atributos, métodos, clase, excepción, encapsulación de datos, herencia y polimorfismo. Bases sólidas de programación en Java: sintaxis de definición de clases, sintaxis de gestión de herencia, polimorfismo. Gestión de excepciones en Java. Entrada/Salida básica en Java. Gestión de contenedores más comunes: conjuntos, listas, mapas, grafos.

METODOLOGÍAS DOCENTES

La asignatura se impartirá en base a sesiones de teoría en las que se presentarán los conceptos teóricos más relevantes, acompañados de ejemplos de aplicación de los mismos. Se intercalarán a lo largo del curso sesiones de laboratorio que servirán para consolidar la asimilación de los contenidos teóricos y poner éstos en práctica.

Se deberá desarrollar un programa de grado de complejidad moderado/alto por grupos, donde se pondrán en práctica los elementos teóricos presentados durante el curso.

OBJETIVOS DE APRENDIZAJE DE LA ASIGNATURA

Conseguir que quien curse esta asignatura sea capaz de:
1. Afrontar con éxito el trabajo en el seno de un equipo que confronte la realización de un programa de complejidad moderada/alta;
2. Llevar a cabo tareas de análisis orientado a objetos, diseño orientado a objetos utlizando correctamente patrones de diseño de software;
3. Desarrollar entornos automatizados de pruebas de programas.
4. Añadir a los programas el nivel de persistencia de datos utilizando bases de datos relacionales.

HORAS TOTALES DE DEDICACIÓN DEL ESTUDIANTADO

Tipo	Horas	Porcentaje
Horas aprendizaje autónomo	86,0	68.80
Horas grupo grande	26,0	20.80
Horas grupo pequeño	13,0	10.40

Dedicación total: 125 h

CONTENIDOS

Introducción a las pruebas del software. JUnit

Descripción:
Diseño de software preparado para ser probado.
Pruebas unitarias: JUnit
Simulación de componentes externos ("Fakes" y "Mocks"). Herramienta Mockito

Objetivos específicos:
Presentar las actividades de pruebas de código en el contexto del proceso de desarrollo de software. Presentar herramientas que facilitan estas actividades. Practicar con JUnit como herramienta que permite automatizar pruebas de software

Actividades vinculadas:
Sesión de laboratorio. Construcción de pruebas para comprobar corrección de código en una clase.

Dedicación: 8h
Grupo grande/Teoría: 3h
Grupo pequeño/Laboratorio: 3h
Aprendizaje autónomo: 2h

Procesos iterativos de desarrollo de software. Introducción a Scrum

Descripción:
Noción de proceso de desarrollo de software. Características principales de los procesos iterativos. Comparación con los procesos en cascada. Introducción a Scrum.

Objetivos específicos:
Presentar brevemente el concepto de proceso de desarrollo de software como un conjunto de actividades realizadas en equipo cuyo objetivo es el desarrollo de un programa. Mostrar las características principales de los procesos iterativos como alternativa a los procesos en cascada, poniendo de manifiesto la fuerte imbricación temporal de análisis, diseño, codificación y pruebas. Mostrar detalles de uno de los procesos iterativos existentes: Scrum.

Actividades vinculadas:
Realización de un proyecto de desarrollo de un programa de complejidad moderada en equipo.

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

Introducción al análisis orientado a objetos

Descripción:
Productos del análisis orientado a objetos: repertorio de casos de uso y modelo conceptual del problema.
Concepto de requisitos. Tipos de requisitos. Técnicas de captura de requisitos. Casos de uso como método para capturar requisitos. Concepto y partes de un caso de uso.
Construcción del modelo conceptual a partir de los casos de uso. Identificación de las clases. Identificación de los atributos de las clases. Identificación y definición completa de las relaciones entre clases.
Ejemplos.

Objetivos específicos:
Categorizar los diferentes tipos de requisitos de un programa. Ser capaz de identificar casos de uso. Ser capaz de desarrollar al completo los casos de uso más relevantes.
Ser capaz de construir el modelo conceptual a partir de los casos de uso: conceptos/clases, atributos y relaciones.

Actividades vinculadas:
Ejemplos de clase y proyecto desarrollado en equipo

Dedicación: 16h 30m
Grupo grande/Teoría: 4h 30m
Aprendizaje autónomo: 12h

Introducción al diseño orientado a objetos

Descripción:
Productos del diseño orientado a objetos: diagramas de clases, diagramas de interacción, documentación de clases diseñadas.
Introducción a los diagramas de interacción en UML: diagramas de secuencia y diagramas de colaboración.
Diagrama de secuencia del sistema
Técnicas para identificar y especificar los métodos de las clases del programa. Patrones GRASP para identificar métodos y asignarlos a la clase pertinente.
Uso de patrones GRASP para construir diagramas de interacción a partir de los casos de uso.

Objetivos específicos:
Ser capaz de construir diagramas de secuencia UML. Ser capaz de construir el diagrama de secuencia de un sistema. Ser capaz de utilizar correctamente los patrones GRASP de asignación de responsabilidades a las clases a partir de los casos de uso para identificar los métodos de las clases que se diseñan.

Actividades vinculadas:
Ejemplos y proyecto realizado en equipo

Dedicación: 16h 30m
Grupo grande/Teoría: 4h 30m
Aprendizaje autónomo: 12h

Introducción a las bases de datos

Descripción:
Concepto de bases de datos. Brever repaso histórico. Tipos de bases de datos. Bases de datos relacionales. Breve introducción al SQL. Uso de una base de datos relacional en Java.

Objetivos específicos:
Saber qué es una base de datos relacional. Ser capaz de diseñar tablas sencillas de bases de datos relacionales. Entender los elementos básicos de un diagrama Entidad-Relacion. Conocer elementos introductorios del lenguaje SQL. Conectar un programa Java con una base de datos SQL

Actividades vinculadas:
Sesión de laboratorio dedicada y proyecto realizado en equipo

Dedicación: 6h
Grupo grande/Teoría: 2h
Grupo pequeño/Laboratorio: 2h
Aprendizaje autónomo: 2h

Completando el diseño orientado a objetos con patrones de diseño

Descripción:
Introducción del concepto de patrón de diseño de software. Patrones GoF. Clasificación.

Objetivos específicos:
Entender el concepto de patrón de diseño de software, sus tipos y los beneficios que su uso comportan en un diseño.

Actividades vinculadas:
Presentación en sesión de teoría

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

Estudio de algunos patrones de diseño relevantes

Descripción:
Presentación y estudio de algunos de los patrones de diseño más relevantes: Estrategia, Decorador, Método Fábrica, Fábrica Abstracta, Compuesto, Observador

Objetivos específicos:
Conocer los detalles de los patrones estudiados. Ser capaz de postular correctamente el uso de patrones de diseño de software en la fase de diseño de un programa. Ser capaces de implementar correctamente los patrones estudiados.

Actividades vinculadas:
Sesiones de teoría
Sesiones de laboratorio para algunos patrones
Proyecto realizado en grupo

Dedicación: 29h
Grupo grande/Teoría: 8h
Grupo pequeño/Laboratorio: 8h
Aprendizaje autónomo: 13h

Agregación de patrones: el patrón modelo-vista-controlador

Descripción:
Estudio del patrón modelo-vista-controlador como agregación de varios de los patrones de diseño antes estudiados.

Objetivos específicos:
Aprender los detalles del modelo-vista-controlador como el resultante de agregar varios de los patrones de diseño estudiados en el tema anterior. Implementar correctamente esta agregación de patrones.

Actividades vinculadas:
Sesión de teoría y proyecto desarrollado en equipo

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

Proyecto

Descripción:
contenido castellano

Dedicación: 41h
Aprendizaje autónomo: 41h

SISTEMA DE CALIFICACIÓN

Entregas parciales de actividades relacionadas con el proyecto: 10%
Examen: 20%
Proyecto: 60%
Evaluación subjetiva: 10%

NORMAS PARA LA REALIZACIÓN DE LAS PRUEBAS.

El examen se realizará sobre el ordenador. No se permitirá el uso de sistemas de mensajería electrónica de ningún tipo.

BIBLIOGRAFÍA

Básica:
- Freeman, E.; Freeman, E.; Sierra, K.; Bates, B. Head First design patterns. 1st ed. Sebastopol, CA: O'Reily, 2004. ISBN 0-596-00712-4.
- Larman, C. Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. 3rd ed. Upper Saddle River, N.J.: Prentice Hall PTR, 2005. ISBN 0-13-148906-2.
- Gamma, E.; Helm, R.; Johnson, R.; Vissides, J. Patrones de diseño: elementos de software orientado a objetos reutilizable. 1st ed. Madrid: Addison Wesley, 2003. ISBN 84-7829-059-1.

RECURSOS

Enlace web:
- Writing effective use cases / Alistair Cockburn. Boston : Addison-Wesley, cop. 2001. ISBN: 0201702258. https://www.infor.uva.es/mlaguna/is1/materiales/BookDraft1.pdf.
https://discovery.upc.edu/discovery/fulldisplay?docid=alma991002517209706711&context=L&vid=34CSUC_UPC:VU1&lang=ca- Java design patterns - example tutorial. https://www.journaldev.com/1827/java-design-patterns-example-tutorial- Design patterns. https://www.oodesign.com/