Software de Comunicaciones

Curso Académico: (2021 / 2022) **Fecha de revisión: 04-06-2021**

Departamento asignado a la asignatura: Departamento de Ingeniería Telemática

Coordinador/a: IBAÑEZ ESPIGA, MARIA BLANCA

Tipo: Obligatoria Créditos ECTS : 6.0

Curso : 4 Cuatrimestre : 1

REQUISITOS (ASIGNATURAS O MATERIAS CUYO CONOCIMIENTO SE PRESUPONE)

Programación; Programación de Sistemas

OBJETIVOS

Al finalizar el curso, el alumno será capaz de entender los principios, conceptos, métodos y técnicas de ingeniería de software. El estudiante será capaz de aplicar sus conocimientos y habilidades para organizar y desarrollar un proyecto de tamaño mediano de software que satisfaga los requisitos fijados, y sea además fiable, fácil de comprender, modificar y mantener.
Para lograr estos objetivos, el estudiante debe adquirir una serie de conocimientos, capacidades y actitudes
Por lo que se refiere a los conocimientos, al finalizar el curso el estudiante conocerá:

- ¿Qué es la ingeniería de software?
- El conjunto de procesos de software y modelos de procesos de software
- Las diferentes formas de expresar los requisitos de software
- Los modelos de sistema que puede ser desarrollado durante el proceso de ingeniería de requisitos
- Los modelos de la arquitectura de software
- Los patrones de diseño principal del software
- Las técnicas utilizadas para probar el software

En cuanto a las capacidades, las podemos clasificar en dos grupos: uno de capacidades específicas y otro de capacidades más genéricas o destrezas.
En cuanto a las capacidades específicas, al finalizar el curso el alumno será capaz de:

- Planificar un proyecto de software completo
- Analizar y formalizar los requisitos de software de un proyecto
- Desarrollar y analizar modelos del software
- Diseñar la arquitectura de software de un proyecto
- Usar las estructuras de datos y las técnicas de programación adecuadas para las tareas de programación que requiera el software
- Favorecer el mantenimiento del sistema en todas sus etapas de desarrollo mediante una documentación adecuada
- Verificar y validar el software desarrollado
- Velar por la calidad del software

En cuanto a las capacidades generales o destrezas, durante el curso se trabajarán:

- La capacidad de utilizar técnicas de Ingeniería del Software para crear productos de software efectivos
- La capacidad para hacer frente a restricciones derivadas de recursos humanos, tiempo y costos
- La capacidad para trabajar en un equipo de desarrollo de software
- La capacidad de presentar y defender en público las soluciones desarrolladas

En cuanto a las actitudes, el alumno tras cursar el curso debería tener:

- Una actitud crítica en cuanto a la idoneidad de distintas técnicas y herramientas que pueden ser aplicadas al desarrollo de un sistema de software
- Valores éticos hacia su trabajo como desarrollados de software
- Una actitud pro-activa hacia el trabajo
- Una actitud de colaboración y trabajo en equipo

Enlace al documento

DESCRIPCIÓN DE CONTENIDOS: PROGRAMA

1. Introducción
   1.1 Introducción a la Ingeniería de Software
   1.2 Procesos de software
2. Análisis de requisitos
   2.1 Requisitos de software
   2.2 Modelos de sistemas
3. Diseño de arquitectura de sistemas
   3.1 Modelos de Arquitecturas de Software
   3.2 Patrones de Software
4. Pruebas de software. Técnicas
   4.1 Introducción a las pruebas de software
   4.2 Tipos de pruebas de software

ACTIVIDADES FORMATIVAS, METODOLOGÍA A UTILIZAR Y RÉGIMEN DE TUTORÍAS

La metodología consiste en impartir clases magistrales, realizar clases de ejercicios prácticos y supervisión por parte del profesor en los laboratorios de programación. Las actividades que se llevan a cabo son:
* Clases magistrales. Presentación por parte del profesor de los principales conceptos a modo de resumen. Se fomentan en este tipo de sesiones tanto la interactividad como la discusión de los principales problemas planteados.
* Clases de ejercicios prácticos: Sesiones en las que se plantean problemas reales y se permite a los estudiantes su análisis así como el planteamiento de posibles soluciones.
* Laboratorios de programación, donde los estudiantes implementan su proyecto bajo la supervisión del personal docente. Estas sesiones suponen trabajo adicional de los estudiantes durante un período de varios días. Los estudiantes son atendidos por el personal docente a través de múltiples canales para resolver sus dudas.

SISTEMA DE EVALUACIÓN

Se realizará evaluación continua y examen final siguiendo los siguientes criterios:

Un examen parcial escrito (20% de la nota final de los estudiantes). El examen contiene preguntas de estudio proporcionadas por el instructor, y problemas prácticos basados en los problemas del proyecto.

Evaluación de un proyecto de ingeniería de software realizado en equipo (40% de grado de estudiantes del curso). El equipo en función de:
El plan de proyecto que presente el equipo en forma escrita.
Avances parciales del proyecto que serán entregados en forma escrita.
Presentación oral del trabajo.

El examen final (40%) contiene preguntas de estudio proporcionadas por el instructor, y problemas prácticos basados en los problemas del proyecto. El examen final es obligatorio para todos los estudiantes. Los estudiantes deben obtener una nota superior o igual a 4 (4/10) para aprobar la asignatura. La nota final debe ser superior a 5 (5/10).

Peso porcentual del Examen Final:	40
Peso porcentual del resto de la evaluación:	60

BIBLIOGRAFÍA BÁSICA

- E. Gamma, R. Helm, R. Johnson, J. Vlissides Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.
- Ian Sommerville Software Engineering, Addison-Wesley.
- J. Rumbaugh, I. Jacobson, G. Booch The Unified Modeling Language Reference Manual, Addisson-Wesley.

BIBLIOGRAFÍA COMPLEMENTARIA

- ACM Ingeniería de Software. Código Ética y Práctica Profesional 5.2, ., .
- Bruegge, B; Dutoit, A Object Oriented Software Engineering Using UML, Patterns and Java, Pearson Education Internationsl, 2004
- C. Ebert, et al. Best Practices in Software Measurement. How to Use Metrics to Improve Project and Process Performance, Springer.
- Dashofy, E; Madyidovic, N; Taylor R, Software Architecture: Foundations, Theory and Practice, John Wiley & Sons, 2009
- Fowler, M; Beck, K.; Opdyke, W; Roberts; D. Refactoring Improving the Design of Existing Code, Addison-Wesley Professional, 1999
- Fox, A; Patterson D. Engineering Software as a Service: An Agile Approach Using Cloud Computing, Strawberry Canyon, 2014

- G. Montoya, I. Sommerville Requirements Engineering. Process and Techniques, Wiley.
- Jorgesen, P Software Testing. A Craftsman´s Approach, CRC Press, 2013
- Martin, R. Clean COde, Prentice Hall, 2008
- Roger S Pressman Ingeniería del software. Un enfoque práctico, Mc Graw Hill.
- W. Perry Effective Methods for Software Testing, Wiley.