# Guía Didáctica - GRADO

## ASIGNATURA: Mantenimiento y evolución del Software

Título: Grado de Ingeniería en Informática
Módulo: Mención en Ingeniería de Software
Créditos: 6 ECTS
Código: 43GIIN

viu Universidad Internacional de Valencia

# Índice

# 1. Organización general

## 1.1. Datos de la asignatura

| | |
|---|---|
| **MÓDULO** | **Menciones** |
| **MATERIA** | **Mención en Ingeniería del Software** |
| **ASIGNATURA** | **43GIIN Mantenimiento y evolución del software**<br>**6 ECTS** |
| **Carácter** | Obligatoria |
| **Curso** | Tercero |
| **Cuatrimestre** | Segundo |
| **Idioma en que se imparte** | Castellano |
| **Requisitos previos** | No existen |
| **Dedicación al estudio recomendada por ECTS** | **25 horas** |

## 1.2. Introducción a la asignatura

Esta asignatura pertenece tercer curso del Grado Ingeniería Informática Mención Ingeniería de Software.

En esta asignatura buscaremos comprender la importancia de concebir el mantenimiento como parte de un proceso de ingeniería de software.

El punto de partida de la asignatura se basa en garantizar el entendimiento de que el software es un producto que se encuentra inmerso en un ambiente cambiante. Los cambios producidos en el ambiente sugieren la necesidad de adaptar el producto a dichos cambios. Asimismo, el software es un producto humano y es susceptible de presentar defectos, los cuales deben ser corregidos para mantener el producto en operación.

Los contenidos de la asignatura incluyen desde aspectos conceptuales del proceso de mantenimiento, la definición de mantenibilidad y su relación con el ciclo de vida del producto software, hasta una descripción de algunos de los elementos destacados de la gestión del mantenimiento.

Los elementos relacionados con el mantenimiento del software aquí presentados incluyen la Gestión de Versiones y características de las herramientas de seguimientos de defectos, la Pruebas y su importancia en los proceso de Verificación y Validación del software, elementos asociados a la calidad del software como las guías de estilo de código y su relación con la

mantenibilidad, la reingeniería y la ingeniería inversa como estrategias de modificación de la arquitectura del software, así como la presentación de estándares relacionados con la mantenibilidad del software.

## 1.3. Competencias y resultados de aprendizaje

### COMPETENCIAS BÁSICAS

**CB1.** *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.*

**CB2**. *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.*

**CB3**. *Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.*

**CB4.** *Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.*

**CB5.** *Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía*

### COMPETENCIAS GENERALES

1. Comprender la importancia del mantenimiento del producto software
2. Conocer los principales componentes del proceso de mantenimiento de software
3. Desarrollar habilidades relacionadas con el uso de elementos asociados la mantenimiento de software

### COMPETENCIAS ESPECÍFICAS DE LA ASIGNATURA

1. Capacidad para desarrollar, mantener y evaluar procesos y servicios relacionados con el mantenimiento del software y su relación con el ciclo de vida del software.

2. Capacidad para establecer acuerdos con los clientes en relación al mantenimiento adecuado de los productos software
3. Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar verificar y documentar procesos destinados al mantenimiento del software
4. Capacidad de aplicar principios ingenieriles de desarrollo de software al mantenimiento.

## RESULTADOS DE APRENDIZAJE

Al finalizar esta asignatura se espera que el estudiante sea capaz de:

1. Conocer qué es la gestión de versiones y su importancia en el proceso de desarrollo de software.
2. Comprender el funcionamiento de un Sistema de Control de Versiones, sus componentes y principales características.
3. Conocer los principales Sistemas de Control de Versiones usados en la actualidad, sus características, y flujos típicos de trabajo.
4. comprender los objetivos y la importancia del mantenimiento de software como proceso.
5. Identificar los tipos y características del mantenimiento
6. Identificar los procesos y actividades asociados al mantenimiento
7. Introducir modelos y técnicas de mantenimiento
8. Comprender la importancia del seguimiento de defectos
9. Conocer los lineamientos generales para la elaboración de reportes de defectos.
10. Identificar los componentes de un flujo de trabajo típico de defectos y los tipos de resoluciones
11. Comprender los conceptos de ingeniería inversa y reingeniería de software
12. Conocer los modelos frecuentes para su desarrollo
13. Identificar enfoques de reingeniería
14. Conocer las fases de un proceso de reingeniería
15. Conocer los conceptos, objetivos y elementos más relevantes del estándar
16. Comprender los procesos y roles definidos en el estándar
17. Comprender la necesidad de definición de una estrategia de mantenimiento
18. Comprender la importancia del uso de pruebas en el desarrollo de software
19. Identificar los distintos tipos de prueba de acuerdo al ámbito de ejecución
20. Conocer las características del desarrollo guiado por pruebas
21. Comprender la importancia de la utilización de guías de estilo de código para contribuir con la mantenibilidad del producto
22. Identificar los elementos más importantes de una guía de estilo de código
23. Comprender el concepto de mantenibilidad, sus atributos asociados, y su relación con la calidad de software.

24. Conocer las métricas más relevantes para determinar el grado de mantenibilidad de un producto software.

# 2. Contenidos/temario

Unidad Competencial 1 / Tema 1

1) Gestión de Versiones
2) Componentes de un Sistema de Gestión de Versiones
3) Servicios en Línea

Unidad Competencial 2 / Tema 2

1. El proceso de mantenimiento y ciclo de vida
2. objetivos
3. categorías
4. cuestiones clave
5. procesos
6. técnicas

Unidad Competencial 3 / Tema 3

1. Herramientas de gestión de defectos
2. Seguimiento de defectos
3. guías para la elaboración de reportes
4. resoluciones
5. flujos de trabajo
6. Bugzilla

Unidad Competencial 4 / Tema 4

1. Ingeniería Inversa y Reingeniería
2. conceptos
3. motivaciones
4. objetivos
5. modelos y enfoques
6. fases y riesgos

Unidad Competencial 5 / Tema 5

1. Estándar ISO/IEC 14764:2006

Unidad Competencial 6 / Tema 6

1. Pruebas
2. conceptos
3. inspección
4. casos de prueba
5. tipos de prueba
6. automatización

Unidad Competencial 7 / Tema 7

1. Guías de estilo de código
2. conceptos
3. elementos
4. guías de estilo de lenguajes populares

Unidad Competencial 8 / Tema 8

1. Mantenibilidad
2. conceptos
3. métricas
4. herramientas

# 3. Actividades Formativas

| ACTIVIDAD FORMATIVA | HORAS | PRESENCIALIDAD |
|---|---|---|
| Clases expositivas | 15 | 60 |
| Resolución de ejercicios prácticos | 25 | 30 |
| Tutorías | 20 | 0 |
| Trabajo Autónomo | 90 | 0 |
| | | |

# 4. Metodologías Docentes

**Clases teóricas impartidas como lecciones magistrales o exposiciones, en las que además de presentar el contenido de la asignatura, se explican los conceptos fundamentales y se desarrolla el contenido teórico.**

**Colección de tareas que el alumnado llevará a cabo a lo largo de toda la asignatura, entre las que podemos encontrar: análisis de casos, resolución de problemas, prácticas de laboratorios, comentarios críticos de textos, análisis de lecturas, etc.**

**Sesiones periódicas entre el profesorado y el alumnado para la resolución de dudas, orientación, supervisión, etc.**

**Trabajo tanto individual como grupal para la lectura crítica de la bibliografía, estudio sistemático de los temas, reflexión sobre problemas planteados, resolución de actividades propuestas, búsqueda, análisis y elaboración de información, investigación e indagación, así como trabajo colaborativo basado en principios constructivistas.**

# 5. Evaluación

## 5.1. Sistema de evaluación

El Modelo de Evaluación de estudiantes en la Universidad se sustenta en los principios del Espacio Europeo de Educación Superior (EEES), y está adaptado a la estructura de formación virtual propia de esta Universidad. De este modo, se dirige a la evaluación de competencias.

Es requisito indispensable aprobar el portafolio y la prueba final con un mínimo de 5 para ponderar las calificaciones.

| | | |
|---|---|---|
| **Evaluación 100%** | **Porfolio actividades 50%** | Portafolio Actividad 1 – 6,25% |
| | | Portafolio Actividad 2 – 6,25% |
| | | Portafolio Actividad 3 – 6,25% |
| | | Portafolio Actividad 4 – 6,25% |
| | | Portafolio Actividad 5 – 6,25% |
| | | Portafolio Actividad 6 – 6,25% |
| | | Portafolio Actividad 7 – 6,25% |
| | | Portafolio Actividad 8 – 6,25% |
| | **Prueba final 50%** | |

Es requisito indispensable aprobar TANTO la evaluación continua COMO la final con un mínimo de 5 (aprobado) para superar la asignatura.

Atendiendo a la Normativa de Evaluación de la Universidad, se tendrá en cuenta que la utilización de **contenido de autoría ajena** al propio estudiante debe ser citada adecuadamente en los trabajos entregados. Los casos de plagio serán sancionados con suspenso (0) de la actividad en la que se detecte. Asimismo, el uso de **medios fraudulentos durante las pruebas de evaluación** implicará un suspenso (0) y podrá implicar la apertura de un expediente disciplinario.

## 5.2. Sistema de Calificación

La calificación de la asignatura se establecerá en los siguientes cómputos y términos:

| Nivel de Competencia | Calificación Oficial | Etiqueta Oficial |
|---|---|---|
| Muy competente | 9 - 10 | Sobresaliente |
| Competente | 7 < 9 | Notable |
| Aceptable | 5 <7 | Aprobado |
| Aún no competente | <5 | Suspenso |

El nivel de competencia en cada una de las actividades realizadas se medirá, teniendo en cuenta **criterios generales derivados de la consecución de los resultados de aprendizaje**, que en términos generales y en función de la adecuación en el planteamiento de los contenidos generales y contenidos específicos, valorarán por norma general y en trabajos escritos, la corrección de la estructura formal y organización del discurso (semántica, sintaxis y léxico) valorándose además la originalidad, creatividad y argumentación de las intervenciones utilizando referencias bibliográficas.

**Sin detrimento de lo anterior, el alumnado dispondrá de una rúbrica simplificada que mostrará los aspectos que valorará el docente, como así también los niveles de desempeño que tendrá en cuenta para calificar las actividades vinculadas a cada resultado de aprendizaje.**

# 6. Bibliografía

Apache Software Foundation. (n.d.). Apache Subversion. Retrieved May 10, 2019, from https://subversion.apache.org/

AXELOS. (n.d.). ITIL | IT Service Management | ITSM | AXELOS. Retrieved May 1, 2019, from https://www.axelos.com/best-practice-solutions/itil

Bourque, P., & Fairley, R. E. (2014). SWEBOK Guide V3.0. IEEE Computer Society. https://doi.org/10.1234/12345678

Bugzilla. (n.d.). Bugzilla Documentation — Bugzilla 5.1.2+ documentation. Retrieved July 14, 2019, from https://bugzilla.readthedocs.io/en/latest/

cachéQuality. (n.d.). cachéQuality |. Retrieved July 17, 2019, from https://www.cachequality.com/

Chacon, S., & Straub, B. (2014). Pro Git (2nd ed.). Berkely, CA, USA: Apress.

Driessen, V. (2010). A successful Git branching model » nvie.com. Retrieved May 12, 2019, from https://nvie.com/posts/a-successful-git-branching-model/

Ernst, M. (2012). Version control concepts and best practices. Retrieved May 1, 2019, from https://homes.cs.washington.edu/~mernst/advice/version-control.html

Free Software Foundation. (n.d.-a). CVS - Open Source Version Control. Retrieved May 10, 2019, from https://www.nongnu.org/cvs/

Free Software Foundation. (n.d.-b). RCS. Retrieved May 10, 2019, from http://www.gnu.org/software/rcs/rcs.html

Git. (n.d.). git/git: Git Source Code Mirror. Retrieved May 12, 2019, from https://github.com/git/git

Google. (n.d.). Google Java Style Guide. Retrieved July 17, 2019, from https://google.github.io/styleguide/javaguide.html

ISACA. (n.d.-a). CMMI Institute - CMMI Services. Retrieved May 1, 2019, from https://cmmiinstitute.com/cmmi/svc

ISACA. (n.d.-b). COBIT 2019. Retrieved May 1, 2019, from http://www.isaca.org/COBIT

ISO/IEC. (n.d.-a). ISO/IEC 25010:2011(en), Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Retrieved July 17, 2019, from https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

ISO/IEC. (n.d.-b). ISO/IEC 20000-1:2018 - Information technology -- Service management -- Part 1: Service management system requirements. Retrieved May 1, 2019, from https://www.iso.org/standard/70636.html

ISO, IEC, & IEEE. (2006). ISO/IEC 14764:2006 - Software Engineering — Software Life Cycle Processes — Maintenance. Electronics (Vol. 2006). Retrieved from https://standards.ieee.org/standard/23026-2006.html

MDN. (n.d.). Bug report writing guidelines. Retrieved July 14, 2019, from https://developer.mozilla.org/en-US/docs/Mozilla/QA/Bug_writing_guidelines

Mercurial. (n.d.). Mercurial SCM. Retrieved May 10, 2019, from https://www.mercurial-scm.org/about

Microsoft. (n.d.). Code Metrics – Maintainability Index – The Ultimate Visual Studio Tips and Tricks Blog. Retrieved July 17, 2019, from https://blogs.msdn.microsoft.com/zainnab/2011/05/26/code-metrics-maintainability-index/

Miller, P. (n.d.). Aegis. Retrieved May 10, 2019, from http://aegis.sourceforge.net/

Oman, P., & Hagemeister, J. (n.d.). Metrics for assessing a software system's maintainability. In Proceedings Conference on Software Maintenance 1992 (pp. 337–344). IEEE Comput. Soc. Press. https://doi.org/10.1109/ICSM.1992.242525

Oracle. (n.d.). Code Conventions for the Java Programming Language: Contents. Retrieved July 17, 2019, from https://www.oracle.com/technetwork/java/codeconvtoc-136057.html

PMI. (n.d.). PMBOK Guide and Standards | Project Management Institute. Retrieved May 1, 2019, from https://www.pmi.org/pmbok-guide-standards

Pressman, R. S. (2010). Software Engineering : a practitioner's approach. McGraw Hill Education (India) Private Limited. Retrieved from https://www.amazon.com/Software-Engineering-Practitioners-Approach-7TH/dp/B01N284CFH

ProjectCodeMeter. (n.d.). ProjectCodeMeter - ProjectCodeMeter Software Sizing for Outsourcing Work Hours Assessment and Development Cost Estimation. Retrieved July 17, 2019, from http://www.projectcodemeter.com/cost_estimation/index.html

Python. (n.d.). PEP 8 -- Style Guide for Python Code | Python.org. Retrieved July 16, 2019, from https://www.python.org/dev/peps/pep-0008/#id8

Radon. (n.d.). Welcome to Radon's documentation! — Radon 2.4.0 documentation. Retrieved July 17, 2019, from https://radon.readthedocs.io/en/latest/index.html

Rosenberg, L. H., & Hyatt, L. E. (1997). Software Re-engineering. Retrieved from https://pdfs.semanticscholar.org/d6b2/89019f9fef924fd3300d1094f29c8bc079f9.pdf

Software Freedom Conservancy. (n.d.). Git. Retrieved May 10, 2019, from https://git-scm.com/

Sommerville, I. (2016). Software engineering. Pearson.

Wijnholds, G., Van Eck, P., Van Der Leek, R., Rigal, S., & Visser, J. (2016). Building maintainable software : ten guidelines for future-proof code. O'Reilly Media Inc.