Modelo Guía Docente

1. Identificación de la asignatura

NOMBRE	Diseño del Software			CÓDIGO	GIISOF01-3-004
TITULACIÓN	Ingeniería Informática del Software		CENTRO	Escuela de Ingeniería Informática	
TIPO	Obligatoria	Nº TOTAL DE CRÉDITOS	6		
PERIODO	Semestral		IDIOMA	Español	

COORDINADOR/ES	TELÉFONO /EMAIL	UBICACIÓN
César Fernández Acebal	985 10 33 70 acebal@uniovi.es	Edificio de Ciencias Despacho 153 - Campus de Llamaquique
PROFESORADO	**TELÉFONO /EMAIL**	**UBICACIÓN**
César Fernández Acebal	985 10 33 70 acebal@uniovi.es	Edificio de Ciencias Despacho 153 - Campus de Llamaquique
Raúl Izquierdo Castanedo	985 10 31 72 raul@uniovi.es	Edificio de Ciencias Despacho 238 - Campus de Llamaquique
Aquilino Adolfo Juan Fuente	985 18 26 82 aajuan@uniovi.es	Edif. Polivalente Despacho 2.6.10 Campus de Gijón

2. Contextualización (en el caso de asignaturas compartidas se contextualizará, si existen diferencias, para cada una de las titulaciones donde se comparte).

El diseño de software desempeña un papel fundamental en la construcción de sistemas informáticos complejos, comprendiendo un conjunto de principios y técnicas que permiten pasar de los requisitos a la implementación obteniendo un código de calidad, entendiendo como tal aquél que es fácil de entender y que debiera ser también, por tanto, fácil de modificar para adaptarse a los cambios en los requisitos. Es por ello una pieza esencial en la formación de los graduados en ingeniería del software, aportándoles los conocimientos específicos necesarios para su práctica profesional.

La asignatura, de carácter eminentemente práctico, pertenece al módulo de tecnología específica en ingeniería del software, encuadrándose a su vez en la materia de ingeniería del software, y pretende, por un lado, proporcionar una visión de más alto nivel sobre los conocimientos adquiridos en otras asignaturas de programación, como *Introducción a la Programación*, *Metodología de la Programación* o *Tecnologías y Paradigmas de la Programación*, así como asentar las bases que le posibiliten luego enfrentarse al diseño y construcción de software de dimensión industrial, aplicando las buenas prácticas existentes (concretamente, ciñéndonos fundamentalmente al paradigma orientado a objetos). Igualmente, la asignatura aporta al alumno las competencias necesarias para poder abordar otras asignaturas como *Arquitectura del Software* y *Diseño de Lenguajes de Programación*.

3. Requisitos (en el caso de asignaturas compartidas, si existen diferencias, se señalarán los mismos para cada una de las titulaciones donde se comparte).

El alumno deberá dominar al menos un lenguaje orientado a objetos (en la asignatura, si bien se podrán comentar características aisladas de diferentes lenguajes, se empleará Java como lenguaje de implementación). Sería igualmente deseable que poseyese al menos nociones básicas de diseño orientado a objetos, como modularidad, bajo acoplamiento, alta cohesión, herencia frente a delegación, etcétera, más allá de la sintaxis particular del lenguaje de programación empleado, si bien en la asignatura se repasarán y ampliarán tales conceptos, para subir rápidamente el nivel de abstracción y centrarse en la enseñanza de patrones de diseño, que representan el grueso de esta asignatura.

Concretamente, el alumno deberá haber cursado y superado al menos las asignaturas de *Introducción a la Programación*, *Metodología de la Programación* y *Tecnologías y Paradigmas de la Programación*.

4. Competencias y resultados de aprendizaje (en el caso de asignaturas compartidas, si existen diferencias, se señalarán los mismos para cada una de las titulaciones donde se comparte).

Se espera que el alumno aprenda técnicas avanzadas de diseño orientado a objetos, fundamentalmente patrones de diseño. Al terminar el curso debería no sólo conocer la mayoría de patrones de diseño fundamentales descritos en el libro clásico sobre la materia (Gamma et al. 1995), sino de razonar el diseño de una aplicación en términos de tales patrones, siendo capaz de identificar dónde se hace necesario el uso de un determinado patrón en contextos diferentes a los vistos en clase. Igualmente, se pretende que el curso sirva para asentar principios básicos de diseño orientado a objetos, como identificar correctamente las clases participantes en un diseño y determinar su comportamiento, colaboraciones y responsabilidades. También debería terminar el curso habiendo mejorado su estilo de programación, aprendiendo a usar guías de estilo y principios de factorización de código.

Además de eso, y de acuerdo con la memoria de verificación del título de Grado en Ingeniería Informática del Software de la Universidad de Oviedo, las competencias que el alumno debe adquirir al cursar la asignatura de *Diseño del Software* son las siguientes:

Competencias técnicas o específicas:

- Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social. (Com.2)
- Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados. (Com.8)
- Conocimiento de las características, funcionalidades y estructura de los sistemas operativos y diseñar e implementar aplicaciones basadas en sus servicios. (Com.10)
- 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. (ISW.1)

- 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. (ISW.4)
- 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. (ISW.6)

Competencias generales:

- Competencia para el diseño de soluciones a problemas (CG-1)
- Capacidad de abstracción (CG-3)
- Análisis y síntesis (CG-4)

El alumno será capaz de (resultados de aprendizaje):

- RA.IS-1. Realizar Proyectos de Ingeniería del Software complejos que den solución a problemas reales y solucionarlos mediante técnicas y tecnologías relacionadas con los procesos de fabricación de software incluyendo frameworks, patrones arquitectónicos, patrones de diseño y de integración persiguiendo el desarrollo de software de calidad [Com2], [Com.8], [Com10], [ISw.1], [ISw.4], [ISw.6], [CG1], [CG3],[CG4]
- RA.IS-4. Desarrollar diseños y programación orientados a objetos con un elevado nivel de competencia [Com.8], [ISw.4], [CG1], [CG4]
- RA.IS-5. Evolucionar y refactorizar diseños existentes ante cambios en los requisitos [ISw.1], [ISw.6]
- RA.IS-6. Determinar el grado de mantenibilidad, fiabilidad y eficiencia de diseños software [Com.8], [ISw.1], [ISw.4], [CG1]

5. Contenidos.

En función de los objetivos planteados se seleccionan los siguientes contenidos:

1. Técnicas y principios de calidad para el diseño disciplinado de software
   - Introducción al diseño
   - Representación del diseño
   - Criterios de diseño
   - Paso del diseño a implementación
2. Patrones de diseño
   - Introducción a los patrones
   - Estructura de un patrón
   - Patrones por familia: de creación, estructurales y comportamiento
   - Catálogo de patrones
3. Evolución disciplinada de diseños y reingeniería
   - Los patrones como herramienta de comunicación
   - El papel del diseño en el proceso de desarrollo de software
   - Introducción a la reutilización de diseños mediante *frameworks*
   - El diseño en metodologías ágiles

6. Metodología y plan de trabajo.

La asignatura utilizará la siguiente tipología de modalidades organizativas:

- En las clases expositivas se explicarán los principios y patrones de diseño desde un punto de vista conceptual. Dado el carácter de la asignatura, estás clases serán dialogadas, fomentando el aprendizaje guiado. El profesor a su vez pedirá a los alumnos como trabajo autónomo la lectura de artículos, wikis y de los capítulos correspondientes a los patrones de diseño concretos del libro de texto de la asignatura que serán discutidos en la clase siguiente.
- En los seminarios se plantearán problemas de diseño reales que habrán de ser resueltos en papel por los alumnos, ya sea de manera individual o en grupo, aplicando los principios y patrones expuestos en las clases teóricas, facilitando el aprendizaje por descubrimiento. Posteriormente se resolverán y discutirán entre todos en clase, analizando las ventajas e inconvenientes de las soluciones propuestas.
- En las clases prácticas de laboratorio se implementarán dichas soluciones aplicando los principios y patrones de diseño adecuados para cada problema. En otros casos el enfoque será a la inversa, y se aplicarán patrones de diseño a la resolución de los problemas propuestos antes incluso de que éstos hayan sido estudiados en clase de teoría. Este doble enfoque pretende que los alumnos se den cuenta de que los patrones no son un fin en sí mismos, sino herramientas que, como diseñadores de software, tenemos para dar solución a determinados problemas de diseño.
- En las tutorías grupales se realizará un seguimiento de los alumnos para detectar lagunas y se les orientará sobre cómo solucionarlas.

La siguiente tabla desglosa por temas y horas las actividades anteriormente descritas:

		TRABAJO PRESENCIAL								**TRABAJO NO PRESENCIAL**		
Temas	***Horas totales***	*Clase Expositiva*	*Prácticas de aula /Seminarios/ Talleres*	*Prácticas de laboratorio /campo /aula de informática/ aula de idiomas*	*Prácticas clínicas hospitalarias*	*Tutorías grupales*	*Prácticas Externas*	*Sesiones de Evaluación*	***Total***	*Trabajo grupo*	*Trabajo autónomo*	**Total**
Técnicas y principios de calidad para el diseño disciplinado de software		6	2	8							26	
Patrones de diseño		11	4	14		1		2			47	
Evolución disciplinada de diseños y reingeniería		4	1	6		1					17	
Total		**21**	**7**	**28**		**2**		**2**	**60**		**90**	**90**

MODALIDADES		Horas	%	Totales
Presencial	Clases Expositivas	21	14	60
	Práctica de aula / Seminarios / Talleres	7	4,7	
	Prácticas de laboratorio / campo / aula de informática / aula de idiomas	28	18,7	
	Prácticas clínicas hospitalarias			
	Tutorías grupales	2	1,3	
	Prácticas Externas			
	Sesiones de evaluación	2	1,3	
No presencial	Trabajo en Grupo			90
	Trabajo Individual	90	60	
	Total	150		

De forma excepcional, si las condiciones sanitarias lo requieren, se podrán incluir actividades de docencia no presencial, en cuyo caso se informará a los estudiantes de los cambios efectuados.

7. Evaluación del aprendizaje de los estudiantes.

Como se ha visto en el apartado anterior, en la metodología docente de la asignatura, las diferentes modalidades de clases están sólidamente engranadas, de tal forma que no se concibe la teoría sin la práctica o viceversa. Las prácticas de la asignatura han sido cuidadosamente pensadas para intentar proporcionar el mejor itinerario didáctico posible al alumno, de tal modo que, al final del curso, todas las piezas encajen entre sí y el alumno acabe adquiriendo de manera natural una compresión natural de una materia compleja como ésta y sea capaz de aplicar las técnicas de diseño adquiridas a problemas de ámbitos muy diversos. Por este motivo, y dado el carácter eminentemente práctico de la asignatura, parece lógico que el mayor peso de la nota final recaiga precisamente en la nota de prácticas. Sin embargo, y relacionado con lo anterior, es necesario comprobar que el alumno cuenta al menos con unos conocimientos teóricos mínimos de la materia (difícilmente sabrá que patrón o patrones de diseño aplicar ante un problema dado si no sabe cuáles son éstos).

Por todo ello, la evaluación en la **convocatoria ordinaria** comprenderá las siguientes actividades:

- Semanalmente en clase de prácticas se planteará un determinado problema de diseño. En unos casos se deberá realizar desde cero y, en otros, se proporcionará un código que funciona pero que adolece de problemas de diseño que el alumno deberá, primero, detectar, y luego intentar resolver con los principios generales de diseño y, en su caso, los patrones de diseño vistos en clase de teoría. Cualquier entrega fuera de plazo o en otra tarea del campus virtual distinta a la establecida para cada grupo y sesión de laboratorio no será tenida en cuenta. Es fundamental hacer notar que lo importante de dichas prácticas, y su utilidad como herramienta de evaluación y aprendizaje, no es tanto la corrección de las mismas como que el alumno se haya esforzado en su realización, aun cuando no haya sido capaz de llegar a la solución «ideal». Es más, en lo concerniente a esta materia, puede llegar a ser más didáctico que el alumno yerre inicialmente y no sea capaz de dar con dicha solución puesto que, a cambio, cuando en la siguiente sesión de

prácticas se explique la solución correcta, rápidamente será capaz de detectar las bondades del patrón correspondiente en términos de flexibilidad del código frente a su solución «sin patrones». Por este motivo dichas prácticas no llevarán asociada una nota numérica como tal, sino únicamente dicha calificación de apto o no apto en función del esfuerzo realizado, pues de lo que se trata es de que el alumno las realice libremente, por su cuenta, sabiendo que no es la corrección del diseño final lo que se tendrá en cuenta. Como se ha dicho, en la siguiente sesión de prácticas se dedicará la primera parte de la clase a comentar en clase las distintas soluciones propuestas por los estudiantes debatiendo entre todos las virtudes y defectos de cada, y se corregirá el ejercicio, dejando disponible a los estudiantes el código fuente de la solución para que puedan compararlo con la suya y hacer los cambios oportunos en su diseño e implementación. Este último punto (dejar el código con la solución a disposición del alumno) consideramos que es una parte esencial del aprendizaje en esta asignatura, y es otro motivo por el que difícilmente se podrían evaluar con una calificación numérica unas prácticas de las que el alumno ya cuenta con la solución.

- En la convocatoria ordinaria, únicamente aquellos alumnos que, **habiendo asistido a al menos un 80 % de las clases prácticas**, cuenten así mismo con al menos **dicho porcentaje de evaluaciones positivas de sus entregas de prácticas semanales**, tendrán derecho a presentarse a un examen práctico consistente en la realización de un ejercicio similar a los planteados en las diferentes sesiones de laboratorio. Se entiende que en dicho examen se evalúa por un lado, y junto a la mencionada calificación de apto o no, el trabajo semanal realizado (que realmente las prácticas presentadas hayan sido hechas de forma autónoma por el alumno), esto es, representa el instrumento con el que cuenta el profesor para la *evaluación formativa y continua en la realización de practicas*, y, por otro, tendrá también la consideración de *evaluación final de practicas de laboratorio*, teniendo un peso global del **60 % en la nota final**, y en el que será necesario obtener una **nota igual o superior a cinco** para superar la asignatura.
- Por otro lado, habrá un examen teórico previo que constará o podrá constar tanto de preguntas de tipo test como de preguntas teóricas a desarrollar, así como de pequeños ejercicios de diseño similares a los vistos en clases de seminario. Dicho examen supondrá el **40 % de la nota final** y, solamente en el caso de la convocatoria ordinaria y para aquellos alumnos que cumplan con el mencionado requisito de asistencia y entrega de prácticas, se deberá obtener una **nota igual o superior a cuatro** para poder hacer media con la evaluación de la parte práctica (en el resto de convocatorias se deberá sacar un cinco en el examen de teoría para poder superar la asignatura). A aquellos alumnos que, cumpliendo los requisitos para la evaluación continua, hayan superado el mínimo en teoría pero suspendan sin embargo la evaluación de prácticas, se le dará la opción de conservar dicha nota de teoría para las convocatorias extraordinarias del presente curso académico, si así lo desean, debiendo examinarse en futuras convocatorias únicamente de la parte práctica. La nota de prácticas no se conserva en ningún caso.

Cumplidos todos los requisitos anteriores, la nota final de la asignatura en la convocatoria ordinaria se calculará de acuerdo con la siguiente fórmula:

Nota final = Nota de teoría × 40 % + Nota de prácticas × 60 %

No obstante, y en previsión de que los alumnos, por la razón que sea, no puedan cumplir el porcentaje mínimo de asistencia requerido o no hayan entregado satisfactoriamente sus prácticas semanales, dichas condiciones desaparecen para las **convocatorias extraordinarias**, en las

que la evaluación constará de un examen teórico y un examen práctico que podrá versar sobre cualquiera de los patrones de diseño estudiados en la asignatura (no necesariamente los vistos en clase de prácticas). Se deberá obtener al menos un cinco en ambas pruebas y la nota final se calculará al igual que en la convocatoria ordinaria, esto es:

Nota final = Nota de teoría × 40 % + Nota de prácticas × 60 %

En el caso de aquellos alumnos a los que se les conceda el **sistema de evaluación diferenciado**, el requisito de asistencia desaparece, y podrán presentarse en la convocatoria ordinaria a ambos exámenes, debiendo obtener en ambos al menos un cinco y calculándose la nota del mismo modo que en los casos anteriores. Para la convocatoria extraordinaria no hay cambios con respecto al resto del alumnado.

En ambas convocatorias (ordinaria y extraordinaria), en el caso de los alumnos que se hayan presentado a ambos exámenes pero que no hayan alcanzado el mínimo exigido en cualquiera de los dos, la nota final se calculará según la siguiente fórmula:

Nota final = mínimo(4, nota según la fórmula anterior)

Por último, los alumnos que no se presenten a alguna de las dos pruebas figurarán como **no presentado**.

De forma excepcional, si las condiciones sanitarias lo requirieren, se podrán incluir métodos de evaluación no presencial, en cuyo caso se informará a los estudiantes de los cambios efectuados.

8. Recursos, bibliografía y documentación complementaria.

Éste será el libro de texto básico de la asignatura:

- Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. 1995. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley. Traducción al español de César Fernández Acebal y Juan Manuel Cueva Lovelle: *Patrones de diseño: Elementos de software orientado a objetos reutilizable*. Pearson Educación, 2003

Así mismo, son recomendables los siguientes libros para distintas partes de la asignatura:

- Martin Fowler. 1999. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley
- Martin Fowler. 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Addison-Wesley Professional
- Robert C. Martin. 2008. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall
- Steve McConnell. 2004. *Code Complete: A Practical Handbook of Software Construction* (2nd ed.). Microsoft Press
- Joshua Bloch. 2008. *Effective Java (2nd Edition)*. Addison-Wesley
- Elisabeth Freeman, Eric Freeman, Bert Bates and Kathy Sierra. 2004. *Head First Design Patterns*. O'Reilly Media
- Joshua Kerievsky. 2004. *Refactoring to Patterns*. Addison-Wesley Professional
- Steve Maguire. 1993. *Writing Solid Code*. Microsoft Press

Y éstas son dos buenas referencias clásicas en línea sobre patrones de diseño:

- http://c2.com/cgi/wiki?DesignPatterns
- http://hillside.net/patterns/