Guía Docente
PROGRAMACIÓN ORIENTADA A OBJETOS

Autores:

- Cristina Cachero Castro
- Estela Saquete Boró

Diciembre 2005

ÍNDICE

ÍNDICE 3

1. Contextualización 5

1.1. Perfil de los créditos de la materia. Adecuación al perfil profesional y académico de la titulación 5

La Programación OO en el EEES 10

1.2. Ubicación y relaciones en el plan de estudios 11

2. Objetivos 15

2.1 Objetivos generales 15

2.2 Competencias académicas y profesionales 15

3. Prerrequisitos 19

3.1 Competencias y contenidos mínimos 19

3.2 Plan de trabajo y actividades para la consecución de los prerrequisitos 19

4. Bloques y temas de contenido 21

4.1 Bloques de contenidos de aprendizaje 21

4.2 Temas o unidades de contenido. Desarrollo 21

5. Metodología y estrategias de aprendizaje 23

5.1 Metodología docente 23

5.2 Estrategias de aprendizaje 23

6. Plan de trabajo de los alumnos: especificación del tiempo y esfuerzo del aprendizaje 25

6.1 Planificación del programa de Teoría 25

6.2 Planificación del programa de Prácticas 25

6.3 Planificación (resumen) 26

7. Bibliografía y materiales recomendados 27

7.1 Bibliografía básica 27

7.2 Bibliografía complementaria 27

7.3 Otros recursos 27

8. Evaluación de los procesos y resultados de aprendizaje 29

8.1 Procedimiento de evaluación 29

8.2 Criterios de evaluación 29

9. Análisis de coherencia de la guía docente 31

Borrador de Anexos para la memoria final de la red 39

Anexo A: Clasificación de las asignaturas del Plan de estudios vigente en la Universidad de Alicante según subcategorías del Libro Blanco (Contenidos Específicos de la Ingeniería) 41

Anexo B: Estimación de créditos ECTS para POO 43

1. Contextualización

Actualmente los títulos en Informática pertenecen al área de las Tecnologías de la Información y Comunicaciones (TIC). En España, la profesión Informática está muy difuminada, ya que existe una multiplicidad de títulos relacionados directamente con la disciplina, que da lugar a profesionales con formaciones muy dispares. Aún más, profesionales formados en otras disciplinas como por ejemplo Ingeniería en Telecomunicaciones, Ingeniería Industrial, Licenciados en Ciencias Físicas y en Matemáticas, etc., también trabajan en este sector, por lo que no se puede hablar de un perfil unificado.

El intento de justificar esta disparidad de títulos en el área de las TIC mediante el establecimiento de unos contenidos básicos y unas atribuciones profesionales bien definidas, diferenciadas y lo suficientemente estancas, ha resultado infructuoso. Este hecho ha provocado que, en el nuevo marco europeo, se haya optado por la reducción a dos únicas titulaciones, que se relacionan con los campos propios de la Informática y de las Telecomunicaciones respectivamente. En este marco, los objetivos primordiales en la formación de un Ingeniero en Informática hacen referencia tanto al ámbito cognoscitivo como a las habilidades y aptitudes que permiten aplicar los conocimientos adquiridos en el ejercicio de la profesión. Así, un Ingeniero en Informática debería ser capaz de abordar problemas desconocidos y adaptarse a la rápida evolución del sector.

0.0. Perfil de los créditos de la materia. Adecuación al perfil profesional y académico de la titulación

Los lenguajes de programación han sido desde siempre piedra angular de todas las propuestas docentes en titulaciones relacionadas con la informática. Dichas propuestas han ido evolucionando a medida que se ha producido un desarrollo histórico de los lenguajes de programación, desarrollo que ha estado estrechamente relacionado con el desarrollo del hardware y software de los ordenadores. Actualmente, es muy común hablar de generaciones de ordenadores y, de una forma similar, de generaciones de lenguajes de programación. La historia de los lenguajes de programación puede ser dividida en cuatro (o cinco, según algunos autores) generaciones. Las dos primeras generaciones siguen una sucesión en el tiempo, mientras que las dos últimas siguen evolucionado en la actualidad de forma casi paralela, debido a que propugnan distintos paradigmas, cada uno de ellos especialmente pensado para la resolución de un tipo de problema distinto.

Estas generaciones son:

- *Primera generación:* Años previos a la segunda guerra mundial. Problemas numéricos. Lenguajes máquina y ensamblador.
- *Segunda Generación*. Principio de los años cincuenta. Notaciones simbólicas. FORTRAN (1957), ALGOL (década de los sesenta), COBOL (finales de los cincuenta), LISP (principio de los sesenta), BASIC (1965). BASIC fue el primer lenguaje incluido en el software distribuido con los primeros ordenadores personales.
- *Tercera generación*. La creciente complejidad, tamaño e importancia de los sistemas y aplicaciones, las necesidades de fiabilidad en los programas y, al mismo tiempo, la falta de adecuación de los métodos tradicionales a dichas necesidades provoca a finales de los sesenta la denominada *crisis del software*. Como respuesta surgen los conceptos de abstracción funcional, método de diseño descendente (top-down), y se propone un conjunto de principios de programación estructurada (Dijstra) que se aplican a nuevos lenguajes: PASCAL (Wirth, 1971). C (lenguaje en el que se desarrolló el sistema operativo UNIX a principios de los setenta), paradigma lógico (PROLOG (1972)) y **paradigma orientado a objetos** (SIMULA (1967), Smalltalk (1980), Eiffel (1986), C++ (1986), Java (1995), …). Este paradigma modifica el tradicional enfoque a procesos por un enfoque a datos, e introduce nuevos conceptos como los de clase, objeto y herencia.

- *Cuarta Generación*. Se sigue aumentando el nivel de abstracción. Aparecen lenguajes que pretenden acelerar el proceso de construcción de aplicaciones y facilitar su mantenimiento, incluso el de posibilitar que el propio usuario final pueda resolver sus propios problemas usando él el lenguaje de programación. Estos lenguajes sobre todo están orientados a la gestión y en especial al tratamiento de bases de datos. Algunos de estos lenguajes de programación tienen su propia estructura y algunos otros realmente lo que hacen es el de permitir al programador crear programas en un lenguaje de tercera generación utilizando muchas menos sentencias.

Esta evolución se ha visto plasmada en el número y tipo de materias propuestas (relacionadas con la programación) por las principales recomendaciones curriculares internacionales de Informática. De entre ellas, quizás una de las de más impacto sea el Computing Curricula. En su versión del 2001 [ACM2001] ya se enfatizaba que uno de los cambios fundamentales respecto a versiones anteriores era el aumento significativo de la importancia de ciertos temas curriculares, entre los que destacaba la Programación Orientada a Objetos. De hecho, en su definición del Cuerpo de Conocimiento de Computer Science la Programación orientada a objetos aparece como un *core topic*, con un número de horas asignadas igual al asignado a la enseñanza de los Algoritmos de Computación Básicos, por poner un ejemplo. Esta importancia está plenamente justificada. Por un lado el paradigma orientado a objetos ha demostrado sobradamente su grado de madurez y la viabilidad de sus propuestas. Por otro, la aplicación de este paradigma supone un cambio cualitativo en el modo de construir aplicaciones ya que, tal y como enfatiza Meyer [MEYER97], proporciona mecanismos que permiten aumentar la calidad del producto software resultante (aplicaciones más correctas, más robustas, más extensibles y más reutilizables).

La importancia del aprendizaje de los distintos paradigmas de programación en las titulaciones de Informática se refleja de manera muy clara en la última revisión de los distintos perfiles que componen esta recomendación (Computing Curricula 2004 [ACM2004]). En esta recomendación se proponen cinco grandes perfiles genéricos de Informática: Computer Science[CC2001], Computer Engineering[CE2004], Software Engineering[SE2004], Information Systems[IS2002] e Information Technology[IT2005]. De entre ellos, el área de conocimiento Programming Fundamentals (dentro de la cual podemos englobar la Programación Orientada a Objetos) aparece con relevancia máxima (nivel 5) en los perfiles Computer Science, Computer Engineering y Software Engineering, que son precisamente los perfiles en los que se enmarcan las actuales titulaciones de Informática en España. Esto implica que los nuevos planes de estudio deberían poner el máximo énfasis (a través de las distintas asignaturas de especialización ofertadas por cada plan más allá de los mínimos exigibles por el currículo de grado) en la enseñanza de esta área de conocimiento en relación con el resto de áreas y perfiles impartidos.

Por otro lado hay que tener en cuenta (tal y como hace el Computing Curricula 2004) que la Programación Orientada a Objetos implica no sólo un lenguaje que soporte clases, objetos, herencia y polimorfismo sino una nueva forma de diseñar el programa, y de hecho nos encontramos con que su enseñanza se afianza a menudo dentro del marco de asignaturas relacionadas con la disciplina de Ingeniería del Software. A decir verdad, existe cierta controversia en cuanto a si se debería enseñar antes Ingeniería del Software y sólo más adelante introducir la programación o si, por el contrario, se debería comenzar con cursos de programación y en cursos posteriores profundizar en conceptos propios de la Ingeniería del Software. Igualmente, existen distintas propuestas en cuanto al orden en que se deberían enseñar los distintos paradigmas: mientras unos autores propugnan comenzar por el paradigma imperativo, otros proponen comenzar directamente con el paradigma orientado a objetos. En cualquier caso, tal y como se afirma en el perfil de Software Engineering "… La programación es una habilidad básica requerida por todos los ingenieros software; también es una habilidad que requiere de mucha práctica para su correcta adquisición. Mientras más pronto los estudiantes practiquen la programación, mayor habilidad adquirirán…" En este perfil [SE2004] la programación orientada a objetos aparece de manera explícita en las siguientes asignaturas:

- **CS102I (The Object-Oriented Paradigm)**: Introduce el concepto de programación orientada a objetos a alumnos con un bagaje en el paradigma imperativo. El curso comienza con una revisión de las estructuras de control y los tipos de datos, con énfasis en los tipos de datos estructurados y en el procesamiento de arrays. Posteriormente introduce el paradigma orientado a objetos, focalizándose en la definición y uso de clases junto con los fundamentos del diseño orientado a objetos. Otros temas incluyen una visión general de los principios generales de algún lenguaje de programación, análisis simples de algoritmos, técnicas básicas de ordenación y búsqueda, y una introducción de principios de ingeniería del software.

  De los contenidos propuestos para CS102I, todos los referidos a la programación orientada a objetos (diseño orientado a objetos, encapsulación y ocultación de información, separación de comportamiento e implementación, clases, subclases y herencia, polimorfismo, jerarquías de herencia) están cubiertos en la asignatura objeto de este estudio.

- **CS 103 (Data Structures and Algorithms):** basada en los fundamentos proporcionados por la secuencia de asignaturas CS101I-102I, esta asignatura introduce los conceptos fundamentales de estructuras de datos y los algoritmos que proceden de ellas. Algunos de los temas tratados son recursividad, la filosofía subyacente de la programación orientada a objetos, los fundamentos de las estructuras de datos (pilas, colas, listas enlazadas, tablas asociativas, árboles y grafos), loas bases del análisis de algoritmos y una introducción a los principios de traducción de lenguajes.

  Además de una profundización en los conceptos OO introducidos en la asignatura anterior, esta asignatura proporciona una visión general de los lenguajes y paradigmas de programación, que es un tema también incluido en nuestra asignatura.

- **SE201 (Introduction to Software Engineering)**: curso que presenta los principios y conceptos básicos de la Ingeniería del Software y da una base para muchos otros cursos posteriores. Cubre de manera extensa la mayor parte de la terminología y conceptos de la ingeniería del software. Al terminar el curso, los alumnos serán capaces de realizar un análisis y diseño básico OO, particularmente utilizando UML. Además, tendrán un conocimiento básico de requisitos, arquitectura software y testeo.

  La conexión entre diseño orientado a objetos utilizando UML y su implementación es un tema cubierto de manera extensiva por la asignatura objeto de este estudio.

- **SE312 (Low-Level Design of Software)**: esta asignatura cubre el diseño y la construcción detallada del software. También cubre de manera profunda los patrones de diseño y la refactorización. Se introducen aproximaciones formales al diseño. Se analizan diseños basados en criterios de calidad internos. Se mejora el rendimiento y la mantenibilidad de programas. Se realiza ingeniería inversa. Se presentan aproximaciones disciplicandas al cambio en el diseño.

  En esta asignatura, entre otras cosas el alumno aprende a utilizar patrones de diseño, así como a programar con un alto nivel de eficiencia. Estos objetivos de aprendizaje están también presente en la asignatura objeto de este estudio.

Por último, destacar que en el perfil de Software Engineering las distintas posibilidades en cuanto al orden de aprendizaje de estas asignaturas se reflejan en distintos ‘itinerarios’, es decir, distinta ordenación temporal de las asignaturas en el plan de estudios.

Versiones anteriores de estas recomendaciones han influido en la elaboración de las directrices generales propias de las titulaciones de informática (Reales Decretos 1459/1990 1460/1990 1461/1990 del 26 Octubre, BOE 20 Noviembre 1990). Lamentablemente, debido a su momento de elaboración, en estos reales decretos no se incluye ningún epígrafe específico para la Programación Orientada a Objetos. Según estas directrices, las materias más relacionadas con la asignatura Programación Orientada a Objetos, son:

- Metodología y Tecnología de la Programación: materia troncal (y por tanto de obligatoria inclusión) para todas las titulaciones en todos los planes de estudios conducentes a la obtención de los títulos de Ingeniero Informático, Ingeniero Técnico en Informática de Gestión, Ingeniero Técnico en Informática de Sistemas. 15 créditos. Contenido: Diseño de Algoritmos, Análisis de Algoritmos, Lenguajes de Programación, Diseño de programas: descomposición modular y documentación, Técnicas de verificación y pruebas de programas.

- Estructura de Datos y de la Información: materia troncal (y por tanto de obligatoria inclusión) para todas las titulaciones en todos los planes de estudios conducentes a la obtención de los títulos de Ingeniero Informático, Ingeniero Técnico en Informática de Gestión, Ingeniero Técnico en Informática de Sistemas. 12 créditos. Contenidos: Tipos Abstractos de Datos, Estructura de datos y algoritmos de manipulación, Estructuras de información: Ficheros, Bases de datos.

Siguiendo estas directrices, en la Universidad de Alicante, la materia troncal *Metodología y Tecnología de la Programación* se estructura en las siguientes asignaturas, troncales para todas las titulaciones, definidas por la universidad:

- *Fundamentos de Programación I.* Asignatura básica que facilita una visión inicial del campo de la programación a través del estudio de los elementos básicos de un lenguaje de programación. Primer curso. Primer cuatrimestre.
- *Fundamentos de Programación II.* Asignatura que facilita la aplicación de los conceptos generales de un lenguaje mediante el estudio y uso de un lenguaje de programación de tipo imperativo. Primer curso. Segundo Cuatrimestre.
- *Diseño y Análisis de Algoritmos.* Asignatura que estudia los conceptos de diseño análisis y verificación de algoritmos y sus técnicas asociadas. Tercer curso. Primer cuatrimestre.

Las tres asignaturas tienen actualmente una carga docente de 6 créditos, lo que hacen un total de 18 créditos, es decir, tres más de los quince mínimos que establecen los Reales Decretos.

Asimismo, en la Universidad de Alicante la materia troncal Estructura de datos y de la Información se materializa en dos asignaturas anuales de 9 créditos cada una:

- *Programación y Estructuras de Datos*. Asignatura que introduce el diseño recursivo, así como los fundamentos del paradigma orientado a objetos de una manera pragmática, a través del estudio de los principales TAD's (pilas, colas, listas, árboles y grafos).
- *Bases de Datos I*. Asignatura básica que proporciona una visión general de los principios subyacentes de las bases de datos relacionales, y que introduce tanto el cálculo relacional como el modelo EER.

Ambas asignaturas se imparten en segundo curso y, debido a su carga docente, son anuales. Juntas, suman los 18 créditos mínimos establecidos para esta materia troncal en los Reales Decretos de las tres titulaciones.

A pesar de esta ausencia significativa del contenido Programación Orientada a Objetos en los Reales Decretos españoles, cabe destacar cómo los planes de estudio de la Universidad de Alicante ponen los medios para subsanar esta carencia mediante la oferta de un número elevado de asignaturas obligatorias, con una carga de 4,5 créditos cada una, que complementan los contenidos de las materias troncales. Dichas asignaturas son:

- **Programación Orientada a Objetos.** Asignatura que introduce de una manera sistemática los principios del paradigma orientado a objetos, y profundiza en los conceptos de clase, objeto, mensaje, herencia, polimorfismo, gestión de errores y persistencia. Segundo curso. Primer cuatrimestre.
- Herramientas de Programación. Asignatura que introduce los entornos de desarrollo, los estándares de nomenclatura, la organización, indentado y comentario del código fuente, la gestión de proyectos, la programación por contrato, la internalización de aplicaciones y el control de versiones. Segundo curso. Primer cuatrimestre.

- Lenguajes y Paradigmas de Programación. Asignatura que da una visión general de los distintos paradigmas de programación, sus puntos fuertes y sus debilidades, y proporciona al alumno una base crítica sobre la cual discernir qué lenguaje puede adecuarse mejor a cada proyecto. Segundo curso. Segundo cuatrimestre.
- Computabilidad: Asignatura que, como su nombre indica, estudia la computabilidad los algoritmos, y hace especial énfasis en máquinas de Turing y funciones recursivas. Segundo curso. Segundo Cuatrimestre.
- Diseño y Programación Avanzada de Aplicaciones. Asignatura que profundiza en el paradigma orientado a objetos, e introduce los objetos distribuidos, las aplicaciones cliente/servidor y las aplicaciones para Internet. Tercer curso. Segundo Cuatrimestre.
- Algoritmia Avanzada. Asignatura que profundiza en conceptos presentados en Diseño y Análisis de Algoritmos, como son los algoritmos de búsqueda exhaustiva y estocástica, programación dinámica y algoritmos de codificación y compresión. Cuarto Curso. Primer Cuatrimestre.

Al analizar la adecuación del perfil de la asignatura a los perfiles de la titulación, debemos destacar que la Programación Orientada a Objetos juega un papel fundamental a la hora de construir sistemas software donde la calidad, fiabilidad y productividad deben estar garantizadas. Los descriptores de esta asignatura cubren los fundamentos de este paradigma, y son básicos para la formación de un informático independientemente de su perfil, ya que cubren un conjunto de técnicas de abstracción básicas para el diseño y la implementación de sistemas de información complejos con un alto grado de fiabilidad y reuso.

Por tanto el contenido de la Programación Orientada a Objetos es relevante en los tres perfiles profesionales generales [EICE04]:

- **Perfil profesional de desarrollo de software**: un ingeniero con este perfil "…sabe diseñar la arquitectura y detallar las especificaciones de funcionamiento; conoce la naturaleza y posibilidades de los distintos lenguajes de codificación y es capaz de realizar la implementación, de todo o parte del producto, mediante el uso de las diferentes metodologías y paradigmas de desarrollo que estén a su alcance; está preparado para realizar la verificación modular de los desarrollos parciales, la integración parcial o completa y las pruebas modulares y de sistema; está en disposición de validar el producto para la aceptación del cliente, de implantarlo y de ponerlo en explotación…" [EICE04]. Tal y como indica esta descripción, la programación orientada a objetos es fundamental en este perfil, ya que es el paradigma predominante en la actualidad para el desarrollo software y aplicaciones de tamaño medio.

- **Perfil profesional de sistemas**: "…un Ingeniero en Informática con perfil Sistemas, es capaz de especificar, modelar, diseñar, implantar, verificar, integrar, configurar, mantener y evaluar el rendimiento de cualquier sistema informático así como cada uno de sus componentes o partes. Por ello debe contar con sólidos conocimientos de las técnicas, dispositivos y herramientas propias del ámbito que le capaciten para la especificación, diseño, montaje, depuración, mantenimiento y evaluación del rendimiento del hardware de computadores y sus periféricos habituales. Asimismo, debe ser competente para el desarrollo del software del sistema que posibilita una gestión eficaz de los recursos hardware del sistema informático..." [EICE04] La Programación Orientada a Objetos proporciona de manera natural mecanismos de integración, seguridad etc que facilitan el desarrollo de sistemas distribuidos, y de hecho es un paradigma que está en la base de muchas de las plataformas de desarrollo actuales (J2EE, .NET, etc).

- **Perfil profesional de gestión y explotación de tecnologías de información**: "…Un Ingeniero en Informática con perfil profesional de Gestión y Explotación de Tecnologías de la Información es responsable de asegurar que las necesidades de Gestión de la Información y del Conocimiento de las organizaciones se satisfacen con el desarrollo y la implantación de soluciones informáticas. Conoce la estrategia empresarial y las diferentes soluciones de las Tecnologías de la Información y de las Comunicaciones

necesarias para apoyar dicha estrategia..". Aunque quizás este sea el perfil en que menos importancia relativa tenga la Programación Orientada a Objetos, un ingeniero de estas características debe poder comunicarse con el ingeniero software en términos comprensibles para ambos, y el paradigma orientado a objetos proporciona una buena base para la discusión de la arquitectura de los sistemas a desarrollar.

Por último, cabe destacar que la utilización en esta asignatura de artefactos aprendidos en otras asignaturas que cubren el área de Ingeniería del Software ayuda a comprender el proceso de desarrollo de software, al mismo tiempo que enfatiza la utilidad de los artefactos software (en nuestro caso diagramas UML) como vehículo de reflexión y comunicación entre programadores.

La Programación OO en el EEES

La asignatura Programación Orientada a Objetos es además perfectamente coherente con la propuesta del Libro Blanco que recoge los cambios que deben efectuarse en los planes de estudio de las titulaciones universitarias de informática en el nuevo espacio europeo de educación superior [EICE04]. Muy sucintamente, a partir del 2010 se prevé que exista una sola titulación de Grado denominada Ingeniería en Informática, que recoge los contenidos generales y básicos de la enseñanza superior informática. Estos estudios de Grado constarán de 240 créditos ECTS y están organizados en 4 años. Los contenidos Formativos Comunes de la titulación representan un 60% de la carga de los estudios, incluyendo la carga asignada al Proyecto Fin de Carrera, dejando el 40% restante para materias ofertadas discrecionalmente por cada Universidad. El título de Grado dará acceso a un segundo ciclo de carácter puramente profesional, denominado Master. El Master está destinado a la especialización profesional de los Ingenieros en Informática, o bien a su preparación para la investigación. Los estudios de Master constarán de entre 60 y 120 créditos ECTS, y permitirán el acceso a la realización de la tesis doctoral con el objeto de obtener el grado de Doctor.

Los Contenidos Formativos Comunes (CFC), que representan el 60% de la carga de los estudios (al menos 144 ECTS de los 240 de los que consta la titulación), y dentro de los que se engloba la materia objeto de este estudio, quedan distribuidos de la siguiente forma:

1) Fundamentos científicos (10% - 15%)
   1.1) Fundamentos matemáticos de la Informática
   1.2) Fundamentos físicos de la Informática

3) Contenidos Generales de la Ingeniería (5% - 10%)
   3.1) Gestión de las organizaciones
   3.2) Ética, legislación y profesión
   3.3) Destrezas profesionales

2) Contenidos específicos de la Ingeniería en Informática (35% - 40%)
   2.1) Programación
   2.2) Ingeniería del Software, Sistemas de Información y Sistemas Inteligentes
   2.3) Sistemas operativos, Sistemas Distribuidos y Redes
   2.4) Ingeniería de Computadores

4) Proyecto Fin de Carrera (6%)

El libro blanco detalla además los contenidos de cada categoría. En concreto, para los contenidos específicos se especifica lo siguiente:

- Programación (P): fundamentos y metodología de la programación, algoritmia, computabilidad, lenguajes de programación, paradigmas de programación, estructuras de datos.
- Ingeniería del Software, Sistemas de Información y Sistemas Inteligentes (IS): desarrollo de software, procesos, requisitos, especificación y diseño, gestión de

proyectos, calidad del software, interacción persona-computadora, bases de datos, inteligencia artificial.

- Sistemas Operativos, Sistemas Distribuidos y Redes (SO): sistemas operativos, sistemas distribuidos, sistemas de tiempo real, arquitectura e infraestructura de redes y servicios telemáticos, seguridad.
- Ingeniería de los computadores (IC): fundamentos, estructura y arquitectura de computadores, tecnología de computadores.

La materia de Programación Orientada a Objetos (POO) se engloba, dentro de los Contenidos Específicos de la Ingeniería en Informática, en las subcategoría Programación, donde cubre parcialmente los epígrafes fundamentos y metodología de la programación, paradigmas de programación y lenguajes de programación. De hecho, en el libro blanco se identifica la Programación Orientada a Objetos como un cambio "revolucionario", que influye de forma determinante en la enseñanza de la Informática, junto con otras áreas como la web, las nuevas tecnologías de red, los gráficos y multimedia, las técnicas de simulación, los sistemas empotrados, las bases de datos relacionales o el uso de sofisticados interfaces para el programador de aplicaciones.

En este contexto, todas las asignaturas relacionadas con la Programación Orientada a Objetos forman parte de los Contenidos Específicos, subcategoría Programación, tal y como mostramos en la tabla 1

EPÍGRAFE	ASIGNATURA
Fundamentos y metodología de la programación	**FUNDAMENTOS PROGRAMACIÓN I**
	FUNDAMENTOS PROGRAMACIÓN II
	HERRAMIENTAS DE PROGRAMACIÓN
Algoritmia, computabilidad	**DISEÑO Y ANÁLISIS DE ALGORITMOS**
	ALGORITMIA AVANZADA
	COMPUTABILIDAD
Lenguajes de programación	**LENGUAJES, GRAMÁTICAS Y AUTÓMATAS**
	PROCESADORES DEL LENGUAJE
Paradigmas de programación,	**LENGUAJES Y PARADIGMAS DE PROGRAMACIÓN**
	PROGRAMACIÓN ORIENTADA A OBJETOS
	DISEÑO Y PROGRAMACIÓN AVANZADA DE APLICACIONES
Estructuras de datos	**PROGRAMACIÓN Y ESTRUCTURAS DE DATOS**

Tabla 1. Asignaturas que recogen los Contenidos Formativos Comunes de Programación.

0.0. Ubicación y relaciones en el plan de estudios

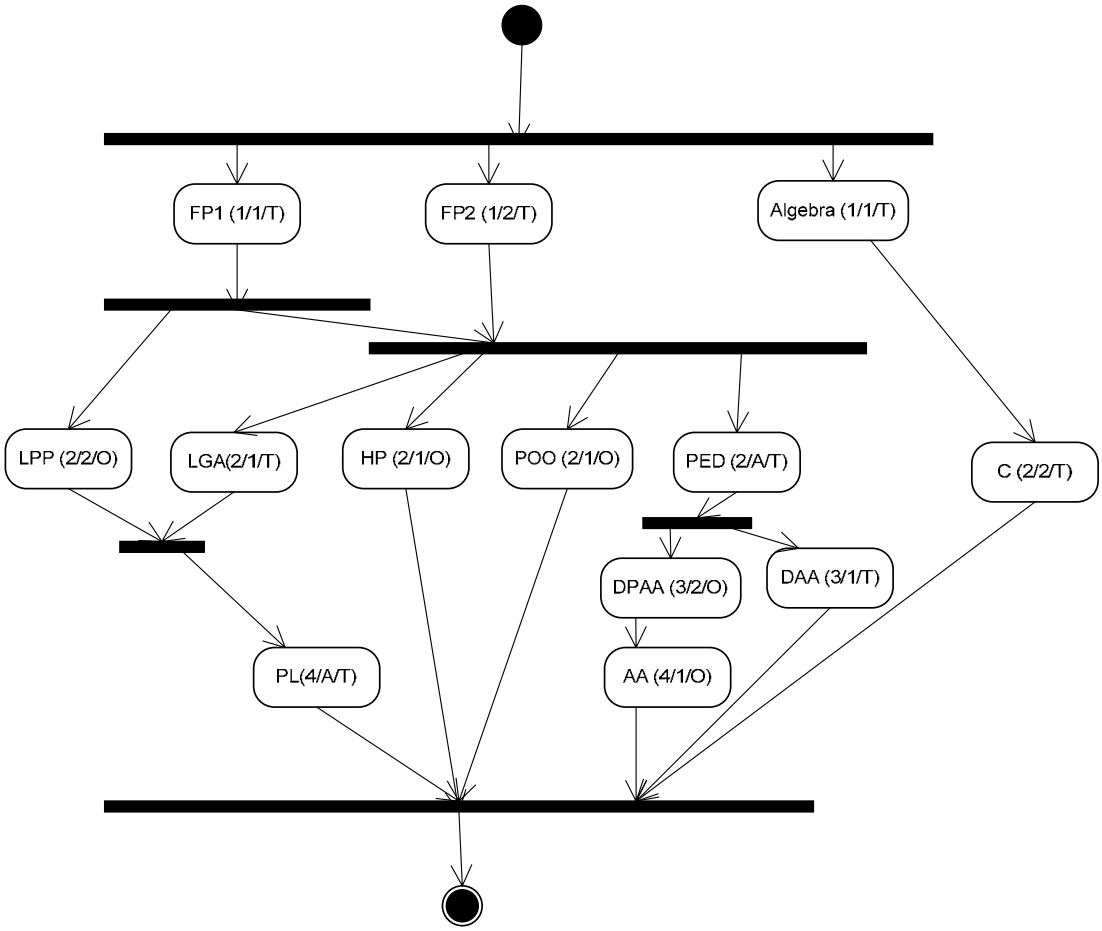
La asignatura Programación Orientada a Objetos es obligatoria en el segundo curso de las tres ingenierías ofertadas por la Universidad de Alicante. Esta asignatura está estrechamente relacionada con otras asignaturas troncales, obligatorias y optativas, que presentamos en la tabla 1 junto con sus correspondientes descriptores. En esta tabla, la primera columna indica el curso en el que se imparte la asignatura (en caso de ser troncal u obligatoria) y el cuatrimestre (primero, segundo o asignatura anual)

CURSO/ CUATR	TIPO	ASIGNATURA	DESCRIPTOR
1/1	T	FUNDAMENTOS PROGRAMACIÓN I (FP1)	INTRODUCCION A LA PROGRAMACION.DISEÑO DE ALGORITMOS

1/2	T	FUNDAMENTOS PROGRAMACIÓN II (FP2)	ANALISIS Y DISEÑO DE PROGRAMAS.LENGUAJES DE PROGRAMACION.DISEÑO DESCENDENTE.
2/A	T	PROGRAMACIÓN Y ESTRUCTURAS DE DATOS (PED)	ESTRCTURA DE DATOS I ALGORITMO DE MANIPULACION. TIPOS ABSTRACTOS DE DATOS. DISSEÑO RECURSIVO.
2/1	**O**	**PROGRAMACIÓN ORIENTADA A OBJETOS (POO)**	METODOLOGIA.CARACTERISTICAS DE LA POO.CLASES Y OBJETOS.DISEÑO ORIENTADO A OBJETOS.LENGUAJES DE PROGRAMACION ORIENTADO A OBJETOS.OBJETOS DISTRIBUIDOS.HERENCIA Y GENERIDAD.PERSISTENCIA EN UN ENTORNO ORIENTADO A OBJETOS.
2/1	O	HERRAMIENTAS DE PROGRAMACIÓN (HP)	ENTORNOS DE DESARROLLO. ESTANDARES DE NOMENCLATURA. INDENTADO Y COMENTARIO EN EL CODIGO FUENTE. ORGANIZACIÓN DEL CODIGO FUENTE. GESTIÓN DE PROYECTOS. PROGRAMACIÓN POR CONTRATO. INTERNALIZACIÓN DE APLICACIONES. CONTROL DE VERSIONES
2/1	T	LENGUAJES, GRAMÁTICAS Y AUTÓMATAS	MAQUINAS SECUENCIALES Y AUTOMATAS FINITOS. GRAMATICAS Y LENGUAJES FORMALES. REDES NEURONALES
2/2	O	COMPUTABILIDAD (C)	MAQUINAS DE TURING. FUNCIONES RECURSIVAS
2/2	O	LENGUAJES Y PARADIGMAS DE PROGRAMACIÓN (LPP)	PROGRAMACION PROCEDIMENTAL.PROGRAMACION FUNCIONAL.PROGRAMACION DECLARATIVA.PROGRAMACION ORIENTADA A OBJETOS.LENGUAJES DE SCRIPT
3/1	T	DISEÑO Y ANÁLISIS DE ALGORITMOS (DAA)	DISEÑO DE PROGRAMAS:DESCOMPOSICION MODULAR Y DOCUMENTACION.TECNICAS DE VERIFICACION Y PRUEBAS DE PROGRAMAS.LA EFICIENCIA DE LOS ALGORITMOS.DIVIDE Y VENCERAS.ALGORITMOSVORACES .ALGORITMOS CON RETROCESO.
3/2	O	DISEÑO Y PROGRAMACIÓN AVANZADA DE APLICACIONES (DPAA)	APLICACIONES DISTRIBUIDAS.APLICACIONES INTERNET.SISTEMAS ABIERTOS.OBJETOS DISTRIBUIDOS.CLIENTE/SERVIDOR
4/1	O	ALGORITMIA AVANZADA (AA)	BUSQUEDA EXHAUTIVA Y ESTOCASTICA.PROGRAMACION DINAMICA.ALGORITMOS DE CODIFICACION Y COMPRESION
4/A	T	PROCESADORES DE LENGUAJE (PL)	COMPILADORES. TRADUCTORES E INTERPRETES. FASES DE COMPILACION. OPTIMIZACION DE CODIGO. MACROPROCESADORES
-	OPT	PROGRAMACIÓN CONCURRENTE (PC)	PROCESOS.SINCRONIZACION,COMPETENCIA Y COOPERACION.EXCLUSION MUTUA.MEMORIA COMPARTIDA.MEMORIA DISTRIBUIDA.CSP
-	OPT	PROGRAMACIÓN EN ENTORNOS INTERACTIVOS (PEI)	PROGRAMACION VISUAL.PROGRAMACION DIRIGIDA A EVENTOS. INTERFACES GRAFICOS DE USUARIOS

-	OPT	PROGRAMACIÓN EN INTERNET (PI)	DESARROLLO Y PROGRAMACION DE SISTEMAS DE ACCESO A BASES DE DATOS DE INTERNET.PLANIFICACION,DISEÑO Y ADMINISTRACION DE SITIOS WEB.MIGRACION DE APLICACIONES A ENTORNOS EN INTERNET.HERRAMIENTAS DE DESARROLLO.DISEÑO Y PROGRAMACION DE ELEMENTOS MULTIMEDIA EN INTERNET

Todas estas asignaturas están claramente vinculadas, tal y como se refleja en el siguiente diagrama de actividad, donde los estados reflejan las asignaturas troncales/obligatorias y las transiciones reflejan los prerrequisitos existentes entre ellas.



La asignatura de POO supone, junto con PED y LPP (impartidas de manera simultánea), el primer contacto que el alumno tiene con el paradigma OO y con C++, aunque cuando cursa esta asignatura el alumno ya viene con nociones de programación estructurada y del lenguaje C, adquiridas en FP1 y FP2. Tras cursar POO, el alumno debe cursar otras asignaturas que requieren dominar la programación orientada a objetos para su correcta asimilación: DPAA, DAA y AA.

2. Objetivos

2.1 Objetivos generales

Además de los objetivos instrumentales generales cOI1, cOI2, cOI3, cOI4, cOI5 y cOI6, desarrollados en el capítulo 1 como objetivos comunes a todas las asignaturas, planteamos los siguientes objetivos adicionales:

) Objetivos Instrumentales Generales (saber y saber hacer)

- Dominar y utilizar la terminología usual de la asignatura no sólo en castellano/valenciano sino también en inglés, lengua franca de la Informática, tal y como reconoce el Libro Blanco de la titulación [EICE04].
- Ser capaz de discernir los tipos de aplicación y las situaciones en las que es posible y necesario aplicar el paradigma orientado a objetos.
- Comprender aquellos conceptos básicos y métodos relacionados con la Programación Orientada a Objetos que son recomendaciones o prerrequisitos para otras asignaturas.
- Comprender, interpretar y analizar el cambio de enfoque en el modo de resolver problemas que supone el uso del paradigma orientado a objetos respecto a otros paradigmas.
- Comprender la enorme importancia de crear software fiable, reutilizable y mantenible.
- Dominar estrategias básicas de reuso como son el uso de librerías y de patrones software.
- Desarrollar su capacidad de abstracción de cara a futuras tareas de análisis y diseño software.
- Experimentar con entornos y herramientas de desarrollo orientado a objetos de libre distribución.
- Ser capaz de comparar distintos lenguajes de programación orientados a objetos y apreciar sus ventajas e inconvenientes en base a su grado de cumplimiento de las principales características del paradigma orientado a objetos.

d) Objetivos Interpersonales Generales (ser y estar)

Estos objetivos se corresponden con los objetivos interpersonales generales comunes a todas las asignaturas de segundo curso: cOIP1, cOIP2 y cOIP3, detallados en el capítulo 1.

d) Objetivos Sistémicos Generales

Estos objetivos se corresponden a los objetivos sistémicos generales comunes a todas las asignaturas de segundo curso. Concretamente, estos objetivos son los objetivos cOS1, cOS2, cOS3 y cOS4, detallados en el capítulo 1.

2.2 Competencias académicas y profesionales

a) Competencias Instrumentales Específicas (saber y saber hacer)

- **Habilidades Cognitivas (saber)**

Además de las habilidades cCIC1 y cCIC2, desarrolladas en el capítulo 1, tendremos en cuenta las siguientes:

Bloque 1: Introducción al Paradigma Orientado a Objetos

- CIC1: Entender el concepto de paradigma y sus implicaciones en el modo de resolver problemas.
- CIC2: Conocer y entender el proceso de evolución de los distintos paradigmas de programación
- CIC3: Entender el tipo de problemas de desarrollo software que palía un uso correcto del paradigma orientado a objetos
- CIC4: Conocer el modo en que el paradigma orientado a objetos ayuda a mejorar las capacidades de reuso del software.

Bloque 2: Conceptos Básicos de la Programación Orientada a Objetos

- CIC5: Entender los conceptos de clase, atributo, operación, interfaz y objeto.
- CIC6: Entender el mecanismo de paso de mensajes
- CIC7: Comprender el modo en que se deben implementar los caminos de comunicación entre clases para permitir el paso de mensajes entre ellas.
- CIC8: Entender y ser capaz de implementar los distintos tipos de relaciones que se pueden establecer a nivel de objeto entre dos clases: asociaciones, agregaciones y composiciones.
- CIC9: Entender el concepto de estado de un objeto.
- CIC10: Entender la relación entre diagramas de clase UML y código

Bloque 3: Herencia y Polimorfismo

- CIC11: Entender el mecanismo de abstracción de la herencia
- CIC12: Saber discernir entre jerarquías de herencia seguras (bien definidas) e inseguras
- CIC13: Comprender los costes de la herencia
- CIC14: Saber decidir cuándo usar herencia y cuándo optar por composición
- CIC15: Entender el concepto de polimorfismo
- CIC16: Entender la diferencia entre ligadura estática y ligadura dinámica en los lenguajes de programación
- CIC17: Entender la relación a nivel de implementación entre herencia y polimorfismo
- CIC18: Saber identificar los distintos tipos de polimorfismo: sobrecarga, sobreescritura, variables polimórficas y genericidad.
- CIC19: Entender las relaciones entre los distintos tipos de polimorfismo

Bloque 4: Otras características del Paradigma Orientado a Objetos

- CIC20: Entender los mecanismos de gestión de errores que ofrecen algunos lenguajes de programación
- CIC21: Entender el concepto de persistencia
- CIC22: Entender el concepto de concurrencia

- **Capacidades Metodológicas (saber hacer)**

Además de las capacidades metodológicas generales cCIM1, cCIM2 y cCIM3, introducidas en el capítulo 1, para esta asignatura consideramos importante:

- CIM1: Ser capaz de interpretar un diagrama de clases UML para a partir de él proceder a la codificación de la aplicación
- CIM2: Ser capaz de aplicar las distintas técnicas de reuso de software (composición, herencia, polimorfismo), interpretando cuál es la más adecuada.

- **Destrezas Tecnológicas (saber hacer)**

Además de la destreza tecnológica cCIT1, introducida en el capítulo 1, se consideran las siguientes:

- CIT1: Manejar con fluidez las herramientas de programación de libre distribución gdb, doxygen y el compilador g++ de C++.

- **Destrezas Lingüísticas (saber hacer)**

  Además de las destrezas lingüísticas cCIL1 y cCIL2, introducidas en el capítulo 1, se consideran las siguientes dos destrezas lingüísticas:
  - CIL1: Adquirir y utilizar con fluidez un buen lenguaje de diseño software, tanto oral como escrito, siendo riguroso en las explicaciones de cualquier interacción o relación entre elementos del sistema.
  - CIL2: Comprender sinónimos y expresiones en otras lenguas utilizadas por otros autores para referirse a conceptos tratados en la asignatura.

b) Competencias Interpersonales (ser y estar)

Las competencias interpersonales se han dividido en competencias para tareas colaborativas y competencias relativas al compromiso con el trabajo. En ambos casos las competencias definidas para la asignatura coinciden con las aportadas el capítulo 1: cCIPTC1, cCIPTC2, CIPTR1, cIPTR2, cIPTR3 y cIPTR4.

c) Competencias Sistémicas

Las competencias sistémicas hacen referencia a la integración de capacidades cognitivas, destrezas prácticas y disposiciones recogidas en el capítulo 1. Dichas competencias se recogen con las etiquetas cCS1, cCS2, cCS3, cCS4 y cCS5. Además, consideramos competencias sistémicas de la asignatura:

- CS1: Capacidad de aplicación de los conocimientos, métodos y destrezas prácticas del paradigma orientado a objetos para el diseño y desarrollo de sistemas de información más robustos y mantenibles.
- CS2: Extrapolación los conocimientos adquiridos a otros lenguajes de programación
- CS3: Capacidad de prueba exhaustiva de los programas implementados

3. Prerrequisitos

3.1 Competencias y contenidos mínimos

- Entender el concepto de paradigma
- Entender el modo de abordar la resolución de problemas en el paradigma orientado a objetos
- Comprender la información contenida en un diagrama de clases UML y ser capaz de codificar dicha información en un lenguaje de programación orientado a objetos
- Entender el concepto de relación entre clases, y cómo a través de los relaciones se produce el paso de mensajes entre objetos del sistema
- Entender el concepto de generalización y cómo este se puede plasmar en jerarquías de herencia seguras
- Entender el concepto de polimorfismo y ser capaz de aplicarlo para mejorar la reusabilidad del código

3.2 Plan de trabajo y actividades para la consecución de los prerrequisitos

Los prerrequisitos necesarios para el estudio y entendimiento de esta asignatura se cubren en las asignaturas FP1 y FP2, que se imparten en el primer curso de las tres ingenierías de informática. Sin embargo, el alumnado de esta asignatura no es totalmente homogéneo, ya que con frecuencia nos llegan alumnos que, sin haber superado dichas asignaturas, se matriculan en Programación Orientada a Objetos. Para ellos, y para todo aquel alumno que desee comprobar/reforzar los conocimientos adquiridos sobre la construcción de algoritmos y el manejo de memoria en C++, en el Campus Virtual se actualiza todos los años una pequeña serie de libros y direcciones de Internet donde se recogen tutoriales de C que tratan los principales problemas con los que el alumno se puede encontrar a la hora de cursar nuestra signatura, principalmente el manejo de punteros. Además, en las horas de laboratorio se imparte un seminario de 3 sesiones (4.5 horas) donde, a nivel práctico, se refrescan conceptos básicos de programación: construcción de algoritmos, manejos de arrays, etc. Además, se intenta que los temas sean lo más autocontenidos posibles, para lo cual se recuerda al principio de cada tema aquellos conceptos que van a necesitar para entenderlo, y se refuerzan dichos conceptos al final. De esta forma aquellos estudiantes que hayan olvidado estos conceptos podrán recordarlos en clase y, si lo creen necesario, profundizar en ellos con la ayuda del material bibliográfico.

4. Bloques y temas de contenido

En este punto y, siguiendo los objetivos y competencias concretos de la programación orientada a objetos, enunciamos el contenido del programa de teoría y práctica de la asignatura de **Programación Orientada a Objetos.**

4.1 Bloques de contenidos de aprendizaje

Bloque I: Introducción al Paradigma Orientado a Objetos

Unidad 0: Presentación y objetivos

Unidad 1: Introducción al paradigma Orientado a Objetos.

Bloque II: Conceptos Básicos de la Programación Orientada a Objetos

Unidad 2: Fundamentos de la POO

Bloque III: Herencia y Polimorfismo

Unidad 3: Herencia

Unidad 4: Polimorfismo

Bloque IV: Otras características del Paradigma Orientado a Objetos

Unidad 5: Características Avanzadas del Paradigma Orientado a Objetos

4.2 Temas o unidades de contenido. Desarrollo

Bloque I: Introducción al Paradigma Orientado a Objetos

Unidad 0: Presentación y objetivos

Unidad 1: Introducción al paradigma Orientado a Objetos

- El progreso de la abstracción.
- Principales Paradigmas de Programación.
- El Paradigma Orientado a Objetos
- Historia de los Lenguajes Orientados a Objetos
- Metas del Paradigma Orientado a Objetos
- Conclusiones
- Ejercicios Propuestos.

Bloque II: Conceptos Básicos de la Programación Orientada a Objetos

Unidad 2: Fundamentos de la POO

- Clase.
- Atributos.
- Operaciones.
- Interfaces
- Objetos
- Metaclases

- Diseño de Aplicaciones OO
- Relaciones entre clases y objetos
- Conclusiones
- Ejercicios Resueltos
- Ejercicios Propuestos

Bloque III: Herencia y Polimorfismo

Unidad 3: Herencia

- Introducción a la Herencia

- Herencia Simple
- Herencia Múltiple
- Herencia de Interfaz
- Herencia de Implementación
- Beneficios y costes de la herencia
- Elección de la técnica de reuso
- Conclusiones
- Ejercicios propuestos

Unidad 4: Polimorfismo

- Polimorfismo y reuso.
- Sobrecarga
- Sobreescritura
- Variables Polimórficas
- Genericidad
- Caso de estudio
- Conclusiones
- Ejercicios Propuestos

Bloque IV: Otras características del Paradigma Orientado a Objetos

Unidad 5: Características Avanzadas del Paradigma Orientado a Objetos

- Gestión de Errores.
- Persistencia
- Concurrencia.
- Conclusiones
- Ejercicios Propuestos.

5. Metodología y estrategias de aprendizaje

5.1 Metodología docente

La metodología docente se ha desarrollado en el capítulo 1 desde el punto de vista generall para el segundo curso de la titulación de Informática. Dicho desarrollo, aunque genérico, se considera válido para el caso particular de la asignatura Programación Orientada a Objetos. Lo que cabe destacar serían las prácticas de laboratorio y las actividades en grupos pequeños, puesto que jugarán un papel fundamental. Las actividades que se proponen son:

Clases de teoría con apoyo de material audiovisual: En lo que se refiere a las clases de teoría, cabe mencionar que éstas se apoyan de material audiovisual disponible para el alumnado y que le puede servir de guía sobre los contenidos más importantes de la asignatura. Además, los profesores de la asignatura hemos creado unos apuntes y estamos editando un libro que incluye todos los contenidos de la misma. De esta forma, el alumnado puede entender y asimilar mejor lo que se está explicando. Dichas explicaciones teóricas se intercalarán con la realización de problemas, ejemplos prácticos y aplicaciones siempre que el contenido lo requiera.

Actividades en grupos pequeños / tutorías docentes: Estas actividades estarán relacionadas con la realización de problemas y cuestiones teórico-prácticas relacionadas con la asignatura, de manera que se intente reforzar y aplicar los conceptos básicos a situaciones reales concretas y fomentar la capacidad de análisis, síntesis y autoevaluación del alumnado.

Prácticas de laboratorio: En cuanto a las prácticas de laboratorio, cabe mencionar que éstas se preparan durante los meses previos al inicio de las prácticas, y son implementadas y testeadas por varios profesores antes del comienzo del curso. Las primeras sesiones se dedicarán a un seminario del lenguaje de programación utilizado en las mismas (C++ en nuestro caso) para permitir al alumno la toma de contacto con dicho lenguaje. El alumno deberá utilizar el material audiovisual y los apuntes de la asignatura como apoyo al desarrollo de dichas prácticas.

Trabajos complementarios: En cuanto a los trabajos complementarios, comentar que dichos trabajos incidirán en la nota final de la asignatura y pueden ser de índole teórica, de índole práctica o de índole teórico-práctica y deberán realizarse de forma individual o por parejas.

Tutorías de atención al alumnado: El alumnado tiene a su disposición unas horas de tutorías en las cuales puede consultar cualquier duda relacionada con la organización y planificación de la asignatura, así como dudas concretas sobre el contenido de la asignatura. Además de dichas tutorías individualizadas, se programarán varias tutorías en grupo, al menos una para cada bloque de la asignatura correspondiente.

5.2 Estrategias de aprendizaje

Las estrategias de aprendizaje se han establecido a nivel común en el capítulo 1. Tal y como se indicó, los medios tradicionales como las trasparencias, apuntes o presentaciones por ordenador, no son los únicos medios sobre los que nos apoyaremos en nuestra docencia. Concretamente, las páginas web y el campus virtual ofrecen innumerables posibilidades que no hay que dejar pasar. Así, se ha elaborado en el campus virtual una página de la asignatura, que incluye toda la información que el alumno necesita. El uso de la misma ha sido mayoritario en las experiencias llevadas a cabo hasta el momento.

Entre otras cosas, en dicha página podemos encontrar:

- Tablón de anuncios: Desde aquí, el alumno puede estar perfectamente informado de cualquier novedad relacionada con la asignatura. Además de recordar los plazos de entrega de cualquier trabajo, fechas de exámenes, etc.
- Tutorías: Indica el horario de atención al alumnado.

- Objetivos: Resume los objetivos que se pretenden alcanzar en esta asignatura.
- Temario: Especifica el temario de esta asignatura.
- Material docente de la asignatura: Aquí se puede encontrar todo el material que como mínimo va a ser necesario para el seguimiento de las clases teóricas y prácticas.
- Bibliografía complementaria: Aparece un listado de bibliografía complementaria que pueden consultar para profundizar en la asignatura o preparar los trabajos complementarios. Toda la bibliografía se encuentra disponible en la Biblioteca de la Universidad de Alicante.
- Plan de prácticas: Se especifica semana a semana qué trabajo debe realizar el alumnado tanto en las horas presenciales como en las no presenciales de forma autónoma.
- Horarios de teoría y prácticas: Contiene información sobre los grupos de teoría y prácticas, así como el profesorado que los imparte.
- Exámenes de autoevaluación: actualmente sólo desarrollados para el seminario de C++ impartido durante las primeras tres sesiones prácticas de la asignatura, con el fin de que el alumno testee si posee el nivel de conocimientos mínimo que le permita afrontar la elaboración de la primera práctica de la asignatura con ciertas garantías de éxito.
- Enlaces de interés: Aquí aparecen una serie de enlaces interesantes que pueden servir para profundizar en algunos contenidos de la materia.

En base a todo esto, la estrategia de aprendizaje que se propone se compone de las siguientes fases:

1. Recopilación de toda la documentación de la asignatura.
2. Planificación de las clases teóricas:
   - Lectura previa del guión correspondiente a la sesión de teoría que se trate.
   - Una vez realizada la clase de teoría, se debe estudiar de forma autónoma su contenido y en caso de no entender algo intentar primero contrastarlo con otros compañeros o utilizando la bibliografía recomendada. Si esto no es suficiente se acudirá a tutorías para intentar solucionar el problema.
5. Planificación de las actividades en grupos pequeños:
   - Una vez entendidas las explicaciones de las clases teóricas se leerá, de forma independiente, la actividad a realizar en grupos pequeños para, al inicio de la actividad, poder preguntar las dudas surgidas en el entendimiento del enunciado.
   - En las actividades en grupos pequeños, cada subgrupo tendrá que hacer la actividad propuesta que será corregida en la propia aula entre todos o por el profesor fuera del aula.
   - Una vez corregida la actividad propuesta, los grupos deben analizar cuáles han sido los errores cometidos para intentar no volverlos a realizar. Si es necesario se pedirá ayuda al profesor correspondiente.
9. Planificación de las clases prácticas:
   - Una vez entendidas las explicaciones de las clases teóricas se leerá, de forma independiente, la práctica de laboratorio que se debe realizar en la sesión correspondiente para, al inicio de la sesión, poder preguntar las dudas surgidas en el entendimiento del enunciado.
   - Parte de las prácticas se realizarán en los laboratorios y parte en horas no presenciales. Se deberá cumplir el calendario de entrega de prácticas. El profesorado corregirá en un plazo máximo de dos semanas dichas prácticas, indicando una vez corregidas los fallos más comunes y publicando los programas utilizados en la corrección. Cada estudiante de forma individual debe analizar cuáles han sido los errores cometidos y solucionarlos para la entrega siguiente. Si es necesario se pedirá ayuda al profesor correspondiente.
7. Autoevaluación: Una vez realizadas todas las actividades previas relacionadas con un tema concreto, el estudiante debe discernir si cree que dicho tema ha sido totalmente entendido. En algunos temas se podrá ayudar de los tests de autoevaluación que hayan sido publicados por los profesores en el campus virtual. En caso de no haber asimilado los contenidos mínimos, el alumno debe incidir en el estudio de los contenidos que crea tener más flojos, utilizando si lo cree conveniente las tutorías y realizando algunos problemas de ampliación, bien de los propuestos en las hojas de problemas o bien haciendo uso de la bibliografía.

12. Evaluación final: Si el resultado de las prácticas propuestas ha sido satisfactorio, el estudiante deberá realizar el examen de teoría, y una vez aprobado, habrá superado la asignatura.

De forma opcional se podrá hacer trabajos complementarios de forma individual o por parejas, para subir la nota siempre y cuando el trabajo realizado a lo largo del curso se considere satisfactorio.

6. Plan de trabajo de los alumnos: especificación del tiempo y esfuerzo del aprendizaje

En las siguientes tablas se esquematiza cuál va a ser el plan de trabajo de esta asignatura. Se distingue entre horas presenciales dedicadas a la realización de actividades en las aulas, donde ell profesorado juega un papel primordial, y horas no presenciales dedicadas al trabajo y esfuerzo personal realizado en la asignatura, de forma autónoma, por los estudiantes.

6.1 Planificación del programa de Teoría

	Horas Presenciales			Horas No Presenciales	TOTAL
ACTIVIDAD	**Trabajo en aula**	**Tutorías organizadas**	**Actividades complementa-rias**	**Aprendizaje autónomo y colaborativo**	
BLOQUE 1:					
Unidad 0	1.5	0		0	1.5
Unidad 1	3	1		3	7
BLOQUE 2:					
Unidad 2	4.5	2		6	12.5
BLOQUE 3:					
Unidad 3	4.5	2	1	6	13.5
Unidad 4	6	2	1	10	19
BLOQUE 4:					
Unidad 5	1.5	3		2	6.5
Preparación del examen final:	1.5			15	16.5
Examen final:	2			0	2
TOTAL:	**24.5**	**10**	**2**	**42**	**78.5**

6.2 Planificación del programa de Prácticas

	Horas Presenciales		Horas No Presenciales	TOTAL
ACTIVIDAD	**Prácticas en aula**	**Tutorías organizadas**	**Realización de las prácticas fuera del horario de la asignatura**	
Unidad 0: Seminario C++	4.5		0	4.5
Práctica 1	4.5		5	9.5
Práctica 2	6	0.5	8	14.5
Práctica 3	6	1	8	15

TOTAL:	**21**	**1.5**	**21**	**43.5**

6.3 Planificación (resumen)

Tras realizar la planificación de la asignatura obtenemos los siguientes resultados:

- Número total de horas presenciales: 59
- Número total de horas no presenciales: 63
- Número total de horas: 122

7. Bibliografía y materiales recomendados

7.1 Bibliografía básica

- TEORÍA
  - **An Introduction to Object-oriented Programming** **Timothy Budd**, 2001. Ed. Addsion Wesley
  - Thinking in C++. 2nd Edition. **Bruce Eckel**. Libro electrónico gratuito. http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html
- PRÁCTICAS
  - **C++: How to Program**. 5ª Edición. **Harvey M. Deitel, Paul J. Deitel.** 2005. Ed. US Imports & PHIPEs
  - Aprenda C++ como si estuviera en primero
  - www.cplusplus.com

7.2 Bibliografía complementaria

- TEORÍA
  - **Fundamentals of Object Oriented Design in UML**. **Mailir Page-Jones**, 2000. Ed. Addison Wesley
  - **Requirements Analysis and System Design**. Developing Information Systems with UML. **L.A. Maciaszek**, 2001. Ed. Addison-Wesley
  - **Object-oriented Software Construction (Prentice Hall International Series in Computer Science)** **Bertrand Meyer**, 2000. Ed. Prentice Hall
  - **The Object Primer 3rd Edition Agile Model Driven Development with UML 2**. **Scott Ambler**. 2004. Ed. Cambridge University Press, ISBN#: 0-521-54018-6
- PRÁCTICAS
  - ***Simply C++ An Application-Driven Tutorial Approach. 2005. Deitel & Deitel.***
  - **Programación en C++: algoritmos, estructuras de datos y objetos**. **L. Joyanes Aguilar**, 2000. Ed. Mc Graw Hill
  - **The C++ Primer**. 3rd. Edition. **S.B. Lippman**, 1999. Ed. Addison Wesley.

7.3 Otros recursos

PÁGINAS WEB DE INTERÉS

- http://se.inf.ethz.ch/touch/TOUCH.pdf.
- http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html
- http://www.hibernate.org/

8. Evaluación de los procesos y resultados de aprendizaje

8.1 Procedimiento de evaluación

Según [MS04] el proceso evaluativo se ubica, al menos, en cuatro espacios de aprendizaje:

- Los contenidos: cuestiones y redes conceptuales
- Las prácticas: trabajos realizados dentro y fuera del aula durante el desarrollo de los créditos teóricos y prácticos
- La participación en el trabajo de aula, aprendizaje virtual y tutorías
- El proyecto o memoria final, porfolio, etc.

Para la evaluación del alumno en Programación Orientada a Objetos se sigue un tipo criterial, i.e. una evaluación que pretende determinar las competencias que posee el estudiante tras el desarrollo del proceso de aprendizaje. En este tipo de evaluación se recoge información mediante un instrumento, prueba, procedimiento o actividad para poder describir e identificar las competencias adquiridas por los estudiantes acerca de un dominio de referencia, descrito en la planificación docente inicial. Así, los criterios de evaluación deben correlacionarse con los objetivos/competencias y especificar principalmente un dominio conceptual claro, la adquisición de procedimientos, técnicas, instrumentos, habilidades y destrezas de ejecución profesional y académica, que concretan el nivel de capacidades personales y profesionales necesarias para el ejercicio profesional.

8.2 Criterios de evaluación

Teniendo en cuenta estas premisas, los criterios de evaluación aplicados a Programación Orientada a Objetos son:

- Teoría: examen final. 40% de la asignatura
- Práctica: Tres entregas. 60% de la asignatura
    - Primera entrega: 20%
    - Segunda entrega 50%
    - Tercera entrega: 30%
    - **Se deben aprobar al menos dos de las tres prácticas para que se haga media**

- Trabajos optativos: HASTA 2 PUNTOS (valorable por el profesor en función tanto de la defensa que haga el alumno como de la síntesis, practicidad y relevancia para la asignatura del trabajo)
    - Obligatoria la defensa del trabajo en clase: presentación de 10-15 minutos
    - Ejemplos de trabajos:
        - Uso de STL en C++
        - Ejemplo de implementación de persistencia en C++
        - Conexión de C++ con Bases de Datos
        - eXtreme Programming
        - Implementación de la práctica con NCurses

Se debe aprobar la práctica para presentarse a teoría. La nota de teoría y práctica NO COMPENSAN (se deben aprobar por separado). La realización de ejercicios propuestos en clase es útil para decidir nota en caso de duda.

La calificación se hará de acuerdo a las siguientes pautas:

Sobresaliente:

- El conocimiento sobre la asignatura es profundo y se extiende más allá del trabajo cubierto por el programa.
- La comprensión conceptual es sobresaliente.
- Los problemas relacionados con la asignatura son resueltos con eficiencia y precisión; los procedimientos de resolución de problemas se ajustan a la naturaleza del problema.
- Las destrezas experimentales son ejemplares y muestran un completo análisis y evaluación de los resultados.
- La actuación en las destrezas transferibles es generalmente muy buena.
- La participación en las clases y distintas actividades ha sido muy correcta y muy satisfactoria.

Notable:

- El conocimiento sobre POO cubre de manera satisfactoria el programa.
- La comprensión conceptual es notable. Los problemas relacionados con la asignatura son resueltos con eficiencia y precisión; los procedimientos de resolución de problemas son generalmente ajustados a la naturaleza del problema.
- Las destrezas experimentales son generalmente buenas y muestran un análisis y evaluación de los resultados aceptables.
- La actuación en las destrezas transferibles es generalmente buena.
- La participación en las clases y distintas actividades ha sido correcta y bastante satisfactoria.

Aprobado:

- El conocimiento y la comprensión del contenido cubierto en el curso es básico.
- Los problemas relacionados con la asignatura son generalmente resueltos de forma adecuada.
- Las prácticas de laboratorio estándares son usualmente desarrolladas con éxito razonable aunque el significado y análisis de los resultados pueden no ser entendidos completamente.
- Las destrezas transferibles están a un nivel básico.
- La participación en las clases y distintas actividades ha sido correcta pero no siempre satisfactoria.

Suspenso:

- El conocimiento y la comprensión del contenido cubierto en el curso no ha sido aceptable.
- Los problemas relacionados con la asignatura no son, generalmente, resueltos de forma adecuada.
- Las prácticas de laboratorio estándares no son usualmente desarrolladas satisfactoriamente y el significado y análisis de los resultados no son entendidos generalmente.
- Las destrezas transferibles están a un nivel deficiente.
- La participación en las clases y distintas actividades ha sido escasa y deficiente.

Queremos hacer notar que para la obtención de matrícula de honor es necesario obtener un sobresaliente alto y hacer un trabajo complementario de calidad.

9. Análisis de coherencia de la guía docente

En la siguiente tabla presentamos el análisis de coherencia de la guía docente de Programación orientada a Objetos. En dicha tabla se han relacionado los objetivos y competencias con los bloques de contenido, el plan de trabajo propuesto para el alumnado y el sistema y criterio de evaluación.

OBJETIVOS GENERALES	COMPETENCIAS ESPECÍFICAS	BLOQUES DE CONTENIDOS				PLAN DE TRABAJO DE LOS ALUMNOS	PROCEDIMIENTOS Y CRITERIOS DE EVALUACIÓN
	Instrumentales (saber)	Bloque 1 (temas)	Bloque 2 (temas)	Bloque 3 (temas)	Bloque 4 (temas)		
OI1	De CIC1 a CIC110 cCIC1 cCIC2	1	2			Enseñanza presencial (Lección magistral /Trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (Aprendizaje on-line/biblioteca/realización de ejercicios y prácticas propuestas). Tutorías individualizadas y organizadas.	Procedimientos: Prácticas. Actividades en grupo. Criterios: Grado de comprensión, interpretación, análisis y aplicación de los conceptos relativos a la programación orientada a objetos.
OI2	De CIC11 a CIC22 cCIC1 cCIC2	1		3,4	5	Enseñanza presencial (Lección magistral /trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (Aprendizaje on-line/biblioteca/realización de ejercicios y prácticas propuestos). Tutorías individualizadas y organizadas	Procedimientos: Prácticas. Actividades en grupo. Criterios: Grado de comprensión, interpretación, análisis y aplicación de los conceptos relativos a la programación orientada a objetos

OI3	De CIC1 a CIC22 cCIC1 cCIC2	1	2	3,4	5	Enseñanza presencial (Lección magistral). Enseñanza no presencial (realización de ejercicios y prácticas propuestos). Tutorías individualizadas.	Procedimientos: Prácticas. Criterios: Grado de comprensión conceptual.
OI4	De CIC1 a CIC22 cCIC1 cCIC2	1	2	3,4	5	Enseñanza presencial (Lección magistral). Enseñanza no presencial (realización de ejercicios y prácticas propuestos). Tutorías individualizadas.	Procedimientos: Prácticas. Criterios: Grado de comprensión conceptual.
OI6	De CIC1 a CIC22 cCIC1 cCIC2	1	2	3,4	5	Enseñanza presencial (Lección magistral). Enseñanza no presencial (realización de ejercicios y prácticas propuestos). Tutorías individualizadas.	Procedimientos: Prácticas. Criterios: Grado de comprensión conceptual.
cOI6	De CIC1 a CIC21	1	2	3,4	5	Enseñanza presencial (Lección magistral /trabajo de aula en grupos). Enseñanza no presencial (Aprendizaje on-line/biblioteca/realización de ejercicios y prácticas propuestos). Tutorías individualizadas y organizadas.	Procedimientos: Prácticas. Actividades en grupo. Trabajo complementario. Criterios: Grado de comprensión, interpretación, análisis y aplicación de los conceptos.

OBJETIVOS GENERALES	COMPETENCIAS ESPECÍFICAS	BLOQUES DE CONTENIDOS				PLAN DE TRABAJO DE LOS ALUMNOS	PROCEDIMIENTOS Y CRITERIOS DE EVALUACIÓN
	Instrumentales (saber hacer)	Bloque 1 (temas)	Bloque 2 (temas)	Bloque 3 (temas)	Bloque 4 (temas)		
OI5	CIM1 CIM2 cCIM1 cCIM2 cCIM3		2	3,4	5	Enseñanza presencial (Lección magistral). Enseñanza no presencial (Aprendizaje on-line/biblioteca).	Procedimientos: Prácticas. Actividades en grupo. Trabajo complementario. Criterios: Nivel de actuación en las destrezas transferibles.
cOI1	CIM1 CIM2 cCIM1 cCIM2 cCIM3	1	2	3, 4	5	Enseñanza presencial (Trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (Aprendizaje on-line/realización de ejercicios y prácticas propuestos). Tutorías individualizadas y organizadas.	Procedimientos: Prácticas. Actividades en grupo. Criterios: Grado de destreza experimental., eficiencia y precisión en la resolución de problemas usando programación orientada a objetos
cOI2	CIT1 cCIT1	1	2	3,4	5	Enseñanza presencial (prácticas de laboratorio). Enseñanza no presencial (Aprendizaje on-line/ realización de prácticas propuestos). Tutorías individualizadas y organizadas	Procedimientos: Prácticas. Actividades en grupo. Criterios: Grado de destreza experimental.

cOI3	CIL1 cCIL1	1	2	3, 4	5	Enseñanza presencial (Lección magistral /trabajo de aula en grupos). Enseñanza no presencial (Aprendizaje on-line/biblioteca/realización de ejercicios y prácticas propuestos). Tutorías individualizadas y organizadas. Memorias de los trabajos realizados.	<u>Procedimientos</u>: Examen. Prácticas. Actividades en grupo. <u>Criterios:</u> Grado de rigurosidad en las explicaciones de los procedimientos aplicados.
cOI4	cCIL2	1	2	3, 4	5	Enseñanza presencial (Lección magistral). Enseñanza no presencial (Aprendizaje on-line/biblioteca).	<u>Procedimientos</u>: Trabajo en grupos. Trabajo complementario. <u>Criterios:</u> Grado de conocimiento de los distintos términos de programación orientada a objetos, en castellano y/o valenciano y en inglés.
cOI5	De CIM1 a CIM2 cCIM3	1	2	3, 4	5	Enseñanza no presencial (Aprendizaje on-line/biblioteca).	<u>Procedimientos</u>: Trabajo en grupos. Trabajo complementario. <u>Criterios:</u> Grado de comprensión, interpretación, análisis y aplicación de los conceptos.

OBJETIVOS GENERALES	COMPETENCIAS ESPECÍFICAS	BLOQUES DE CONTENIDOS				PLAN DE TRABAJO DE LOS ALUMNOS	PROCEDIMIENTOS Y CRITERIOS DE EVALUACIÓN
	Interpersonales (ser y estar)	Bloque 1 (temas)	Bloque 2 (temas)	Bloque 3 (temas)	Bloque 4 (temas)		
cOIP1	CIPTC1 cCIPTC1 cCIPTC2		2	3,4	5	Enseñanza presencial (Trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (realización de ejercicios y prácticas propuestos). Tutorías organizadas. Memorias de los trabajos realizados.	<u>Procedimientos</u>: Actividades en grupo. .<u>Criterios:</u> Grado de destreza en trabajos participativos.
cOIP2	cCIPTR1 cCIPTR2		2	3, 4	5	Enseñanza presencial (Trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (realización de ejercicios y prácticas propuestos). Tutorías organizadas. Memorias de los trabajos realizados.	<u>Procedimientos</u>: Actividades en grupo. <u>Criterios:</u> Grado de destreza en trabajos participativos.
cOIP3	cCIPTR2		2	3,4	5	Enseñanza presencial (Trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (realización de ejercicios y prácticas propuestos). Tutorías organizadas. Memorias de los trabajos realizados.	<u>Procedimientos</u>: Actividades en grupo. <u>Criterios:</u> Grado de destreza en trabajos participativos.

OBJETIVOS GENERALES	COMPETENCIAS ESPECÍFICAS	BLOQUES DE CONTENIDOS				PLAN DE TRABAJO DE LOS ALUMNOS	PROCEDIMIENTOS Y CRITERIOS DE EVALUACIÓN
	Sistémicas	Bloque 1 (temas)	Bloque 2 (temas)	Bloque 3 (temas)	Bloque 4 (temas)		
cOS1	cCS1 cCS4 cCS5	1	2	3,4	5	Enseñanza presencial (Lección magistral /trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (Aprendizaje on-line/biblioteca/realización de ejercicios y prácticas propuestas). Tutorías individualizadas y organizadas.	<u>Procedimientos</u>: Prácticas. Actividades en grupo. Trabajo complementario. <u>Criterios:</u> Nivel de actuación en las destrezas transferibles.
cOS2	cCS2 cCS3	1	2	3, 4	5	Enseñanza presencial (Lección magistral /trabajo de aula en grupos). Enseñanza no presencial (Aprendizaje on-line/ realización de ejercicios y prácticas propuestas). Tutorías individualizadas y organizadas	<u>Procedimientos</u>: Actividades en grupo. Trabajo complementario. <u>Criterios:</u> Grado de análisis y evaluación de los problemas resueltos con programación orientada a objetos

cOS3	cCS1 cCS2 cCS4 cCS5	1	2	3,4	5	Enseñanza presencial (Lección magistral /trabajo de aula en grupos/ prácticas de laboratorio). Enseñanza no presencial (Aprendizaje on-line/biblioteca/realización de ejercicios y prácticas propuestas). Tutorías individualizadas y organizadas.	<u>Procedimientos</u>: Prácticas. Trabajo complementario. <u>Criterios:</u> Nivel de actuación en las destrezas transferibles.
cOS4	cCS2		2	3, 4	5	Enseñanza presencial (Lección magistral /trabajo de aula en grupos). Enseñanza no presencial (Aprendizaje on-line/ realización de ejercicios y prácticas propuestas). Tutorías individualizadas y organizadas	<u>Procedimientos</u>: Prácticas. Actividades en grupo. Trabajo complementario. <u>Criterios:</u> Grado de análisis y evaluación de los problemas resueltos con programación orientada a objetos

Borrador de Anexos para la memoria final de la red.

[ACM2001] Computing Curricula 2001 Project. http://www.acm.org/education/curricula.html

[ACM2004] Computing Curricula 2004 Project. http://www.acm.org/education/curricula.html

[CC2001] Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Volumen de Computing Curricula Series. http://www.acm.org/education/curricula.html

[CE2004] Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering. Volumen de Computing Curricula Series. http://www.acm.org/education/curricula.html

[EICE04] Libro Blanco EICE v1.1.4.

[IS2002] Curriculum Guidelines for Undergraduate Degree Programs in Information Systems**.** Volumen de Computing Curricula Series. http://www.acm.org/education/curricula.html

[IT2005] Draft of Information Technology 2005. http://www.acm.org/education/curricula.html

[MEYER97] Object-Oriented Software Construction, Second Edition. Prentice Hall Professional Technical Reference. 1997. ISBN 0-13-629155-4

[MS04] Redes para Investigar el Currículo. Diseño del Aprendizaje en el EEES. *M.A. Martínez & N. Sauleda Parés.* 2004

[SE2004] Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Volumen de Computing Curricula Series. http://www.acm.org/education/curricula.html

Anexo A: Clasificación de las asignaturas del Plan de estudios vigente en la Universidad de Alicante según subcategorías del Libro Blanco (Contenidos Específicos de la Ingeniería)

P: Programación
IS: Ingeniería del Software, Sistemas de Información y Sistemas Inteligentes
SO: Sistemas operativos, Sistemas Distribuidos y Redes
IC: Ingeniería de Computadores

Curso: 1

Tipo: TRONCAL

Asignatura	Créditos	Área
ÁLGEBRA (9168)	6	-
CÁLCULO INFINITESIMAL (9169)	9	-
ESTADÍSTICA (9162)	6	-
FUNDAMENTOS DE PROGRAMACIÓN I (9171)	6	P
FUNDAMENTOS DE PROGRAMACIÓN II (9172)	6	P
FUNDAMENTOS FÍSICOS DE LA INFORMÁTICA (9167)	10,5	-
INFORMÁTICA BÁSICA (9165)	12	IC
MATEMÁTICA DISCRETA (9170)	6	-

Tipo: OBLIGATORIA

Asignatura	Créditos	Área
LÓGICA COMPUTACIONAL (9188)	6	-

Curso: 2

Tipo: TRONCAL

Asignatura	Créditos	Área
BASES DE DATOS I (9164)	9	IS
COMPUTABILIDAD (9177)	4,5	P
ESTRUCTURAS DE COMPUTADORES (9166)	6	IC
LENGUAJES,GRAMÁTICAS Y AUTÓMATAS (9176)	4,5	P
PROGRAMACIÓN Y ESTRUCTURAS DE DATOS (9163)	9	P
SISTEMAS OPERATIVOS I (9174)	4,5	SO

Tipo: OBLIGATORIA

Asignatura	Créditos	Área
HERRAMIENTAS DE PROGRAMACIÓN (9452)	6	P
LENGUAJES Y PARADIGMAS DE PROGRAMACIÓN (9189)	6	P
PROGRAMACIÓN ORIENTADA A OBJETOS (9190)	**4,5**	**P**

Curso: 3

Tipo: TRONCAL

Asignatura	Créditos	Área
DISEÑO Y ANÁLISIS DE ALGORITMOS (9173)	6	P
SISTEMAS OPERATIVOS II (9175)	4,5	SO

Tipo: OBLIGATORIA

Asignatura	Créditos	Área
BASES DE DATOS II (9192)	6	IS
DISEÑO Y PROGRAMACIÓN AVANZADA DE APLICACIONES (9193)	4,5	IS
FUNDAMENTOS DE ARQUITECTURAS DE COMPUTADORES (9194)	6	IC
GRÁFICOS POR COMPUTADOR (9191)	4,5	P
SISTEMAS DE INFORMACIÓN EN LA EMPRESA I (9195)	6	-
SISTEMAS DE INFORMACIÓN EN LA EMPRESA II (9196)	6	-

Curso: 4

- **Tipo: TRONCAL**

Asignatura	Créditos	Área
ANÁLISIS Y ESPECIFICACIÓN DE SISTEMAS DE INFORMACIÓN (9179)	6	IS
ARQUITECTURA E INGENIERÍA DE COMPUTADORES (9178)	12	IC
FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL (9182)	4,5	IS
INGENIERÍA DEL SOFTWARE I (9180)	6	IS
PROCESADORES DE LENGUAJE (9184)	9	P
REDES (9185)	7,5	SO
SISTEMAS DE TRANSPORTE DE DATOS (9186)	6	SO
TÉCNICAS DE INTELIGENCIA ARTIFICIAL (9183)	4,5	IS

- **Tipo: OBLIGATORIA**

Asignatura	Créditos	Área
ALGORITMIA AVANZADA (9197)	4,5	P

Curso: 5

- **Tipo: TRONCAL**

Asignatura	Créditos	Área
INGENIERÍA DEL SOFTWARE II (9181)	6	IS
SISTEMAS INFORMÁTICOS (9187)	15	-

- **Tipo: OBLIGATORIA**

Asignatura	Créditos	Área
SISTEMAS OPERATIVOS EN RED (9198)	9	SO

Anexo B: Estimación de créditos ECTS para POO

Tal y como podemos ver en la Tabla 1, en el plan de estudios actual, conducente a la obtención del título de Ingeniero en Informática por la Universidad de Alicante, la carga docente acumulada asignable a los Contenidos Formativos Comunes suma un total de 255 créditos (de los 364,5 que componen la titulación), divididos de la siguiente forma:

DIVISIÓN CRÉDITOS ACTUALES SEGÚN CATEGORÍAS DE LOS CFC	**Créditos**	**%T/O**
FUNDAMENTOS CIENTÍFICOS	43,5	17,06%
CONTENIDOS GENERALES ING	12	4,71%
PROYECTO	15	5,88%
CONTENIDOS ESPECÍFICOS ING	**184,5**	**72,35%**
TOTAL	255	*100%*

Dentro de los Contenidos Específicos, tenemos el siguiente desglose en las cuatro grandes subcategorías:

Programación	***70,5***	***38,21%***
Ingeniería Sw, SI y Stmas Inteligentes	46,5	25,20%
SO, SD y Redes	*31,5*	*17,07%*
Ingeniería Computadores	*36*	*19,51%*

Si suponemos que los porcentajes de la tabla anterior se van a mantener en la nueva distribución de créditos ECTS por asignatura, y teniendo en cuenta el cálculo de 140,4 ECTS (39% del total de 240 créditos) que es el máximo recomendado por el libro blanco (tomando mínimos para el resto de categorías) para el conjunto de contenidos específicos, obtenemos los siguientes créditos para cada subcategoría:

CONTENIDOS ESPECÍFICOS ING	**39,00%**	**140,4**
Programación	***38,21%***	***53,65***
Ingeniería Sw, SI y Stmas Inteligentes	*25,20%*	*35,39*
SO, SD y Redes	*17,07%*	*23,97*
Ingeniería Computadores	*19,51%*	*27,40*

Si ahora adaptamos los créditos de las asignaturas incluidas en la subcategoría de Programación a esos 53,65 ECTS, mediante una sencilla regla de tres nos quedaría:

ASIGNATURA	Cred Act	ECTS
FUNDAMENTOS PROGRAMACIÓN I	6	4,77
FUNDAMENTOS PROGRAMACIÓN II	6	4,77
HERRAMIENTAS DE PROGRAMACIÓN	4,5	3,58
DISEÑO Y ANÁLISIS DE ALGORITMOS	6	4,77
ALGORITMIA AVANZADA	4,5	3,58
COMPUTABILIDAD	4,5	3,58
LENGUAJES, GRAMÁTICAS Y AUTÓMATAS	4,5	3,58
PROCESADORES DEL LENGUAJE	9	7,15
LENGUAJES Y PARADIGMAS DE PROGRAMACIÓN	4,5	3,58
PROGRAMACIÓN ORIENTADA A OBJETOS	4,5	3,58

DISEÑO Y PROGRAMACIÓN AVANZADA DE APLICACIONES	4,5	3,58
PROGRAMACIÓN Y ESTRUCTURAS DE DATOS	9	7,15
TOTAL	67,5	53,65

Este cálculo proporciona una asignación de 3,58 créditos ETCS a la asignatura de Programación Orientada a Objetos (unas 108 horas), lo que es ligeramente inferior a las 122 horas calculadas en la sección 6. Aún más, el plan de estudios actual se basa en un decreto que no está actualizado, y por tanto no recoge la importancia que los curricula más actuales reconocen al paradigma Orientado a Objetos. Es por ello que consideramos necesario, con el fin de reconocer el esfuerzo que el alumno debe realizar para la asimilación de esta materia, y con el fin de impartirla con la profundidad y rigor que los Curricula Internacionales sugieren, aumentar significativamente la carga docente de esta asignatura en los nuevos planes de estudio, y asignarle un mínimo de 6 créditos ECTS (es decir, entre 150 y 180 horas de trabajo, si asumimos que cada crédito ECTS equivale a 25-30 horas).

Además, debemos tener en cuenta que la carga lectiva actual de la titulación de Ingeniería Informática en la UA es claramente excesiva (364,5 créditos frente a los poco más de 300 de otras universidades, i.e. un 21% superior). Aún más, con la conversión realizada en este trabajo hemos encajado la docencia impartida actualmente en 5 años en los como mucho cuatro años (tres de asignaturas más uno de trabajos tutorizados) de la nueva titulación. Una alternativa (que no hemos aplicado por desconocer cuál va a ser la política a aplicar en los títulos de Máster) sería haber considerado no 240 ECTS (que pensamos que será lo que se convalide a los Ingenieros Informáticos) sino 300 (grado+máster) para aplicar los porcentajes. Creemos que es absolutamente necesaria una reflexión profunda en el seno de la universidad para discernir cuáles son en verdad contenidos fundamentales y cuáles se podrían trasladar a un futuro título de Master, para de este modo no sobrecargar al alumno con contenidos. En esta futura reorganización de asignaturas, la Programación Orientada a Objetos es claramente un contenido del título de grado, ya que su temario supone la primera aproximación específica a este paradigma de programación.