Aprendizaje Basado en Proyectos aplicado a la asignatura Ingeniería del Software: Guía del docente y Guía del estudiante.

Alfredo Goñi, Jesús Ibáñez,
Jon Iturrioz y José Ángel Vadillo

UPV/EHU / LSI / TR 03-2012

PROGRAMA ERAGIN III

GRUPO DE INGENIERIA DEL SOFTWARE:

ALFREDO GOÑI

JESÚS IBÁÑEZ

JON ITURRIOZ

JOSÉ ÁNGEL VADILLO

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

FACULTAD DE INFORMÁTICA

GUÍA DEL DOCENTE

Índice

1. Descripción y contexto de la asignatura

1.1. Información general

La asignatura objeto del presente proyecto docente es **Ingeniería del Software,** que se imparte actualmente en tres idiomas (castellano, euskara e inglés) en la **Facultad de Informática** de la UPV/EHU, Campus de Gipuzkoa. Pertenece al segundo curso del **Grado en Ingeniería Informática** y es cursada en su segundo cuatrimestre.

La asignatura tiene asociados un total de 6 créditos ECTS, siendo 4 teóricos y 2 prácticos. Este creditaje le supone al alumno o alumna una dedicación horaria total de 60 horas presenciales y 90 horas no presenciales a lo largo del cuatrimestre. De acuerdo a los criterios de planificación establecidos por el Centro, esta carga horaria se distribuye entre 15 semanas presenciales y 3 semanas dedicadas a la finalización de la evaluación (tanto continua como conjunta, por lo que la carga se distribuye de la siguiente forma:

- 4,5 horas presenciales (3 módulos de 1,5 horas cada uno) a lo largo de las 15 semanas presenciales
- el tiempo dedicado a evaluación en la semana 17 ó 18
- una media de 5 horas no presenciales durante las 18 semanas.

No obstante, la distribución semanal no es homogénea, por lo que hay semanas en las que la carga es superior a la media y otras en las que es inferior, sobre todo en lo que respecta a la no presencialidad.

El hecho de que la asignatura se imparta en inglés no proviene de la definición del plan de estudios sino a una decisión estratégica del Centro. Según los compromisos adquiridos en el marco del Plan EHUNDU nos hemos comprometido a ofertar todo el segundo curso de la titulación en inglés, e Ingeniería del Software es una de las cuatro asignaturas por las que se ha comenzado dicha oferta. Es de esperar por tanto que esta sea estable en el tiempo con los tres grupos de castellano, euskara e inglés.

A la vista de los números de matrícula en primero para el curso 2011/12, de la experiencia en los dos años anteriores y de las previsiones de resultados a final de cuatrimestre se espera un **número de alumnos/as** entre 50 y 60 (20-25 en los grupos de castellano y euskara y 10/15 en el grupo de inglés).

1.2. Competencias del Grado.

Ingeniería del Software forma junto con la asignatura Bases de Datos una de los cuatro submódulos de la Rama Común Informática del Grado, que vertebra la enseñanza obligatoria después de la Formación Básica, tal y como se puede observar en la tabla 1 que figura a continuación:

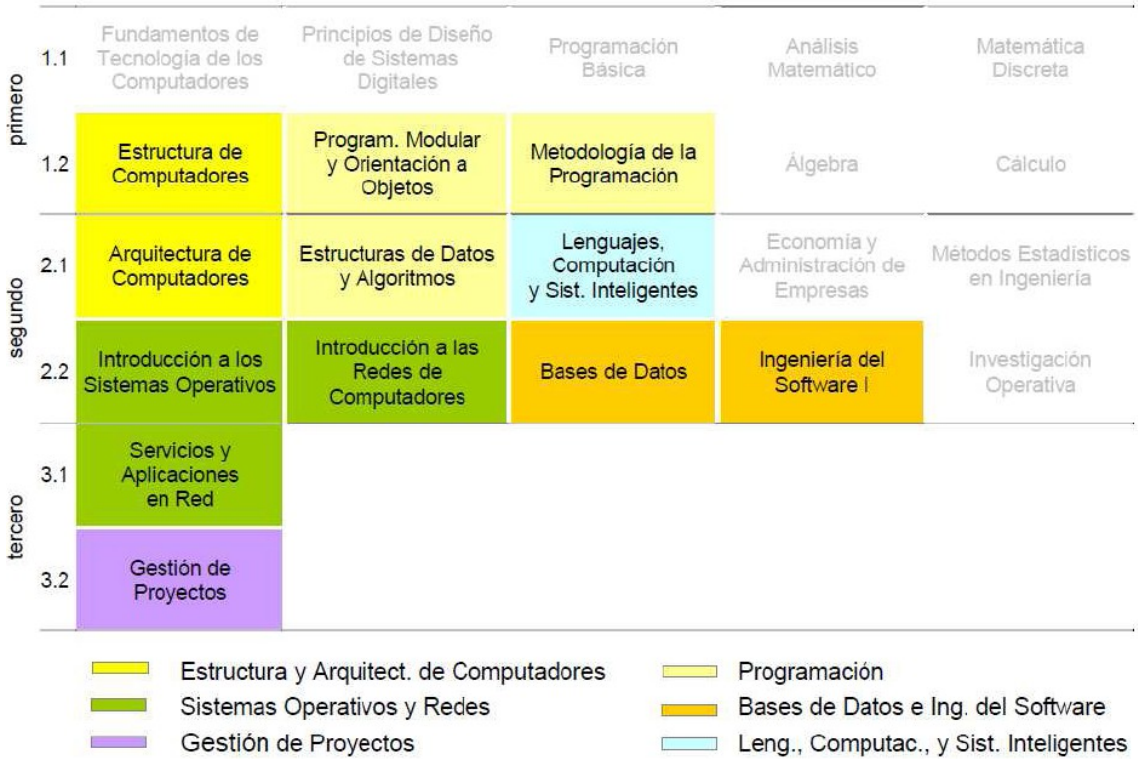


Tabla 1: Módulos y asignaturas de la Rama Común Informática.

Dada su orientación hacia la adquisición de competencias profesionales fundamentales, el **Módulo de Bases de Datos e Ingeniería del Software** tiene una enorme responsabilidad en cuanto a asegurar la adquisición de las Competencias Generales de la Titulación, y de ello es buena prueba la inusual extensión de la lista de las que se le asignan[1]. De acuerdo con ello, las dos asignaturas del Módulo deben garantizar la adquisición de las competencias **RI** correspondientes a la Rama Común de Informática que le corresponden según la tabla 2, y que son las siguientes:

RI1: Capacidad para **diseñar, desarrollar,** seleccionar y evaluar **aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad**, conforme a los principios éticos y a la legislación y normativa vigente.

[1] http://www.ehu.es/documents/340468/516505/Lista+de+competencias.pdf

RI2: 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.

RI3: Capacidad para **comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software**.

RI5: **Conocimiento**, administración **y mantenimiento de sistemas, servicios y aplicaciones informáticas**.

RI8: Capacidad para **analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados**.

RI12 Conocimiento y **aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos**.

RI13 Conocimiento y **aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los sistemas de información, incluidos los basados en web**.

RI16: **Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.**

RI17:Capacidad para **diseñar** y evaluar **interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas**.

		Competencias específicas de la rama común informática (RI)																
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Programación	(18 cr.)	x					x	x	x										
Estruct. y Arquit. de Comput.	(12 cr.)	x								x									
Sist. Operativos y Redes	(18 cr.)					x					x	x		x	x				
B. de Datos e Ing. del Soft.	(12 cr.)	x	x	x		x			x				x	x			x	x	
Leng., Computac. y Sist. Int.	(6 cr.)						x									x			
Gestión de Proyectos	(6 cr.)	x	x	x	x														x

Tabla 2: Distribución de las competencias RI (Rama Común Informática) en los distintos módulos.

Por otra parte, el **Documento de Grado** determina las competencias transversales a desarrollar en la asignatura en términos de Competencias Generales de la Titulación. En concreto, en el módulo de la asignatura deben desarrollarse las competencias transversales que le corresponden según la tabla 3 de asignación de competencias de la titulación. El listado de las competencias **C** mencionadas es el siguiente:

C1: Capacidad para **concebir, redactar, organizar, planificar, desarrollar** y firmar **proyectos en el ámbito de la ingeniería en informática**, que tengan por objeto, de acuerdo con los conocimientos adquiridos, el **desarrollo** o la explotación **de sistemas, servicios y aplicaciones informáticas**.

C2: Capacidad para **dirigir las actividades objeto de los proyectos del ámbito de la informática** de acuerdo con los conocimientos adquiridos.

C3: Capacidad para **diseñar, desarrollar, evaluar y asegurar la accesibilidad, ergonomía, usabilidad y seguridad de los sistemas, servicios y aplicaciones informáticas**, así como de la información que gestionan.

C4: Capacidad para **definir, evaluar y seleccionar plataformas** hardware y **software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas**.

C5: Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas **empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad**.

C6: Capacidad para **concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas** integrando hardware, software y redes de acuerdo con los conocimientos adquiridos.

C8: Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran **versatilidad para adaptarse a nuevas situaciones**.

C9: Capacidad para **resolver problemas con iniciativa, toma de decisiones, autonomía y creatividad**. Capacidad para **saber comunicar y transmitir los conocimientos, habilidades y destrezas de la profesión de Ingeniero Técnico en Informática**.

C10: Conocimientos para la **realización de mediciones**, cálculos, valoraciones, tasaciones, peritaciones, estudios, informes, **planificación de tareas y otros trabajos análogos de informática**.

			Competencias de la titulación										
			C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
	Programación	(18 cr.)	X				X	X		X	X	X		
	Estruct. y Arquit. de Comput.	(12 cr.)				X				X	X			
	Sist. Operativos y Redes	(18 cr.)				X	X	X		X	X	X		
	B. de Datos e Ing. del Soft.	(12 cr.)	X	X	X	X	X	X		X	X	X		
	Leng., Computac. y Sist. Int.	(6 cr.)								X	X			
	Gestión de Proyectos	(6 cr.)	X	X					X		X	X	X	X

Tabla 3: Distribución de las competencias C (Titulación) en los módulos de la Rama Común.

Dentro del módulo la asignatura de Ingeniería del Software hará hincapié en todas las Competencias mencionadas, tanto de la Titulación comos específicas de la Rama Común de Informática. Se han resaltado en color los conceptos más directamente implicados en la asignatura.

1.3. Descripción general de la asignatura

El objetivo principal de la asignatura es capacitar a los alumnos para diseñar e implementar aplicaciones de entidad y complejidad manifiestamente superiores a las que están acostumbrados tras su primer año y medio de curriculum. Se pretende por tanto dotarles de herramientas y métodos que hagan viable el salto a la producción de un software más cercano al del mundo real. El corpus teórico que vertebra los distintos elementos utilizados en el curso es el Proceso Unificado de Desarrollo del Software (UP), considerado hoy en día como un estándar.

Así se introduce la dificultad de capturar los requisitos y producir un diseño correcto y eficaz, y al mismo tiempo se proporciona el lenguaje de modelado UML como herramienta en la que apoyarse para enfrentar dicha dificultad y desarrollar los proyectos siguiendo un proceso sistemático.

El desarrollo de software de calidad requiere que sus componentes sean reusables y que se pueda modificar a bajo coste para afrontar cambios en las necesidades o portabilidad a otros entornos, por lo que se introduce el concepto de arquitectura software de varios niveles.

Por último se introducen asimismo herramientas de desarrollo para incorporar a las aplicaciones desarrolladas por los estudiantes dos características ubicuas en las aplicaciones de uso real: la capa de persistencia y el entorno cliente/servidor.

1.4. Temario

La asignatura está articulada en cuatro temas:

TEMA 1: Captura de requisitos usando UML
- 2.1.- Modelo de Casos de Uso
- 2.2.- Modelo del Dominio

TEMA 2: Diseño
- 3.1.- Diagramas de secuencia UML
- 3.2.- Patrones de responsabilidad GRASP
- 3.3.- Arquitecturas Software de varios niveles

TEMA 3: Implementación usando Java
- 4.1.- Interfaces de usuario gráficas: SWING/AWT
- 4.2.- Persistencia de objetos: db4o
- 4.3.- Computación distribuida: RMI

TEMA 4: Aspectos complementarios de la Ingeniería del Software

El Tema 4 es muy reducido y normalmente se ha solido impartir a modo de introducción, concentrándose en una motivación general de la disciplina que incluye información contextual de tipo histórico. Dadas las decisiones estratégicas tomadas en el presente proyecto docente hemos estimado más conveniente trasladarlo al final.

Se pretende evitar la saturación de contenidos teóricos restringiendo el foco de la asignatura a las fases, modelos y herramientas estrictamente imprescindibles para que el estudiante pueda ir realizando un trabajo práctico de cierta complejidad en el que ponga a prueba lo estudiado. Este trabajo (al que llamamos la Práctica) se desarrolla durante todo el curso, pero es en la fase final (implementación) cuando exige una mayor carga no presencial del alumno. Para la semana 12 del curso los estudiantes tienen que haber adquirido la batería de conocimientos y técnicas necesarios, y las tres últimas semanas deben concentrarse casi en la resolución de los múltiples problemas y desafíos que surgen al implementar la Práctica.

Así por ejemplo la fase de pruebas, extraordinariamente importante para cualquier metodología de desarrollo del software, es excluida del temario en beneficio de un curso posterior (Ingeniería del Software II). Tampoco se trabajan todos los posibles artefactos que UP proporciona, limitándonos a un conjunto mínimo de modelos que permita expresar los diferentes niveles de abstracción. Por último, las características que para los estudiantes son nuevas en el desarrollo del software (interfaces gráficas de usuario, persistencia de datos y computación distribuida) se presentan utilizando herramientas muy accesibles y con una curva de aprendizaje relativamente suave (AWT/SWING, DB4O y RMI).

1.5. Conocimientos previos

El análisis de requisitos y el diseño se realizan utilizando el paradigma orientado a objetos. Los estudiantes deben haber adquirido cierta destreza en el manejo de los principios que sustentan dicho paradigma, ya que los han estudiado en profundidad en la asignatura Programación Modular y Orientada a Objetos (primer curso, segundo cuatrimestre) y han aplicado en la asignatura Estructura de Datos y Algoritmos (segundo curso, primer cuatrimestre).

Respecto a la fase de implementación, para abordarla con garantías de éxito es imprescindible que los alumnos tengan soltura en la concepción, codificación y prueba de algoritmos en Java, utilizando un entorno de desarrollo como puede ser Eclipse. Esta capacidad se ha adquirido de manera gradual desde la asignatura introductoria

Programación Básica (primer curso, primer cuatrimestre), pasando por Programación Modular y Orientada a Objetos (primer curso, segundo cuatrimestre), y Estructuras de Datos y Algoritmos (segundo curso, primer cuatrimestre).

Sólo se asume un conocimiento superficial de Bases de Datos, ya que la asignatura homónima se imparte simultáneamente. Es por ello que en el curso se utilizan un SGBDOO (Sistemas de Gestión de Bases de Datos Orientadas a Objetos) como db4o, ya que permite realizar operaciones relativamente complejas sin necesidad de aparato teórico adicional.

Con el objetivo de reforzar los conocimientos previos necesarios para el desarrollo del Proyecto se propondrá una serie de laboratorios guiados opcionales, que el alumno/a puede desarrollar para afianzar aspectos relacionados con la orientación a objetos, polimorfismo, ligadura dinámica, etc….

1.6. Competencias específicas y objetivos de aprendizaje

Una vez superada la asignatura, el alumno/a deberá haber adquirido determinadas competencias específicas de la materia, es decir, deberá ser capaz de:

- **CA1:** Distinguir las diversas etapas que componen todo proceso de ingeniería del software.
- **CA2:** Entender un sistema software con orientación a objetos descrito mediante el lenguaje UML
- **CA3:** Diseñar un sistema software en una arquitectura de varios niveles a partir de los requisitos.
- **CA4:** Implementar un sistema a partir del diseño de la aplicación

A partir de aquí podemos matizar con un poco más de detalle los objetivos de aprendizaje. Damos una descripción abstracta de cada uno de ellos y a continuación lo analizamos en términos de competencias concretas y mensurables:

1. **OA1:** Conocer la existencia de metodologías basadas en etapas para afrontar la enorme complejidad que puede alcanzar un proyecto informático. Este objetivo tiene relación con la competencia **CA1**.
    - Describir a grandes rasgos la necesidad histórica, el surgimiento y la evolución de la disciplina Ingeniería del Software.

- Describir a grandes rasgos las principales metodologías de desarrollo del software y los ciclos de vida que propugnan.
- Distinguir y conocer las distintas etapas del desarrollo del software, y muy especialmente las de Análisis de Requisitos, Diseño, Implementación y Pruebas.

2. **OA2:** Experimentar la necesidad de aplicar una metodología para el desarrollo de un proyecto informático en sus diferentes etapas del ciclo de vida. El alumno debe conocer y saber aplicar ambos para adquirir la competencia **CA2**.
   - Seguir con aprovechamiento las diversas etapas que componen todo proceso de Ingeniería del Software.
   - Dominar los principios y disciplinas del Proceso Unificado del desarrollo de Software (UP).
   - Manejar los conceptos y principales diagramas del Lenguaje Unificado de Modelado en su aplicación a la Ingeniería del Software.
   - Manejar la herramienta de modelado StarUML para documentar y expresar de forma visual los resultados de las distintas fases del UP.
3. **OA3:** Entender la necesidad y saber abordar las dificultades de la captura de requisitos del cliente, persona o personas que encargan el sistema. Generalmente se trata de no informáticos y es imprescindible el establecimiento de un lenguaje común. Las técnicas para solucionar este problema están relacionadas con la competencia **CA2** y sirven de puente para empezar a trabajar **CA3**.
   - Identificar los requisitos de un sistema propuesto a partir de las especificaciones (QUÉ QUIERE) de un determinado cliente.
   - Utilizar con soltura Diagramas de Casos de Uso y Flujos de Eventos para modelar los requisitos capturados.
   - Utilizar con soltura y precisión los Diagramas de Clases para construir un Modelo del Dominio que incluya todos los objetos necesarios para cumplir con las especificaciones del cliente.
4. **OA4:** Entender la necesidad y saber abordar las dificultades del análisis y diseño de un sistema informático cumpliendo fielmente con las especificaciones, gestionando su evolución y describiéndolo de forma que el equipo de trabajo comprenda en cada momento el punto en que se encuentra. La adquisición de habilidades en el

uso de herramientas para expresar mediante diagramas y texto el análisis y diseño del sistema es el corazón de la competencia **CA3**.

- Obtener un diseño a partir de los requisitos del sistema (CÓMO SE HACE).
- Utilizar con soltura los Diagramas de Clases para completar el Modelo de Dominio con las utilidades necesarias para diseñar la aplicación.
- Diseñar y construir aplicaciones mediante arquitecturas de tres capas (presentación, lógica de negocio y gestión de datos), utilizando con criterio sus principios y dominando técnicas específicas para ello.
- Utilizar con soltura Diagramas de Diseño para modelar la cooperación entre objetos de las diferentes clases.
- Aplicar conceptos básicos de patrones en el diseño de una solución (GRASP).

5. **OA5:** Entender la necesidad y saber abordar las dificultades de una correcta interpretación de la especificación de diseño de un sistema informático y de su traducción en un desarrollo correcto.. De este modo se efectuará la transición entre **CA3** y **CA4** y se desarrollará plenamente esta última.
   - Manejar con soltura el uso del entorno de programación (ECLIPSE) para el desarrollo y pruebas de un sistema software.
   - Utilizar una infraestructura gráfica (AWT/SWING) para dotar a las aplicaciones de una interfaz de usuario orientada a las necesidades del usuario.
   - Utilizar un un sistema de gestión de bases de datos (db4o) para implementar la capa de persistencia de una aplicación.
   - Utilizar una infraestructura de ejecución distribuida (RMI) para dotar a las aplicaciones de una arquitectura cliente/servidor.
   - Utilizar el lenguaje de programación Java para acometer todas las tareas de implementación listadas en este objetivo de aprendizaje.

Hemos descrito TODOS los objetivos formativos de la asignatura de Ingeniería del Software en lugar de centrarnos en aquellos afectados por el Proyecto. Ello se debe a que nuestra meta es que el trabajo que realicen en el marco de la metodología PBL incida en TODAS las competencias fundamentales, y por tanto trabaje TODAS las fases de la Ingeniería del Software.

Sólo en casos muy puntuales (algunas competencias del objetivo OA1) necesitarán un refuerzo que no puede ser integrado en el Proyecto. Para ellos se han diseñado Actividades complementarias que formarán parte del diseño de la asignatura (véase el apartado 3.3.5). Por ello consideramos que el alumno, al terminar el Proyecto habrá adquirido TODAS las competencias listadas bajo los objetivos de aprendizaje.

Además, nos gustaría resaltar la estrecha relación entre estos objetivos de aprendizaje y las siguientes competencias de la titulación (véase la sección 1.2 para su definición): **C1**, **C2**, **C3**, **C5**, **C9**, **C10**, **RI1**, **RI2**, **RI3**, **RI5**, **RI8** y **RI16**.

2. Formulación general del Proyecto

Pasaremos a describir, primero de manera sucinta y después en detalle, los elementos y la estructura del curso que hemos diseñado. Dado que el porcentaje del mismo que se impartirá con la metodología PBL creemos que es mejor una exposición que englobe a toda la asignatura, mencionando en cada caso los módulos de la misma que no se consideren PBL y que en todo caso estarán supeditados a los que sí lo son.

2.1. Aspectos de la asignatura que influyen en el planteamiento inicial

Hay una serie de características de la asignatura que han guiado el diseño del presente proyecto docente. Las expondremos en primer lugar porque proceden de nuestra experiencia como profesores de la misma y ayudan a entender muchas de las decisiones que se han tomado.

2.1.1. Técnicas y herramientas: necesidad, dominio y elección

La asignatura tiene un fuerte componente técnico. Las herramientas y artefactos utilizadas en el proceso de desarrollo del software se caracterizan por su multiplicidad y por la complejidad de su uso combinado (que al fin y al cabo es lo que asegura que la producción de software pueda acometerse con estándares de tipo industrial). El alumno debe dominar un subconjunto de ellas y luego desarrollar criterios para decidir cuándo son útiles y cuándo un sobrecarga de trabajo, por no mencionar un estorbo.

Es por ello que los objetivos de aprendizaje **OA3**, **OA4** y **OA5** tienen el mismo encabezamiento "Entender la necesidad y saber abordar las dificultades de ...". Es una forma de compactar tres objetivos de aprendizaje:

1. El alumno tiene que reconocer la dificultad de la tarea
2. El alumno tiene que darse cuenta de que existen técnicas y herramientas que permiten reducirla en gran medida
3. El alumno tiene que aprender a manejar con soltura las técnicas o herramientas mencionadas, resolviendo problemas que antes era incluso incapaz de plantear

En caso de que el estudiante no experimente el salto cualitativo que produce su uso se limitará a desarrollar las técnicas sin criterio y por puro imperativo académico. Esto sugiere que la metodología PBL puede proporcionar una ayuda valiosa, ya que es el propio estudiante el que tiene que buscar los medios para superar las dificultades.

Pero no existe en nuestra asignatura ni siquiera el planteamiento de que el estudiante pueda intervenir en la elección de las técnicas y herramientas más adecuadas para cada problema, entre otras cosas porque decidir cuáles satisfacen mejor nuestras necesidades es una obra mucho más compleja. De hecho nuestra selección concreta dista mucho de ser casual: es más bien el resultado de años de experiencia con asignaturas del área en el intento de maximizar el número de problemas resolubles con el mínimo esfuerzo de adaptación a las exigencias de las técnicas utilizadas. Nuestro cóctel es:

- **Análisis y diseño orientado a objetos.** Como hemos mencionado se trata de un estándar, pero el esfuerzo necesario para que los alumnos lo asuman como tal sería enorme, ya que deberían conocer las diferentes alternativas y tener elementos de juicio para compararlas, cosa que es muy difícil sin una cierta experiencia en el desarrollo de proyectos software de cierta enjundia.
- **Lenguaje de programación Java.** Una vez hecha la elección anterior, el uso de lenguajes no orientados a objetos introduciría muchas disfunciones en el desarrollo que habría que resolver dedicando un tiempo del que no se dispone. En todo caso una elección equivocada (por ejemplo, porque el alumno tenga experiencia previa en un lenguaje orientado a procedimientos como Ada o Pascal) puede ser fatal. Sí podría usarse otro lenguaje orientado a objetos como C++ o C#.
- **Entorno de programación Eclipse.** Es difícil encontrar una herramienta tan completa, tan intuitiva y con un soporte parecido. Por otro lado los estudiantes vienen utilizándolo en asignaturas anteriores, especialmente en Programación Modular y Orientada a Objetos. Pero es factible dejar al alumno cambiar a un entorno que le resulte más familiar siempre que respete las dos elecciones anteriores.
- **El Proceso Unificado de Desarrollo del Software (UP) como ciclo de vida de referencia.** En este punto tenemos una cierta ventaja, ya que UP es un proceso lo suficientemente general como para que tanto el estudiante como el profesor vaya tomando las características y elementos que le van siendo necesarias. El profesor tiene que insistir en la libertad que tiene el alumno para tomar aquello que le resulta útil para los objetivos que se levan planteando, aunque procurando que no se desestimen algunos artefactos y herramientas por simple desconocimiento. Por tanto siempre hay que saber como mínimo un poquito más de lo que se usa.

- **Artefactos UML: diagramas de Casos de Uso, Flujos de Eventos y Diagramas de Secuencia.** El uso de UML como lenguaje de modelado es indiscutible, hasta el punto de que podríamos confiar en que los alumnos lo descubran por sí mismos sin mucho esfuerzo. Lo que no resulta en absoluto evidente es qué partes de UML resultan útiles en una primera aproximación, y de nuevo es tarea del profesor podar el árbol de búsqueda y orientar a los estudiantes a que utilicen exactamente los artefactos indicados. Hemos probado con otras posibilidades (por ejemplo, Diagramas de Colaboración), y aunque prácticamente cualquier opción adicional resulta enriquecedora el tiempo necesario para evaluarla y dominarla no compensa por el retraso que introduce en el curso en detrimento del objetivo principal: que la aplicación resultante tenga un aspecto mucho más cercano al del mundo real.
- **StarUML como herramienta de modelado.** Aún cuando los estudiantes son alentados a desarrollar la documentación y los artefactos a mano cuando ello supone una ventaja (papel y lápiz, pizarra blanca y roturadores, …) hay ocasiones en que el uso de una aplicación puede resultar muy eficiente. Lo cierto es que no hay demasiada elección si se quiere emplear una que no sea de pago.
- **Metodologías ágiles.** Sin renunciar al marco general del UP los estudiantes también son instruidos para evitar la sobredocumentación. El uso de principios de las metodologías ágiles reducen mucho la curva de aprendizaje y permiten la realización de proyectos más dinámicos. Esta cuestión será clave en el desarrollo del Proyecto PBL que tratamos de introducir en el presente informe.
- **Java AWT/SWING para el diseño de interfaces gráficas de usuario (GUI).** El diseño de GUI's es una habilidad nueva y muy motivadora para los estudiantes, por lo que es importante dotarles de herramientas que les permitan dar rienda suelta a su creatividad sin desviar demasiado su atención.
- **Db40 para implementar la persistencia.** Como se ha indicado, no se puede suponer que los estudiantes tengan soltura en el manejo de bases de datos. Por otro lado su aprendizaje paralelo en la asignatura concurrente de Bases de Datos se desarrollará bajo el paradigma más convencional de las BD relacionales. ¿Por qué entonces introducir el esquema nuevo introducido por las BD orientadas a objetos? La respuesta es que, por un lado, poner en marcha **db4o** y efectuar mediante el mismo todas las operaciones fundamentales de un BD es tan sencillo que puede ser entendido en apenas una semana (con clases magistrales). Por otro lado la

integración de BD relacionales en sistemas orientados a objetos requiere el uso de puentes JDBC/ODBC amén de tener que diseñar la conversión entre dos modelos de datos muy diferentes.

- **Java RMI para desarrollar aplicaciones distribuidas.** De nuevo tenemos a nuestro alcance una herramienta sencilla, tal vez no demasiado potente, pero que permite al estudiante construir una arquitectura cliente-servidor con muy poco esfuerzo, ya que se integra perfectamente con todos los elementos del análisis y desarrollo orientados a objetos.

Dicho todo esto, ¿cómo conseguir que el estudiante tenga un poco de autonomía para "descubrir" por sí mismo una lista tan larga de tecnologías y metodologías sin perderse en el propio proceso de búsqueda? La aproximación que vamos a elegir es la propuesta de ejemplos. Siempre que sea posible mantendremos la siguiente aproximación:

1. Los problemas se le proporcionarán a los grupos de estudiantes como parte del Proyecto vendrán con una solución parcial, es decir, bajo la hipótesis de que algunas características ya están resueltas y otras no.
2. El trabajo con versiones ya iniciadas permitirá que los estudiantes experimenten la incertidumbre y, por qué no decirlo, la frustración de tener que revisar, corregir y ampliar un trabajo realizado por otros en condiciones de insuficiente documentación. Posteriormente se les proporcionará la documentación que faltaba para que aprecien la diferencia, y por tanto la utilidad de los distintos artefactos.
3. Paralelamente podremos introducir características ya resueltas precisamente con la tecnología que queremos que "descubran". Como los grupos estarán obligados a examinar en detalle la versión proporcionada, encontrarán estas soluciones y podrán apreciar su sencillez (como hemos dicho, uno de los elementos esenciales de nuestro curso es el esfuerzo que hemos hecho en seleccionar siempre la opción más sencilla e integrada para que el alumno no pierda tiempo en adaptarse a ella de manera innecesaria).

2.1.2. Experiencia a incorporar de la Práctica de la asignatura

Nos inspiraremos en la Práctica de los últimos años desarrollada mediante metodología docente tradicional porque su resultado tiene las características de un Proyecto (problema real, fases o hitos, adquisición de competencias profesionales y producto final en servicio).

1. La transformación de práctica individual en proyecto colaborativo nos hace pensar que podemos ser más ambiciosos con el alcance del mismo.
2. Tenemos que relajar la supervisión que actualmente ejercemos. Nuestra opción será utilizar al principio técnicas como las indicadas al final del apartado anterior y permitir hacia el final una mayor autonomía y expresión de la creatividad y dinámica de los grupos. Esto permitirá a los estudiantes discernir claramente los mínimos de la asignatura de aquello que pueden realizar para obtener resultados académicos más brillantes.
3. Al igual que hemos intentado hacer en el pasado, en lugar de plantear un proyecto con especificaciones fijas que orientan al estudiante a ir completando cada fase de desarrollo antes de pasar a la siguiente, iremos planteando el problema a resolver en forma de iteraciones, característica esencial del UP. Esto quiere decir que cada grupo deberá enfrentarse a un enunciado inicial con un producto sobre el que trabajar y unos objetivos a superar. Completar los objetivos exigirá del estudiante recorrer todas las fases de desarrollo, desde el análisis de requisitos hasta la implementación, y una vez terminados se le suministrará una extensión del enunciado pidiendo nuevas funcionalidades que obligarán a replantear el producto y a recorrer de nuevo el proceso. En cada iteración se volverán a revisar y profundizar las técnicas y conceptos de cada una de las fases, resolviendo cada vez problemas más enjundiosos.
4. Además de producir un desarrollo iterativo procuraremos introducir elementos propios de las metodologías ágiles: iteraciones cortas limitadas por el recurso tiempo más que por los resultados prefijados, flexibilidad ante el cambio propugnando talleres de redefinición de requisitos, limitación de la documentación a sus aspectos más útiles (evitando contemplarla como una disciplina inflexible), determinación de producir código "presentable" al final de cada iteración, es decir que funcione como para enseñárselo al cliente, por oposición a prototipos desechables que funcionan como esqueletos de lo que aún está por decidir.

2.1.3. Heterogeneidad del alumnado

Como se ha indicado los alumnos vienen con expectativas y competencias muy dispares, y en este contexto una aproximación tipo "café para todos" puede ser bastante ineficiente . Por el contrario un Proyecto de las características del que proponemos podrá dimensionarse en función de los intereses y capacidades de cada grupo. En particular, en

la iteración final se permitirá que los grupos indiquen y justifiquen sus propios objetivos de mejora del producto, de manera que el resultado de su trabajo refleje su competencia, su ambición y su realismo.

2.1.4. Enunciado(s) del Proyecto

Es importante que los enunciados reflejen las características de los proyectos reales, con las dificultades inherentes de los mismos: mantendremos la incertidumbre e incoherencia propia de los clientes que no tienen una formación técnica, para que sean ellos los primeros que se den cuenta de que tienen un problema básico: entender lo que deben hacer y que el cliente entienda lo mismo. Este planteamiento introduce asimismo variedad, ya que hay un margen de interpretación con el que cada grupo podrá jugar.

Esta aproximación también introduce la noción de que las soluciones siempre se pueden mejorar, y de que la mejora se debe detener en algún momento por la limitación de recursos. En este sentido la idea que queremos transmitir es propia de las metodologías ágiles: intentaremos que cada grupo desarrolle *la mejor aplicación posible en el tiempo fijado por el Proyecto*.

2.2. Planteamiento de la asignatura en su versión PBL

Tal y como se ha indicado el Proyecto para la asignatura se planteará en forma de enunciado incremental que se enriquece a medida que los estudiantes van superando las iteraciones. Asimismo es muy importante la idea de suministrarles junto al enunciado una solución parcial, es decir un punto de partida que, por un lado, les limita y obliga a hacer un trabajo de análisis y comprensión, y por otro les da pistas acerca de cómo pueden abordar los problemas.

2.2.1. Escenario

Antes incluso de la formación de grupos el planteamiento será el siguiente. Cada estudiante acaba de llegar a la importante empresa Sinking Soft. Inicialmente el desconcierto es total: el recién llegado no sabe qué se espera de él, y el equipo técnico que le recibe tampoco sabe hasta qué punto puede contar con él ni tiene tiempo o ganas de averiguarlo. Cada estudiante deberá enfrentarse a esta situación (primero en solitario y poco después en un equipo de trabajo).

Una de las primeras cosas que le sucederán es que le caerán "marrones": trabajos que han quedado sin terminar y que hay que completar para que cumplan todas las especificaciones. Inicialmente se le darán mal documentados, pero progresivamente

empezará a ver que otros trabajos vienen acompañados de información útil (artefactos) que simplifican mucho su tarea. Si consigue superar este estadio inicial con éxito tendrá un encargo de mayor responsabilidad (el Proyecto, que también será un trabajo dejado a medias por otros compañeros) formando un equipo, pero siempre teniendo en cuenta que su actuación está a prueba y que hasta que no complete dicho proyecto no se le ofrecerán responsabilidades reales ni condiciones laborales mejores.

2.2.2. Pregunta motriz

¿Tú crees que vas a durar mucho en Sinking Soft?

2.2.3. Formación de grupos

Inicialmente se propondrá la actividad individual **P0** (véase la sección 3.3) que permitirá a los estudiantes una cierta inmersión en el escenario e interacción suficiente como para formar los grupos del Proyecto, y al profesor hacer una evaluación del grupo y de las expectativas de seguimiento de la evaluación continua.

La realización de esta tarea individual servirá de filtro para determinar qué estudiantes están dispuestos a realizar el esfuerzo de dedicación sistemática exigible para la evaluación continua, y por ello no podrán integrarse en grupos para el Proyecto quienes no hayan completado la misma.

Pero muchos ejercicios proporcionados a lo largo del curso irán inmersos en el mismo escenario, aún sin ser parte integral del Proyecto. Por ejemplo, para las tres primeras semanas los estudiantes recibirán un producto software incompleto supuestamente desarrollado en Sinking Soft y dejado a medias por un desarrollador que no consiguió durar mucho en la empresa. Se trata de un sistema de reservas de canchas de tenis para el club deportivo "El Jubileta Feliz" y se pretende que el estudiante:

- Construya una pequeña interfaz de usuario (objetivo explícito)
- Dote de persistencia a los datos de reserva (objetivo implícito)

Atendiendo a que el número de alumnos previsto en los grupos más grandes es de 25 consideramos que los grupos deben estar formados por un número de ***3 personas***. Menos de tres resulta complejo a la hora de trabajar estrategias como asignaciones e intercambios de roles en diferentes actividades. Para más detalles sobre este punto véase la metodología SCRUM (sección 3.1)

Es importante que los miembros del grupo tengan unas horas comunes libres para poder realizar sus reuniones y este será uno de los criterios que se les pedirá para su formación. Serán los alumnos los que se agrupen inicialmente, solo interviniendo el profesor si se observan conflictos e incompatibilidades a la hora de crearse los grupos (alumnos descolgados, número de alumnos no múltiplo de tres, etc.).

En este sentido no imponemos muchas restricciones, pero sí es importante que se conozcan *a priori* las reglas de juego y lo que supone para el grupo que un integrante no realice las actividades en plazo e intensidad requeridos. La realización de pruebas de evaluación a integrantes del grupo elegidos al azar (método del representante) o la calificación de todo el grupo con la media de las calificaciones individuales son algunas prácticas que se efectuarán a modo de control de la exigibilidad individual y los alumnos deben saber de su existencia y considerarlas a la hora de formar el grupo.

2.3. Presentación del Proyecto

Una vez realizada la actividad individual e iniciada la formación de grupos se presentará el Proyecto propiamente dicho en forma de especificación de necesidades del cliente.

2.3.1. Escenario del Proyecto

"Los habitantes de Villatripas de Arriba han mantenido el aspecto de su pueblo y restaurado sus viviendas para alquilarlas como casas rurales. Sin embargo el negocio no marcha bien: casi todos los turistas se hospedan en Villatripas de Abajo, que además de ser mucho más feo, está habitado por imbéciles, como todo el mundo sabe.

La razón parece estar en que en Villatripas de Abajo tienen una web estupenda que enseguida capta a los clientes potenciales dándoles información útil y atractiva y guiándoles en su elección. Sin embargo, la web de Villatripas de Arriba tiene enlaces muertos, no permite meter información individualizada por cada casa, no obliga al cliente a introducir un adelanto para reservar la casa (con lo que muchas reservas no se cumplen), y además no verifica que las casas cumplan con los requisitos de la Dirección General de Turismo Rural (número mínimo de baños, existencia de cocina, etc...) por lo que han tenido que hacer frente a denuncias.

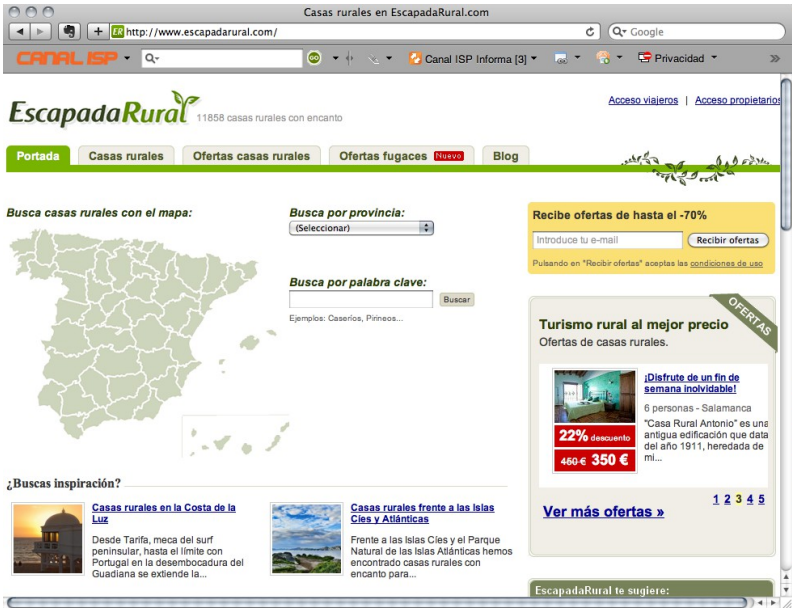
Don Manolo Pillaste, alcalde de Villatripas de Arriba, ha decidido renovar la web, pero como la otra vez le engañaron, quiere estar seguro, antes de hacer el encargo, de que le instalan todas las funcionalidades. Sinking Soft es una de las candidatas y tiene que

construir y presentar una descripción del servicio a desarrollar. Como no parece un cliente importante (la verdad es que sólo han aceptado un presupuesto miserable) la empresa se lo va a encargar al grupo de novatos. Si salen adelante y satisfacen al cliente se les prorrogará el contrato otros cuarenta minutos. Si no, se les echa a la calle y no se les devuelve la fianza que se les ha cobrado en concepto de uso del papel higiénico de la empresa."

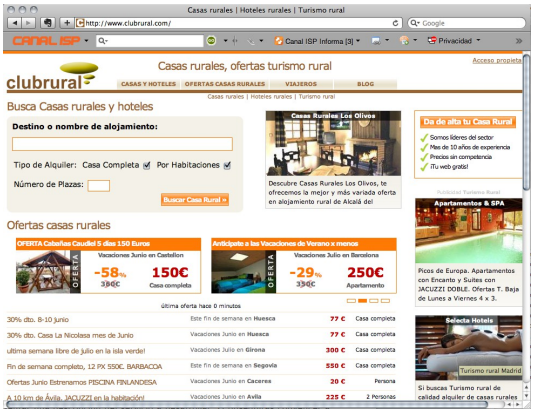
2.3.2. Primeros pasos

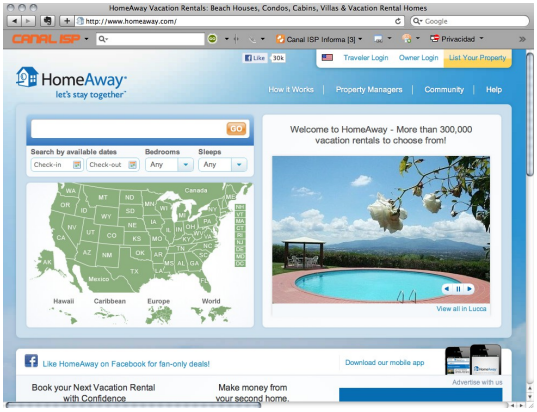
Lo primero que tiene que hacer el estudiante es tener una mínima familiaridad con el problema. Existen en el mercado muchas aplicaciones web que permiten gestionar el alquiler de casas rurales, tanto para clientes como para los propietarios de las mismas. Una búsqueda en Google del texto "casas rurales" nos devolverá infinidad de resultados entre los que se encuentran:

- http://www.escapadarural.com/

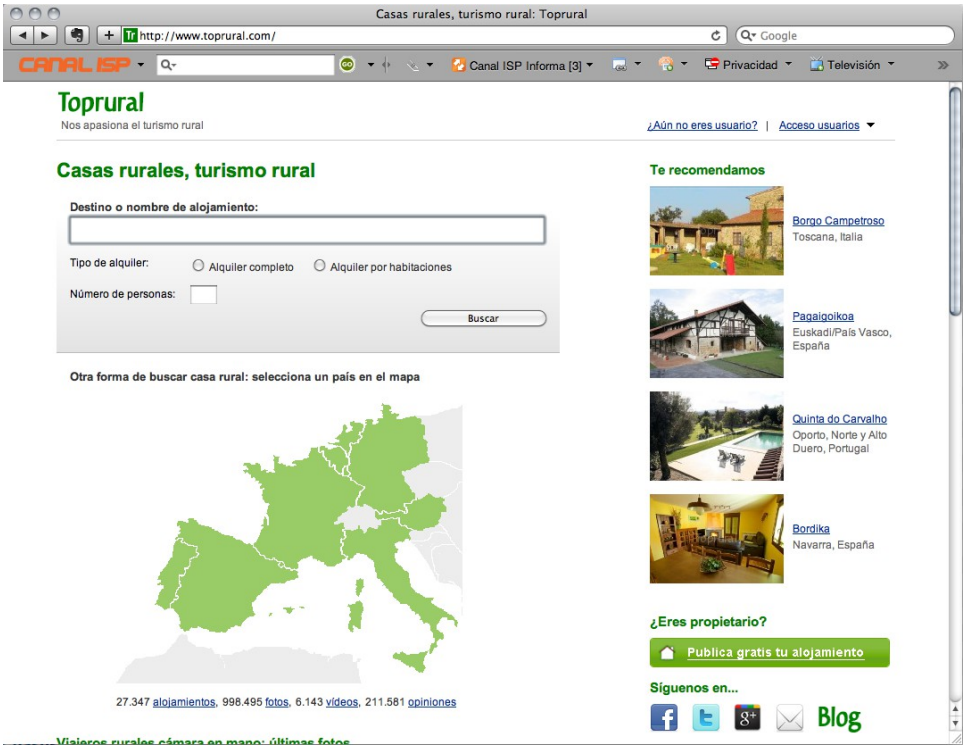


- http://www.clubrural.com/ o http://www.homeaway.es/





- http://www.toprural.com/



El Proyecto que vamos a proponer es el desarrollo de una aplicación de gestión de casas rurales que nazca con el objetivo de superar a las actualmente existentes (que harían el papel de Villatripas de Abajo) en al menos alguna característica. No se espera que el producto final quede en explotación, pero sí que obtenga como resultado un prototipo que sea funcional y que muestre las bondades de la aplicación propuesta.

Cada una de las aplicaciones que se puede encontrar como modelo satisface un conjunto de requisitos de los usuarios de las mismas. Se supone que antes de crear cualquier aplicación hay que realizar un proceso al que se denomina captura de requisitos.

2.3.3. Preguntas guía (procedimiento)

Hay una serie de cuestiones que deben plantearse al inicio de cualquier reunión del grupo con respecto a la tarea en curso, y que son una pequeña adaptación de la metodología PBL:

1. ¿Qué es exactamente lo que debemos hacer? (especificación)
2. ¿Qué es lo que no tenemos por qué hacer? (alcance)
3. De las cosas que sabemos, ¿cuáles pueden ser útiles para resolver la tarea? (recursos)
4. ¿Qué deberíamos saber hacer y no sabemos? (objetivos de aprendizaje)

5. ¿Existe algún procedimiento más o menos estándar para hacerlo? ¿Cuál es? ¿Dónde hay información sobre el mismo? (técnicas)
6. En caso de haber varios procedimientos, ¿cuál es el más sencillo? (limitación de recursos)
7. ¿Hay en la versión incompleta del Proyecto que nos han entregado algo que nos pueda servir como modelo para plantear, resolver o documentar el trabajo restante? (inspiración).

2.3.4. Preguntas guía (contenido)

Adicionalmente tenemos otra batería de preguntas guía orientadas a las fases de desarrollo del Proyecto e irán planteándose dependiendo del momento en la que nos encontremos. Contemplamos cuatro fases:

1. Identificación de los requisitos. ¿Cuáles son las funcionalidades que va a ofrecer nuestro Proyecto?
2. Captura de requisitos. ¿Qué elementos creemos que hay que identificar en la captura de requisitos? ¿Existen procedimientos regulados y eficientes para capturarlos sin errores?
3. Diseño del sistema. Una vez capturados los requisitos, ¿Cómo diseñaremos cada caso de uso? ¿Utilizaremos algún paradigma concreto? ¿Existe algún tipo de artefacto específico para el diseño? ¿Hay algún principio organizativo que facilite el diseño? ¿Existen patrones de diseño ya establecidos? ¿Cómo vamos a realizar la persistencia de los objetos?
4. Implementación del sistema. Una vez diseñados los casos de uso, ¿En qué lenguaje de programación vamos a implementarlo? ¿Que tecnología vamos a utilizar para implementar cada nivel? ¿Qué Base de Datos vamos a utilizar para realizar la programación distribuida (cliente/servidor)?

2.4. Desarrollo iterativo del Proyecto y modificación de la estructura del curso

La Ingeniería del Software ha pasado en su breve historia por una serie de etapas. En ellas se ha intentado aprender de los fracasos y éxitos (por ese orden) cosechados por los equipos de desarrollo de software a gran escala, tanto en empresas como en agencias gubernamentales. Como en muchos otros campos del conocimiento humano estas fases han tenido un carácter pendular, tratando de buscar el punto óptimo entre la gestión y el

desarrollo de los proyectos. Cuatro de las tendencias más importantes que han caracterizado esta evolución son incorporadas como metodologías en nuestro curso de la asignatura Ingeniería del Software I:

1. la necesidad de utilizar una metodología sólida integrada en la cultura de la organización (en nuestro caso el UP)
2. el uso de notaciones lo más estandarizadas posible orientado a producir software manejable, mantenible y reutilizable (en nuestro caso UML)
3. el desarrollo del software en iteraciones, por oposición a los ciclos de vida que pretenden agotar al máximo cada fase antes de proceder a la siguiente
4. la adopción de metodologías ágiles que eviten la sobrecarga de artefactos, fomenten la discusión entre grupos de interés y produzcan versiones ejecutables y utilizables de manera continua.

Como se ve, algunos de estos aspectos están más bien basados en conocer y aplicar una teoría subyacente, mientras que otros no pueden ser inculcados sin una práctica sostenida. Así sucede con las metodologías ágiles en el desarrollo de software, que promueven basar cualquier proceso en breves iteraciones temporizadas (2-4 semanas) en cada una de las cuales se aplican todas las fases del desarrollo (Requisitos, Diseño e Implementación) para obtener cada vez un producto software que mejore al producto desarrollado en la iteración anterior.

Esta metodología dado de carácter iterativo e incremental favorece la trazabilidad y evolución de los proyectos software. Es el tiempo la variable que gobierna el desarrollo y no el producto. De este modo siempre se sabe exactamente en qué fase se encuentra un proyecto, puesto que no existe el concepto de "*fase en la que debería estar*". Por otro lado cobra importancia el concepto de "*el mejor producto que podemos producir en tiempo X*", idea muy interesante desde el punto de vista didáctico, ya que permite que grupos de alumnos con diferentes preparaciones, diferentes capacidades y, sobre todo, diferentes expectativas académicas, puedan sacar el mejor provecho de sí mismos sin verse negativamente afectados por las características de los restantes grupos.

Por ello hemos configurado nuestro proyecto docente siguiendo PBL e inspirados en la propia materia que impartimos. Consideramos que si vemos los objetivos de aprendizaje de esta asignatura como un *producto a desarrollar*, podemos aplicar estas metodologías al diseño y planificación mismos de la asignatura. Nos gustaría resaltar el aspecto novedoso de este enfoque, en el que en vez de desarrollar secuencialmente los

temas, estos se plantean de manera iterativa, revisando y profundizando cada concepto y técnica en varias ocasiones a lo largo del desarrollo del curso *sobre el mismo Proyecto*. Esta aproximación también permite una visión motivadora por parte del alumno, que verá cómo su capacitación aumenta al poder abordar con éxito aspectos progresivamente más complejos y desafiantes, lo que se traduce en sus manos en la transformación de una aplicación raquítica y pobremente documentada en … lo que sea capaz de hacer con su intelecto y su tiempo en los cuatro meses del curso, pero siempre un resultado muy distante del punto de partida.

El Proyecto sufrirá tres iteraciones o "pasadas" en las cuales la progresión será de tipo horizontal. El resultado de cada iteración será un producto ejecutable que se irá enriqueciendo con funcionalidades, aspectos, casos de uso, interfaces y modelos de dominio crecientemente complejos, que estimularán al o a la estudiante a dominar de manera progresiva las diferentes técnicas y herramientas de la Ingeniería del Software. Para ser más exactos, las que nos da tiempo a introducir en seis créditos, contando con su precaria formación previa y con la restricción de que no debe quedar ningún hueco insalvable en su camino hacia el desarrollo de una aplicación digna en fase de explotación (es decir, que el decorado de su actuación sea completamente realista).

Al basar además este proceso iterativo en metodologías ágiles tendremos que en cada una de las "pasadas" se habrán de contemplar todas las fases del desarrollo, ya que en cada iteración se partirá de una ampliación de los requisitos y se concluirá con un producto ejecutable. Esto quiere decir que queda fuera de lugar la estructura secuencial del temario Análisis de Requisitos (Tema 1) ⇒ Diseño (Tema 2) ⇒ Implementación (Tema 3). Más bien los contenidos de la asignatura se aprenderán en oleadas: dado que cada iteración propondrá problemas cada vez más complejos ello dará pie a que en cada una de ellas se amplíe y profundice el estudio de cada fase del desarrollo, y sobre todo a que se dominen más a fondo las técnicas y herramientas que les corresponden.

Por tanto los Temas 1, 2 y 3 (véase la sección 1.4) ya no serán los que doten de estructura al curso, sino que se repartirán a lo largo de las iteraciones del Proyecto. A continuación se describe cuál es el enfoque propuesto para abordar el temario en el marco de PBL.

Iteración 1: Se introduce el Proyecto y se forman los grupos para abordarlo. El planteamiento inicial está orientado a la identificación de los requisitos iniciales del Proyecto. Algunos de estos requisitos pueden estar ya correctamente documentados e incluso implementados, mientras que en otros la fase de diseño será crucial. De entre los

requisitos capturados los estudiantes deben seleccionar los más importantes, teniendo en cuenta su centralidad para la arquitectura de la aplicación, los riesgos estimados para su implementación y, por supuesto, las prioridades del cliente. Se estudiará cómo obtener los diagramas de secuencia a partir de los casos de uso y flujos de eventos seleccionados, cómo conseguir una evolución consistente del Modelo del Diseño a partir del Modelo del Dominio, y cómo mantener la consistencia entre los diferentes artefactos. La iteración termina en fecha prevista de antemano y la solución de los requisitos debe estar incorporada a la aplicación.

En esta iteración el profesor tiene que prestar atención para evitar, por un lado, que los alumnos prescindan del soporte metodológico, y en el otro extremo que adopten metodologías inadecuadas (por ser demasiado restrictivas o por exigir una carga de gestión desproporcionada). También es importante que no seleccionen demasiados requisitos para el tiempo que disponen,

Iteración 2: Se pueden incorporar nuevas especificaciones del cliente, normalmente expresadas con vaguedad. El objetivo consiste en que los alumnos sepan capturar y modelar estos nuevos requisitos funcionales del sistema utilizando Casos de Uso, Flujos de Eventos y enriqueciendo del Modelo del Dominio. La lista de requisitos obtenida por los grupos deberá ser acotada, teniendo en cuenta el tiempo necesario para su desarrollo y su complejidad. El resto de las fases deben realizarse de manera similar a la iteración anterior. Entre las características de implementación nuevas estará el desdoblamiento de la aplicación en una arquitectura cliente/servidor usando RMI.

Iteración 3: A partir de aquí los grupos de trabajo tendrán mayor libertad para proponer modificaciones en las funcionalidades según sus propios criterios, espíritu emprendedor y eficacia trabajando. Los profesores seguirán proporcionando soporte de clientes y usuarios finales para evitar que los estudiantes intenten abarcar funcionalidades de alto coste y reducido valor añadido (no se puede suponer que los estudiantes tengan mucha experiencia ni como propietarios ni como usuarios de casas rurales). El resultado final de esta iteración será el Proyecto resultante del grupo y será evaluado como tal.

Aspectos complementarios (Tema 4): Llegados a este punto es de esperar que los temas 1-3 de la asignatura hayan sido incorporados al proceso de aprendizaje con ventaja, y que una vez desarrollado el Proyecto el alumno sabe apreciar los beneficios que ofrece seguir una serie de procesos y métodos en el desarrollo de software.

Los aspectos relacionados con la ubicación de la asignatura **a)** en el marco del currículo formativo de las y los estudiantes, **b)** en el mapa de competencias profesionales en la industria de desarrollo del software, y **c)** en la evolución conceptual e histórica de la informática, serán introducidos en un apartado que, a modo de colofón, se les suministrará en las semanas finales del curso, en paralelo a sus esfuerzos por ultimar el Proyecto. A esta parte la hemos denominado Aspectos complementarios.

Es al final de la asignatura el momento donde pueden suscitar mas interés para proyectos futuros, tanto académicos como profesionales. Es profesor procederá a impartir (en principio fuera de la metodología PBL) los siguientes contenidos:

a) Reconocimiento de los beneficios de aplicar una disciplina al desarrollo del software
b) Reconocimiento de las etapas del ciclo de vida universalmente aceptadas como método de descomposición, incluyendo las fases que no hemos tenido tiempo de trabajar en detalle durante el curso.
c) Repaso de otras metodologías existentes y comparación crítica con el Proceso Unificado de Desarrollo del Software y del Lenguaje Unificado de Modelado.
d) Utilización real de las técnicas aprendidas en la industria del software.
e) Encuadre de la asignatura Ingeniería del Software en el mapa del Grado, destacando su relación con otras asignaturas posteriores de Especialidad.

Una de las ventajas del enfoque iterativo radica en que los alumnos consiguen obtener un producto software (inicialmente sencillo) mucho antes que si hubiesen seguido un proceso secuencial. Esto facilita la detección de errores en fases tempranas de la asignatura, ofrecen al estudiante una perspectiva general de cuáles son las consecuencias que tienen las decisiones tomadas en una fase (p.ej. en el diseño) en las fases posteriores (p.ej. en la implementación), de manera que en la siguiente iteración ya se tiene una idea del impacto de una decisión equivocada. Creemos que el desarrollo de temario de forma iterativa permitirá realizar un seguimiento adecuado del proceso de aprendizaje y dará ocasión a ofrecer retroalimentación de calidad en cada iteración.

Finalmente, pero no menos importante, nos gustaría resaltar que, siguiendo las pautas que nos recomienda la metodología PBL, los conocimientos a adquirir por los alumnos para desarrollar el Proyecto y correspondientes a los temas 1, 2 y 3 no serán expuestos previamente en el aula, sino que serán los propios alumnos los que deberán identificar cuáles son, dónde documentarse y cómo aprenderlos.

2.5. Carga de trabajo y duración del Proyecto

En las tablas 4, 5 y 6 se recogen la cargas horarias previstas para la asignatura de Ingeniería del Software según el proyecto docente descrito en este documento. Como es preceptivo, hemos separado las correspondientes a actividades encuadradas en la metodología PBL de aquellas cuya inclusión en dicha metodología sería inapropiada o dudosa.

PBL	Presenciales	No presenciales	Total	% Nota
Primera Iteración	15,5	20	35,5	15
Segunda Iteración	10,5	16	26,5	25
Tercera Iteración	6	30	36	35
Finalización del Proyecto	3	18	21	
TOTAL PBL	**35**	**84**	**119**	**75**

Tabla 4: Carga horaria para el alumno en actividades PBL y peso previsto en la evaluación.

No obstante, algunas de las actividades clasificadas como NO PBL pueden estar relacionadas con las PBL, sea porque sus premisas y/o resultados confluyen con el componente arterial de la asignatura, influyendo de manera decisiva en el desarrollo del Proyecto (dependencia de datos), sea porque tienen una correspondencia ajustada al discurso y prácticas de la PBL, pero con objetivos menos ambiciosos y con entregables que no pueden ser calificados de producto (dependencia conceptual).

NO PBL	Presenciales	No presenciales	Total	%Nota
Primera Iteración	7,5	7	14,5	10
Segunda Iteración	3	3	6	10
Tercera Iteración	0	0	0	0
Aspectos complementarios	4,5	6	10,5	5
TOTAL NO PBL	**15**	**16**	**31**	**25**

Tabla 5: Carga horaria para el alumno en actividades NO PBL y peso previsto en la evaluación.

El apartado **Finalización del Proyecto** se refiere a las actividades que, siendo en rigor parte de la Tercera Iteración, se solapan con la exposición final de **Aspectos Complementarios**. Esto se ha hecho buscando una mayor claridad en la división de los tiempos que se pueden asociar a cada fase. También incluye los tiempos correspondientes a la defensa del Proyecto (véase el capítulo 3).

Para elaborar la planificación hemos partido de 15 semanas lectivas con una dedicación semanal de 9,5 horas entre presenciales y no presenciales (véase la sección 1.1). Los módulos horarios de la Facultad están organizados sobre la base de una reserva máxima semanal de 4,5 horas presenciales por asignatura del Grado. Asimismo, la reglamentación de nuestro centro no permite utilizar la semana decimoquinta para sesiones presenciales obligatorias, lo que en nuestro caso no supone quebranto alguno, ya que las utilizaremos para las defensas de los Proyectos. Cada grupo dispondrá de media hora libremente acordada con el profesor para tal efecto, y la asistencia de otros alumnos como público será optativa.

	Presenciales	No presenciales	Total	%Nota
TOTAL(PBL + NO PBL)	**50**	**100**	**150**	**100**

Tabla 6: Carga horaria total para el alumno.

En nuestra propuesta hemos considerado 2 horas de trabajo no presencial por cada hora presencial, lo cual significa una carga presencial de 50 horas frente a una no presencial de 100. El uso de la relación 50/100 en lugar de la habitual 60/90 merece algunas precisiones.

Para empezar queremos volver a subrayar que la importancia del Proyecto en la evaluación de la asignatura (75%) apunta claramente a un desplazamiento de la carga de trabajo hacia bloques de tiempo fuera de la clase reglada. De este modo se prevé un mayor protagonismo del trabajo autoorganizado en la parte final del Proyecto (Tercera Iteración y Finalización de Proyecto). El organizar sesiones presenciales oficiales al ritmo normal del curso en esta etapa supondría forzar a los y las estudiantes a sumergirse en un trabajo complejo durante exactamente 1 hora y media y luego abandonarlo abruptamente, ya que se encontrarían entre dos sesiones de asignaturas diferentes. Es claramente preferible que dediquen sus esfuerzos al trabajo en grupo de sacar adelante las diferentes disciplinas del Proyecto en bloques horarios más grandes y con mayor libertad.

Siguiendo este principio algunas sesiones de las semanas 9, 10, 11 y 14 no tendrán lugar como tales porque el grupo no tiene en principio necesidad de las mismas. Por ello en la planificación no aparecen horas presenciales (véase el capítulo 4). Sin embargo queremos dar flexibilidad porque, si bien este sería el comportamiento por defecto, está previsto que se puedan convocar sesiones (no obligatorias) en forma de **workshops**, en las que se traten de manera específica líneas de actuación o dificultades encontradas en el Proyecto, todo ello bajo la dirección del profesor. El único requisito para esta convocatoria será que al menos dos grupos la soliciten, que describan con claridad las materias bajo discusión y, por supuesto, que asistan a la sesión.

Por todo lo mencionado, en la parte final de la asignatura está prevista una cierta permeabilidad entre horas no presenciales y presenciales, de manera que los grupos más necesitados de ello puedan aumentar la cantidad de las últimas en aras de tener una tutorización un poco más estrecha por parte del profesor y una mayor interacción positiva con compañeros de otros grupos.

Si en lugar de la distinción presencial/no presencial recabamos el dato con arreglo a la adecuación a la metodología objeto del presente proyecto docente, el total de horas dedicadas a actividades PBL resulta de 119, lo que supone un 79% del total. Hay una pequeña desviación de 4 puntos respecto a la evaluación (75%), lo cual es deseable ya que se prevé que las horas PBL tengan para el alumno un rendimiento algo inferior al menos en la Primera Iteración. Desde el punto de vista dinámico se observa que las actividades ajenas a la metodología PBL pierden protagonismo conforme avanza el curso: de un 29% en la Primera Iteración bajan al 18% en la Segunda Iteración hasta desaparecer por completo en la Tercera Iteración. La razón de que reaparezcan en forma de Aspectos Complementarios al final del cuatrimestre se ha explicado en la sección 2.4.

3. Metodología y Sistema de Evaluación

3.1. Cooperación y trabajo en equipo: el proceso SCRUM[2].

En el momento de diseñar una actividad de trabajo cooperativo nos encontramos muchas veces con que los estudiantes no cuentan con herramientas ni habilidades básicas para los mismos, empezando por el hábito de tomar decisiones y terminando por la forma de mantener una documentación efectiva que promueva el reparto de tareas. Adicionalmente deberíamos asegurar que el Proyecto desarrollado en equipo presta atención e incorpora los cinco ingredientes básicos del aprendizaje cooperativo:

1. Interdependencia positiva
2. Exigibilidad individual
3. Interacción cara a cara
4. Habilidades interpersonales y de trabajo en grupo
5. Reflexión del grupo

Para salvar al menos parcialmente este obstáculo hemos apreciado que el dotar a los alumnos de una metodología sencilla de entender y aplicar puede ser un impulsor crucial. Es por ello que les propondremos el uso de **SCRUM**. Esta metodología es ampliamente utilizada en metodologías ágiles de desarrollo del software[3]. por su naturaleza de proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar en grupo de manera colaborativa y obtener el mejor resultado posible de un proyecto.

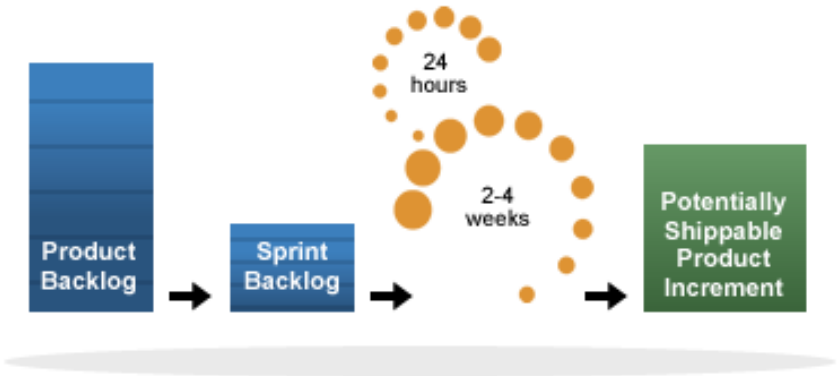


Figura 1: El proceso SCRUM (fuente http://pkab.wordpress.com/2008/07/11/scrum)

[2] El término procede del rugby y su significado es melée. En una melée si un jugador del equipo se cae, entonces también se cae todo el equipo.
[3] http://www.proyectosagiles.org/

La metodología está basada en roles, lo que supone una ventaja a la hora de dotar a los alumnos una manera natural de distribuir las responsabilidades y tareas:

1. El **Dueño del proyecto**. En un proyecto real sería el cliente, pero en nuestro caso tendrá que ser desempeñado por el profesor.
2. El *Scrum Master* o **Responsable del Proyecto**. En cada iteración sería un alumno distinto, lo cual garantiza que todos pasen alguna vez por este rol.
3. El *Scrum Team* o **Equipo del Proyecto**. Lo formarían los otros dos miembros del grupo .

En SCRUM un Proyecto se ejecuta en bloques temporales cortos y fijos llamados **sprints**. La noción de *sprint* es absolutamente análoga a la de iteración, y el nombre únicamente recalca la importancia del tiempo y la distancia fija a una meta. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

Las tres iteraciones de nuestro Proyecto se planificarán, de acuerdo con la metodología, mediante dos acciones principales:

1. **Selección de requisitos** (una sesión de clase). El cliente presenta al equipo la lista de requisitos más o menos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración. En nuestro caso los profesores en nuestro rol de clientes marcaremos las pautas de la selección de requisitos para cada iteración, sobre todo en la iteración inicial.
2. **Planificación de la iteración**. El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas. Esta planificación se recoge en un **backlog** del Proyecto. El volumen de trabajo obliga al grupo a repartirse el trabajo entre los tres. El profesor puede elegir al azar al alumno que debe explicar el estado del *backlog*, cómo se ha realizado el trabajo hasta un momento dado y cuáles son los próximos pasos. Cada uno de los alumnos se centra en su parte, pero no puede desentenderse de lo que hacen los compañeros, porque al final debe estar en condiciones de explicar el estado del desarrollo del Proyecto. De esta manera conseguimos garantizar la **interdependencia positiva** y la **exigibilidad individual**.

3. **Ejecución de la iteración**. Para ello SCRUM propone una reunión diaria de sincronización (15 minutos máximo). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. En la reunión cada miembro del equipo responde a tres preguntas:
    - ¿Qué he hecho desde la última reunión de sincronización?
    - ¿Qué voy a hacer a partir de este momento?
    - ¿Qué impedimentos tengo o voy a tener?

Figura 2:. Ejemplo de *backlog*

Somos conscientes de la dificultad para los alumnos de realizar una reunión diaria, y dada la dedicación real de los estudiantes a la asignatura debería ser suficiente con dos o tres reuniones semanales. Además el profesor puede fomentar en parte del horario de nuestras sesiones lectivas estas reuniones, lo que le permite monitorizar el proceso y exigir al grupo informes con las respuestas de cada miembro del grupo a las tres preguntas planteadas anteriormente. De esta forma fomentamos y garantizamos la **interacción cara a cara**, otro de los ingredientes básicos del trabajo cooperativo.

El *backlog* puede ser una estructura muy compleja que sirva como elemento de gestión global, pero en nuestro caso será suficiente con una lista de tareas de grano suficientemente fino como para describir de manera clara las responsabilidades y tareas del grupo y de cada uno de sus miembros . La lista surge durante la planificación de la iteración, las tareas son las que el equipo define como necesarias para conseguir el objetivo, identificándose el responsable de hacer el trabajo y estimándose el esfuerzo requerido para completarla. Por último, el *backlog* debe reflejar claramente el avance diario del Proyecto, así como los riesgos y problemas identificados sin exigir procesos complejos de gestión. Es también una herramienta de soporte para la comunicación

directa del equipo y promueve la **reflexión crítica del grupo**. El *backlog* debe también ser accesible para el profesor como herramienta fundamental de seguimiento del Proyecto. Para su soporte físico del hay varias posibilidades al alcance de los alumnos, lo cual facilitaría el acceso y modificación *online*:

- Hojas de cálculo compartidas en Google Docs, DropBox u otra aplicación en la nube.
- Pizarra física o pared (con pegatinas simulando las tareas)
- Aplicaciones de Gestión de Proyectos

En resumen SCRUM, por su propia naturaleza, fomenta el **compromiso** conjunto y la **colaboración** entre los miembros del equipo. La **transparencia** entre todos es fundamental para poder entender la situación real del Proyecto y así poder hacer las mejores adaptaciones que permitan conseguir el objetivo común. Por ello, su implantación como marco de trabajo en el desarrollo del Proyecto creemos que potencia decisivamente la adquisición de **habilidades interpersonales y de trabajo en grupo**.

3.2. Desarrollo de las iteraciones en el tiempo

Las tres iteraciones montadas sobre el mismo Proyecto no pueden ser tan breves como en el mundo real, dado que los alumnos no son trabajadores a tiempo completo de la asignatura, y serán de 5, 3 y 4+2 semanas respectivamente. Presentado el Proyecto al inicio del curso, en cada iteración, se irán planteando tareas de complejidad creciente, trabajándose en cada una de ellas los aspectos de cada tema que sean necesarios para la consecución de esa tarea (uno de los principios de la metodología PBL) de manera que el resultado enriquezca la versión anterior del producto, tanto en funcionalidad como en calidad. Las dos últimas semanas serán especiales puesto que servirán al alumno o alumna para completar los flecos de la Tercera Iteración, y al profesor para introducir otros aspectos que no entran de manera natural en el marco del Proyecto y que hemos agrupado en el nuevo tema final denominado Aspectos Complementarios. Esta recolocación al final del cuatrimestre de las cuestiones de contextualización de la asignatura (origen, motivación e historia de la Ingeniería del Software, su papel dentro del Grado según especialidades, su importancia y valoración en el entorno profesional, etc.) tiene también efectos colaterales positivos. Esperamos que, de este modo, el alumno o alumna reciba la información en un momento en el que su madurez le permita otorgarle significación y provecho. Según nuestra experiencia, esto rara vez ocurre cuando se le

proporciona a modo introductorio en un momento en que su desubicación con respecto a los objetivos del curso es máxima.

La forma en que quedan configurados los diferentes elementos constructores del Proyecto docente se describe en la figura 3, en la que se detallan todos los elementos constructores del proyecto docente que han sido descritos anteriormente, como Temas, Objetivos de Aprendizaje (**OA1-5**) e Iteraciones. Asimismo se introducen las Actividades ordinarias (**A1-8**), las Actividades de desarrollo del Proyecto (**P0-3**) y las Actividades de Laboratorio (**L1-6**) que han de ser descritas en el resto del presente capítulo.

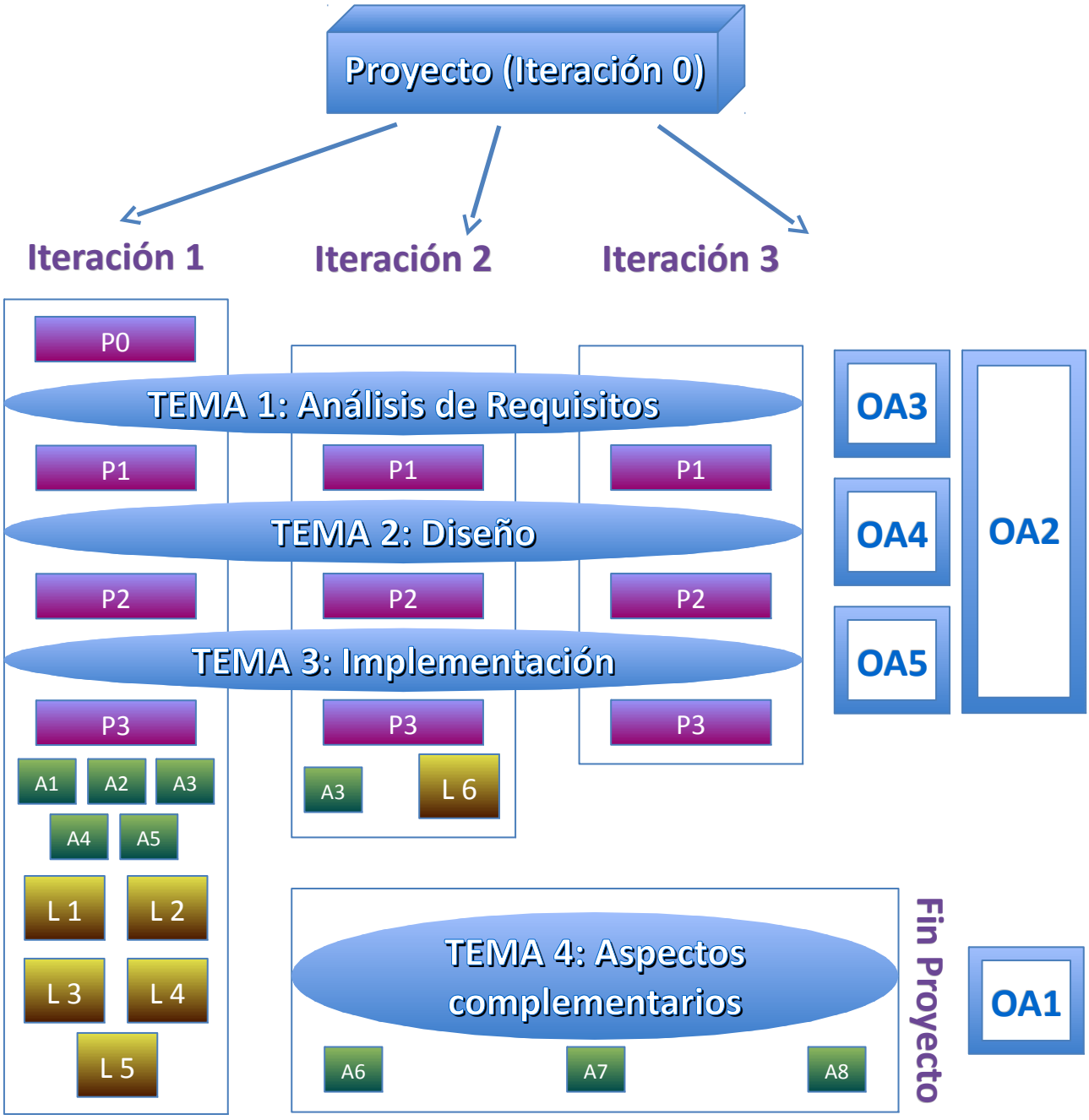


Figura 3: Organización del curso en torno a la metodología PBL

3.3. Actividades

Para diseñar el presente proyecto docente hemos organizado y/o recogido una serie de actividades didácticas orientadas a la consecución de los objetivos de aprendizaje indicados en la sección 1.3, que quedarán recogidas mediante tablas.

3.3.1. Tipos de evaluación

Todas las actividades descritas son obligatorias, y por tanto conllevarán algún tipo de evaluación. Sin embargo, para evitar una excesiva rigidez y un número demasiado alto de notas a integrar hemos considerado dos modalidades principales y una variante adicional:

1. **Filtro.** La valoración de la actividad se calificará como APTA o no APTA, pero no tendrá capacidad aditiva sobre la calificación final.
2. **Punto Fijo**: La actividad tendrá un peso concreto en la calificación final.
3. **Extra Bonus**: Independientemente de que la actividad se evalúe según alguno de los esquemas anteriores, se premiarán las actividades que de alguna forma sean sobresalientes con puntos adicionales sobre la calificación final.

En las tablas descriptivas de las actividades se indicará en cuál de las primeras dos modalidades de evaluación se encuadra cada una.

3.3.2. Tipos de tarea

Entre un catálogo inicial de 25 tipos de tarea consideradas hemos decidido incluir en nuestra asignatura 10. Téngase en cuenta que una misma actividad puede incluir varias tareas, con sus respectivos tipos:

1. Actividades cooperativas: Puzle, póster…
2. Breve exposición de un tema por parte de un grupo al resto de la clase
3. Lluvia de ideas (brainstorming)
4. Entrevista a expertos
5. Discusión dirigida a través de preguntas que se plantean
6. Búsqueda de información y puesta en común en el grupo
7. Elaboración colaborativa de fuentes de información y recursos
8. Presentación del Proyecto
9. Cuestionarios de autoevaluación

10. Laboratorios cerrados dirigidos.
11. Análisis de escenario.
12. Uso de las preguntas-guía para dirigir la discusión (apartados 2.3.3 y 2.3.4).
13. Contraste por pares.

Las tablas que describen cada una de las actividades concretas propuestas se identifican mediante **TT-n** (n=1..13) los tipos de tareas incluidos en cada una.

3.3.3. Actividades de Desarrollo del Proyecto

El *enunciado inicial del Proyecto* (desarrollo de una aplicación de software que permiten elección y alquiler remotos de casas rurales y la gestión de las mismas por sus propietarios) se presentará una vez formados los grupos tras la actividad individual inicial. Junto con el enunciado posiblemente se les entregarán artefactos correspondientes a una iteración ya desarrollada del Proyecto (**Iteración 0**).

La primera actividad de desarrollo será **P0**, y en la misma se planteará una reflexión por parte de los grupos acerca de los requisitos del Proyecto, teniendo en cuenta no sólo las especificaciones y artefactos recibidos sino también algunos ejemplos reales.

El resto de Actividades de desarrollo del Proyecto (**P1-3**) serán recurrentes y deberán revisarse y completarse en cada una de las tres iteraciones. Para ello tendrán que buscar documentación e inspirarse en el material que se les ha entregado para conocer y aplicar los aspectos de cada fase, comenzando por los más básicos y profundizando a medida que se complican las características del producto a desarrollar.

Dado que al final de cada iteración cada grupo debe tener una aplicación operativa, aunque quizás incompleta, debería ser posible hacer una presentación de control ante el cliente. Sin embargo esto puede suponer un indeseado cuello de botella si el profesor tiene que revisar en tiempo real todos los trabajos. La solución será introducir una sesión de contraste entre pares, a realizar en el laboratorio al final de las iteraciones 1 y 2. Cada grupo tendrá como tarea asistir a la defensa del Proyecto de otro grupo y detectar posibles errores, inconsistencias o fallos de funcionamiento. Como conclusión debe escribir un **Informe de Inspección** que se entregará al profesor e incorporará al *backlog* del grupo auditado, junto con una versión "congelada" del Proyecto.

Este contraste será una tarea de tipo FILTRO, pero podrá tener consecuencias posteriores. Si un fallo grave es detectado e incluido en el Informe pero no es corregido por el grupo inspeccionado, este sufrirá una penalización adicional que en casos extremos

puede implicar la anulación del Proyecto. Pero si un fallo grave no es detectado y se propaga a iteraciones posteriores la responsabilidad será compartida entre el grupo inspeccionado y el inspector.

La inspección final será la Defensa del Proyecto y se efectuará por el profesor al terminar la tercera iteración. Esta tendrá forma de exposición pública ante la clase.

Tras la inspección de cada iteración el grupo deberá incorporar al *backlog* una **Memoria** correspondiente a las tres fases del Proyecto estructuradas en las correspondientes actividades directoras del mismo. Así el informe de **P1** corresponderá a las tareas y artefactos relacionados con el Análisis de Requisitos (Tema 1), **P2** con los del Diseño (Tema 2) y, finalmente, **P3** con los de la implementación de cada solución (Tema 3).

Finalmente, en las últimas semanas del cuatrimestre se realizará un control para confirmar el nivel de exigibilidad individual adecuado en el desarrollo del Proyecto (lo denominamos **Control Anti-Doping**).

A continuación se detalla la naturaleza de las actividades **P** que hemos englobado en la categoría "de desarrollo del Proyecto" y que constituyen el eje de la actividades de aprendizaje en la asignatura. Tanto estas como las descritas en apartados posteriores se describirán utilizando la siguiente ficha de modelo:

<table>
<tr><td>ACTIVIDAD:
CODIGO-Nombre de la actividad</td><td>OBJ. APRENDIZAJE:
Los que se trabajan en la actividad.</td><td>TIPO TAREAS:
Los utilizados para la consecución de los objetivos (apartado 3.3.2)</td><td>DEDICACION:
Presencial:
No presencial:</td><td>TIPO EVALUACION:
FILTRO o PUNTO FIJO. El EXTRA-BONUS es siempre posible</td></tr>
<tr><td colspan="5">DESCRIPCIÓN:
Planteamiento de la actividad. Objetivos explícitos para los estudiantes. Modalidad (individual o en grupo).</td></tr>
<tr><td colspan="5">ENTREGABLES:
Cuáles son los materiales que se deben entregar como consecución de la actividad.</td></tr>
<tr><td colspan="5">EVALUACION:
Aspectos a considerar y criterios de validación de que se han cumplido los objetivos de aprendizaje.</td></tr>
<tr><td colspan="3">ELEMENTOS DE COOPERACION:
Ingredientes para el aprendizaje cooperativo considerados como más adecuados para la consecución de la actividad y cómo se han utilizado.</td><td colspan="2">RECURSOS:
Qué se le ofrece al alumno para la realización de la actividad.</td></tr>
</table>

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
P0- Identificación de requisitos del Proyecto	OA3	TT-11, TT-6, TT-3 y TT-1	Presencial: 1,5 No presencial: 6	PUNTO FIJO

DESCRIPCIÓN:

El objetivo de esta actividad consiste en identificar los requisitos del sistema. Es la tarea que inicia el trabajo en grupo y debe insistirse en los aspectos cooperativos de la misma ante los estudiantes.

Para este objetivo se presentará la pregunta motriz y se indicarán cuáles son los primeros pasos a realizar. Al menos se deberán estudiar 3 sistemas en profundidad, realizando posteriormente una comparativa entre las funcionalidades de los sistemas e indicando cuáles son las que consideran fundamentales, y cuales pueden considerarse aspectos de mejora. Cabe destacar que una de las condiciones del Proyecto consiste en que el sistema propuesto debe superar a los sistemas actualmente existentes en alguna característica, por tanto, deberán proponer una funcionalidad no existente. Como consecuencia de la búsqueda de nuevas funcionalidad por parte de los grupos, se obtendrá una lista de nuevas funcionalidades a añadir al sistema de casas rurales, que se debatirá en clase, y donde se votarán para obtener las más innovadoras.

ENTREGABLE:

Cada grupo realizará una tabla comparativa con las funcionalidades ofrecidas por cada plataforma y una exposición oral, donde cualquiera de los miembros pueda defender el Proyecto. Esta tabla deberá presentar al menos alguna funcionalidad no contemplada en ninguna de las plataformas existentes. Se empezará a construir el *backlog* del Proyecto.

EVALUACIÓN:

Se tendrán en cuenta sobre todo la relevancia de los aspectos estudiados desde el punto de vista de un hipotético cliente de la aplicación (propietario o locatario).

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Interdependencia positiva: Cada alumno debe estudiar el profundidad un sitio web. Exigibilidad individual: Todo alumno tiene que ser capaz de defender el Proyecto (Método del Representante).	Enunciado del problema y lista de webs de referencia. Ejercicios anteriores resueltos de manera colaborativa. Manual de SCRUM

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
P1- Captura de requisitos del Proyecto	OA2 y OA3	TT-12, TT-1, TT-3, TT-11, TT-6 y TT-13	Presencial: 9 No presencial: 12	PUNTO FIJO

DESCRIPCIÓN:

Una vez identificados los requisitos del sistema en la actividad P0, se procederá a su modelado en UML (casos de uso, modelo del dominio y flujos de eventos). Este proceso se realizará en 3 iteraciones incrementales. En la primera iteración habrá que modelar los requisitos de un conjunto básico de las funcionalidades obtenidas en la actividad P0. En la siguientes dos iteraciones, se realizará el modelado de requisitos de un conjunto de casos de uso que se se irán seleccionando de las funcionalidades obtenidas en la actividad P0, en base a su creciente complejidad.

Los alumnos deben hacer utilizar las preguntas-guía para averiguar las técnicas adecuadas para cumplir la tarea. Se profundizará en la metodología SCRUM.

Cada grupo deberá defender su Proyecto ante otro, que hará de Inspector (iteraciones 1 y 2) y ante el profesor (iteración 3). Se debe evitar que el grupo inspector de las iteraciones 1 y 2 sea el mismo.

ENTREGABLE:

Los grupos deberán realizar el modelado de requisitos de las funcionalidades establecidas en cada iteración, utilizando alguna herramienta de modelado. Todos los miembros del grupo deberán ser capaces de defender el modelo de requisitos propuesto.

Cada grupo deberá mantener una Memoria de la fase de Captura de Requisitos que se irá enriqueciendo en las sucesivas iteraciones.

Cada grupo, en su rol de inspector, deberá escribir un Informe de Inspección que se incorporará al *backlog* del grupo inspeccionado.

EVALUACIÓN:

Se valorará la comprensión de los artefactos propios del Modelo de Casos de Uso y del Modelo de Dominio. El seguimiento del *backlog* será el elemento evaluador fundamental. El Informe de Inspección será tenido en cuenta tanto para el grupo inspector como para el inspeccionado. La Defensa Final tendrá una valoración específica.

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Exigibilidad individual: Todo alumno tiene que ser capaz de defender el Proyecto (Método del Representante). Contraste por pares.	Requisitos del sistema (Entregable actividad P0). Manual de SCRUM. Lista de preguntas-guía. Guión para la Inspección.

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
P2- Diseño del Proyecto	OA2 y OA4	TT-12, TT-1, TT-3, TT-6 y TT-13	Presencial: 12 No presencial: 18	PUNTO FIJO

DESCRIPCIÓN:

Esta actividad se realizará 3 veces a los largo del Proyecto. Una vez validados y establecidos los requisitos del sistema para cada iteración, el objetivo de esta actividad consiste en realizar el diseño de la aplicación. Esto conlleva diseñar los diagramas de secuencia de los casos de uso, el modelo del diseño y la arquitectura en al menos 3 niveles. Cada alumno se encargará de realizar alguno de los diagramas de secuencia. El modelo del diseño se realizará de forma colaborativa recogiendo todas las necesidades obtenidas en los diferentes diagramas.

Si no hay ningún contratiempo los grupos los conformarán los mismos miembros que en la fase de requisitos. Los alumnos deben hacer utilizar las preguntas-guía para averiguar las técnicas adecuadas para cumplir la tarea. Se profundizará en la metodología SCRUM.

Cada grupo deberá defender su Proyecto ante otro, que hará de Inspector (iteraciones 1 y 2) y ante el profesor (iteración 3). Se debe evitar que el grupo inspector de las iteraciones 1 y 2 sea el mismo.

ENTREGABLE:

Los grupos deberán realizar el diseño del sistema utilizando la herramienta de modelado utilizada en la fase de requisitos y una exposición oral, donde cualquiera de los miembros deberá ser capaz de defender el Proyecto. Se pedirá una memoria con los diagramas y las decisiones de diseño tomadas.

Cada grupo deberá mantener una Memoria de la fase de Diseño que se irá enriqueciendo en las sucesivas iteraciones.

Cada grupo, en su rol de inspector, deberá escribir un Informe de Inspección que se incorporará al *backlog* del grupo inspeccionado.

EVALUACIÓN:

Se valorará la comprensión de los artefactos propios del Modelo de Diseño. El seguimiento del *backlog* será un elemento evaluador fundamental. El Informe de Inspección será tenido en cuenta tanto para el grupo inspector como para el inspeccionado. La Defensa Final tendrá una valoración específica.

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Interdependencia positiva: El modelo del diseño se realiza de forma colaborativa a partir del trabajo realizado de manera individual. Exigibilidad individual: Todo alumno tiene que ser capaz de defender el diseño del Proyecto (Método del Representante). Contraste por pares.	Modelo de captura de requisitos (Entregable actividad P1). Manual de SCRUM. Lista de preguntas-guía. Guión para la Inspección.

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
P3- Implementación del Proyecto	OA2 y OA5	TT-12, TT-1, TT-3 y TT-6	Presencial:9 No presencial: 24	PUNTO FIJO

DESCRIPCIÓN:

Esta actividad se realizará 3 veces a los largo del Proyecto. Una vez validado el diseño del sistema de cada iteración, el objetivo de esta fase consiste en realizar su implementación.

Cada alumno deberá al menos implementar un caso de uso completo (interfaz, lógica de negocio y acceso a datos). Adicionalmente cada alumno será responsable de una parte de aplicación: instalación y despliegue(RMI), lógica de negocio y BD.

Si no hay ningún contratiempo los grupos los conformarán los mismos miembros que en la fase de requisitos y análisis del sistema. Los alumnos deben hacer utilizar las preguntas-guía para averiguar las técnicas adecuadas para cumplir la tarea. Se profundizará en la metodología SCRUM.

Cada grupo deberá defender su Proyecto ante otro, que hará de Inspector (iteraciones 1 y 2) y ante el profesor (iteración 3). Se debe evitar que el grupo inspector de las iteraciones 1 y 2 sea el mismo.

ENTREGABLE:

Los grupos deberán realizar la implementación de las casos de uso diseñados en cada iteración. Cualquiera de los miembros deberá ser capaz de defender el Proyecto. Se pedirá una memoria explicando cuáles han sido las decisiones de implementación.

Cada grupo deberá mantener una Memoria de la fase de Implementación que se irá enriqueciendo en las sucesivas iteraciones.

Cada grupo, en su rol de inspector, deberá escribir un Informe de Inspección que se incorporará al *backlog* del grupo inspeccionado.

EVALUACIÓN:

La aplicación debe estar en funcionamiento. En la práctica este requisito opera a nivel global sobre la calificación, ya que si la aplicación ni siquiera funciona no será posible obtener el 30% de esta actividad. El seguimiento del *backlog* será un elemento evaluador fundamental.

El Informe de Inspección será tenido en cuenta tanto para el grupo inspector como para el inspeccionado. La Defensa Final tendrá una valoración específica.

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Interdependencia positiva: Cada alumno es responsable de mantener su parte operativa. Exigibilidad individual: Desarrollado de un caso de uso en su totalidad.	Modelos del diseño (Entregable actividad P2). Manual de SCRUM. Lista de preguntas-guía. Guión para la Inspección.

3.3.4. Actividades de Laboratorio

Los laboratorios suponen la oportunidad de ejercitar competencias con un fuerte componente técnico. Se han diseñado de forma que con cada uno de ellos se puedan resolver de cuatro a seis tareas de complejidad creciente. Las actividades **L** o de laboratorio están especialmente indicados para familiarizarse con una técnica o herramienta que suponga una clara novedad para el estudiante.

<table>
<tr><td>ACTIVIDAD:
L1-L6 Laboratorios</td><td>OBJ. APRENDIZAJE:
OA2, OA3, OA4 y OA5</td><td>TIPO TAREAS:
TT-9 y TT-10</td><td>DEDICACIÓN:
Presencial:7,5
No presencial: 7</td><td>TIPO EVALUACIÓN:
FILTRO y PUNTO FIJO</td></tr>
<tr><td colspan="5">DESCRIPCIÓN:
Para comprender correctamente para qué sirven y cómo se utilizan las herramientas utilizadas en la asignatura es necesario que los alumnos realicen unos laboratorios prácticos guiados que les ayuden a aprender su manejo. Como todos esos laboratorios tienen unos objetivos definidos, al final de cada uno de ellos los alumnos tendrán que responder a un cuestionario para demostrar que han aprovechado el laboratorio y enviar el código (o informe automático) que demuestre que los han realizado. Los laboratorios serán los siguientes:
L1. Arquitecturas software en varios niveles: separación entre la presentación y la lógica del negocio (OA4)
L2. Uso de AWT/SWING para implementar las interfaces de usuario (nivel de presentación) (OA5)
L3. Uso de StarUML para definir los requisitos (OA3)
L4. Uso del sistema de gestión de bases de datos db4o para implementar el nivel de datos (OA5)
L5. Uso de StarUML para realizar el diseño (OA4)
L6. Uso de Java RMI para implementar y acceder de manera remota a la lógica del negocio (OA5)</td></tr>
<tr><td colspan="5">ENTREGABLE:
Cuestionario y resultados del laboratorio.</td></tr>
<tr><td colspan="5">EVALUACIÓN:
Cuestionario de evaluación (100%) y corrección del código (apto/no apto)</td></tr>
<tr><td colspan="3">ELEMENTOS DE COOPERACIÓN:
Reflexión grupal. Interdependencia positiva.</td><td colspan="2">RECURSOS:
Enunciados de los laboratorios. Herramientas, manuales y referencias adicionales.</td></tr>
</table>

3.3.5. Actividades ordinarias

Estas actividades no están directamente ligadas con el Proyecto. En unos casos buscan reforzar aspectos de la asignatura en los que los grupos pueden encontrar dificultades especiales. Es el caso de las de la Primera Iteración, que cuenta con la mayoría de ellas. También tienen una función de generar dinamismo en las clases y abrir nuevas líneas de pensamiento en los grupos.

En la Segunda Iteración el número de actividades disminuye, ya que la mayor parte de las técnicas han sido presentadas en la primera parte del curso y los grupos van adquiriendo más autonomía en la toma de decisiones. De hecho en este punto del cuatrimestre sólo se propone la actividad **A3** (que en realidad puede y debe ejercerse en cualquier momento en el que se detecte que los estudiantes cometen errores más o menos generalizados) . Es este mismo motivo el que provoca que solamente se celebre un laboratorio (el **L6**) para introducir RMI, la tecnología que necesitarán para el acceso remoto a su aplicación.

Al llegar la Tercera Iteración los alumnos tendrán ya cierta destreza en el desarrollo de software y conocerán cuáles son las implicaciones que tienen los modelos de requisitos y diseño en la aplicación final. Este última iteración es la más abierta, dejando la mayoría de las mejoras a criterio de la creatividad de cada grupo. En esta fase no se proponen actividades específicas, ya que el objetivo es responsabilizar al estudiante en la planificación y desarrollo del Proyecto.

Al final de la mencionada Tercera Iteración (en la fase que hemos denominado Finalización del Proyecto) se vuelve a proponer una serie de actividades que ni están englobadas en la metodología PBL ni tienen relación alguna con el Proyecto (**A6-8**), pero que sí consideramos importantes y complementan adecuadamente la formación que deben adquirir los estudiantes en la asignatura Ingeniería del Software I.

La descripción de las actividades ordinarias (o actividades **A**) se ha realizado utilizando tablas que siguen la misma plantilla que las más específicas.

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A1- Panel del Modelo de Referencia	OA2	TT-1	Presencial:3 horas (dos sesiones) No presencial: 3 horas	FILTRO

DESCRIPCIÓN:

Los alumnos operarán en grupos de 3 para construir un panel (póster) en el que reflejen todas las ideas fundamentales asociadas al Proceso Unificado de Desarrollo del Software. En principio se sugerirá que el panel se estructure por fases o por disciplinas. su finalización debe ser previa al comienzo de la fase de implementación del Proyecto. Lo más probable es que no dispongamos de un entorno de exposición permanente por lo que su tamaño y portabilidad deben ser requisitos iniciales. Cada dos semanas dedicaremos una parte de una clase para revisar el avance de los paneles.

ENTREGABLE:

Póster e informe sobre su realización

EVALUACIÓN:

Se repartirá una rúbrica con los aspectos a valorar, en los que no debe faltar la legibilidad, la organización y la aportación de ideas esenciales para comprender mejor la UP

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Método del representante. Es un buen candidato para iniciar el método, ya que transmite el mensaje de seriedad con bajo coste y, sobre todo, con riesgo limitado	Ninguno

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A2 - El cliente siempre tiene razón	OA3	TT-3 y TT-4	Presencial:1.5 No presencial: 2	FILTRO

DESCRIPCIÓN:

Extraer los requisitos de los clientes para conseguir el producto software que sea exactamente el que necesitan dichos clientes, esto es, que resuelva de manera eficaz y eficiente sus problemas no es tarea fácil. Muchas veces los clientes no saben realmente lo que quieren, o no saben expresarlo correctamente (o los desarrolladores son incapaces de entenderlo fácilmente), o suponen que es algo tan obvio que no necesitan ni decírselo a los desarrolladores, o desconocen que determinada tarea o proceso que realizan de una manera podría realizarse de otra manera utilizando un sistema informático. En esta actividad, el profesor asumiría el rol de un cliente interesado en comprar un software a desarrollar por los alumnos, los cuales tendrían que extraer los requisitos. Para ello, se organizaría una sesión en donde los alumnos plantearan preguntas concretas al cliente acerca de sus necesidades, o peticiones de documentos, impresos, gráficos, etc. utilizados por el cliente. El cliente intentaría pedir siempre lo máximo intentando que el coste fuera el mínimo. No sería un experto en informática por lo que no respondería a preguntas técnicas, pero sí sería un usuario habitual de informática por lo que siempre mantendría que todo lo que pide es fácil de hacer porque ha visto cosas similares funcionando. Tampoco facilitaría información que no le pidieran explícitamente.

ENTREGABLE:

Los alumnos tendrían que elaborar el MCU, el MD y los flujos de eventos del sistema.

EVALUACIÓN:

Participación y relevancia de las preguntas (50%) y corrección de la documentación entregada (50%)

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Reflexión grupal. Interdependencia positiva	Se puede recomendar un guión on-line.

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A3- Discusión dirigida de soluciones propuestas por los alumnos	OA3,OA4 y OA5	TT-5	Presencial:3 (dos sesiones)	FILTRO

DESCRIPCIÓN:

En clase se proponen diferentes ejercicios sobre temas concretos (captura requisitos, elaboración de diagramas de secuencia, implementación de métodos). Estos ejercicios se recogen al final de la clase. Se pretende que obtengan soluciones en grupos de 2/3 alumnos. El profesor analiza las soluciones y algunas de ellas se presentan en una sesión de aula a todo el grupo. Se establece un debate sobre la corrección de las soluciones elaboradas por el alumno con la orientación del profesor. El objetivo es, de forma colaborativa, llegar a una solución discutida y consensuada por la mayoría. Es de especial relevancia esta actividad en esta asignatura debido a que para un mismo enunciado puede haber varias soluciones válidas.

ENTREGABLE:

El entregable intermedio de la actividad son las soluciones previas.

EVALUACIÓN:

Se pueden evaluar las soluciones previas dadas por los alumnos, pero el objetivo fundamental es la participación en la fase de discusión.

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Reflexión grupal. Interdependencia positiva	Al final de la actividad se dá a los alumnos una solución válida al ejercicio objeto de la discusión.

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A4- Juego de los objetos animados	OA4	TT-3 y TT-1	Presencial: 0,75	FILTRO

DESCRIPCIÓN:

Los alumnos funcionarán en grupos de cuatro elegidos al azar. El profesor planteará un caso de uso en el que el diseño orientado a objetos sea relevante y lo dejará expuesto de forma que sea visible para toda la clase. Cada alumno desempeñará el papel de un objeto (del dominio o de diseño) y tendrá una cartulina donde resulte visible para el resto del grupo. El profesor lanzará el mensaje de inicio sobre el alumno InterfazDeUsuario, que tendrá que escribir en su cartulina el método correspondiente. A partir de aquí cada objeto que recibe un mensaje deberá decidir a qué objeto le envía el siguiente y con qué parámetros. Una vez terminada esta fase se les indicará que deben sistematizar lo aprendido en forma de diagrama de secuencia. El alumno InterfazDeUsuario será responsable de reflejar por escrito los resultados a los que el grupo llegue, y el mismo tiene que ser coherente con lo que figura en las cartulinas. Opcionalmente se les puede pedir que indiquen los patrones de diseño aplicados. Finalmente se hará una puesta en común de los resultados. El profesor desplegará una solución del diagrama de secuencia en la pizarra en función de lo que hayan hecho los grupos.

ENTREGABLE:

Informe escrito durante la actividad

EVALUACIÓN:

Requisitos mínimos: Cumplimiento de los aspectos formales.

Extra Bonus: Se valorará la participación en la puesta en común.

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Es conveniente que los integrantes de un grupo no estén lo suficientemente cerca como para poder escribir en las cartulinas de los compañeros.	Enunciado del problema, prontuario con los patrones de diseño.

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A5 -Puzle de patrones GRASP	OA4	TT-1	Presencial: 0,75 No presencial: 1.5	FILTRO

DESCRIPCIÓN:

Los patrones de diseño GRASP "Alta cohesión" y "Bajo acoplamiento" indican cómo repartir las responsabilidades entre las clases de diseño de manera que se generen diseños software fáciles de gestionar y evolucionar. El objetivo de esta actividad es que los alumnos conozcan ambos patrones y sepan utilizarlos en cada caso (previamente habrán estudiado los tres patrones principales: "Creador", "Experto" y "Controlador").

Inicialmente se plantea un problema de diseño y se pregunta a los alumnos cuál es la solución. Para responder al problema se crean grupos de 2 personas, donde cada uno lee/entiende un documento acerca de cada patrón. A continuación se realizan reuniones de expertos de 3 miembros donde aclaran las dudas respecto a cada patrón.

Finalmente cada miembro de cada grupo explica a su compañero el patrón estudiado. El alumno puede tomar notas en todo momento.

ENTREGABLE:

Un test sobre ambos patrones con los apuntes recogidos. Una solución por grupo al problema inicialmente planteado.

EVALUACIÓN:

Test (50%). Solución común(50%).

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Interdependicia positiva: cada alumno debe explicar a su compañero el tema estudiado. Exigibilidad individual: Test individual.	Descripción de los patrones "Alta cohesión" y "Bajo acomplamiento".

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A6 - Concurso de desastres informáticos.	OA1	TT-6 y TT-2	Presencial:1.5 No presencial:6	PUNTO FIJO

DESCRIPCIÓN:

Los alumnos trabajarán individualmente o en parejas. El máximo de trabajos diferentes no puede ser mayor de 15. Cada pareja debe buscar y elegir un desastre informático documentado en el que se evidencien las consecuencias acarreadas por defectos en la calidad del software.

ENTREGABLE:

Los alumnos deberán realizar una presentación de una única transparencia exponiendo el error e indicando cuáles considera que han sido las causas por las que se ha producido y las consecuencias que ha acarreado.

EVALUACIÓN:

Requisitos mínimos: El incidente elegido debe ser verificable y corresponder a un error de software. Se deben respetar el formato y el tiempo asignados. Se debe haber entendido correctamente el incidente y ser capaz de responder a dudas planteadas por profesores y compañeros.

Concurso: Los propios alumnos valorarán en votación los trabajos de los compañeros según las siguientes categorías: a) qué incidente es más grave, b) qué incidente ha sido mejor explicado y c) qué exposición ha sido más ilustrativa (es decir, cuál les ha hecho aprender más conceptos nuevos). El 20% de trabajos con más votos en cada categoría recibirá Extra Bonus. No entrarán en concurso incidentes de más de 20 años de antigüedad.

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Interdependencia positiva: Cada alumno presenta un tema y lo presenta a los otros dos. Exigibilidad individual: uno de los tres presenta el trabajo.	Ninguno.

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A7- Presentación de aspectos contextuales	OA1	TT-6,TT-7,TT-8	Presencial:1.5 No presencial: 6	PUNTO FIJO

DESCRIPCIÓN:

La ingeniería del software abarca multitud de áreas de aplicación. En objetivo de esta actividad consiste en familiarizar al alumno con las áreas relacionadas con la ingeniería del software. El alumno deberá escoger unos de los siguientes temas:

Crisis del software

Modelos de referencia o ciclos de vida de los procesos de desarrollo de software. (tipos de ciclos de vida, fases, etc.)

Herramientas de ayuda para las diferentes fases del desarrollo de software

Calidad de software

Gestión de Proyectos Informáticos.

Introducción al Proceso Unificado de Desarrollo de Software

ENTREGABLE:

El alumno deberá realizar un informe descriptivo del tema elegido (3/4 caras máximo). En el informe se deberán incluir las fuentes consultadas (al menos tres) y las horas de dedicación al trabajo. En grupos de 3 alumnos prepararán una presentación PowerPoint o similar (6-8 diapositivas), de los 6 temas que uno de los miembros expondrá delante de sus compañeros de clase

EVALUACIÓN:

Trabajo escrito individual (50%). Presentación compartida(50%).

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Interdependencia positiva: Cada alumno presenta un tema y lo presenta a los otros dos. Exigibilidad individual: uno de los tres presenta el trabajo.	Ninguno

ACTIVIDAD:	OBJ. APRENDIZAJE:	TIPO TAREAS:	DEDICACIÓN:	TIPO EVALUACIÓN:
A8-Vendedores del Rational Unified Process de IBM.	OA2	TT-7 y TT-2	Presencial:1.5 No presencial: 6	PUNTO FIJO

DESCRIPCIÓN:

Existen diferentes métodos de desarrollo de software que han sido utilizados con más o menos éxito. Uno de ellos (de los de más éxito, se entiende) es el elemento vehicular de nuestra asignatura: el Proceso Unificado de desarrollo de Software o simplemente el Proceso Unificado (Unified Process en inglés). Una implementación específica de dicho proceso es la realizada por la casa Rational Software, absorbida actualmente como división de IBM.

En esta actividad se trata de realizar presentaciones comerciales donde vendedores del producto traten de convencer a posibles clientes (desarrolladores de software, directores de proyectos informáticos, financieros,…) de las bondades del producto. La actitud de estos últimos tiene que ser crítica con los primeros por diferentes motivos: desarrolladores que ya utilizan un método de desarrollo y son reacios a cambiar, directores de proyectos informáticos que no están seguros de que se vaya a ganar en productividad, financieros que no desean invertir en un nuevo producto porque no ven claro el retorno de la inversión,…

ENTREGABLE:

Los alumnos (vendedores) deberán realizar una presentación y contestar/rebatir todas las objeciones que les plantean los alumnos (desarrolladores, directores, financieros,…)

EVALUACIÓN:

Presentación y defensa oral.

ELEMENTOS DE COOPERACIÓN:	RECURSOS:
Interdependencia positiva: Cada alumno presenta un tema y lo presenta a los otros dos. Exigibilidad individual: uno de los miembros del grupo presenta el trabajo.	Documentación inicial sobre el RUP (de uso no obligatorio).

3.4. Calendario de entregas

		ITERACIÓN 1	ITERACIÓN 2	ITERACIÓN 3
Actividad	**Semana/s**	**ENTREGABLES**		
P0	**1**	**Tabla de requisitos identificados para el proyecto**		
P1	**2,6,10**	**Memoria de requisitos validados por el cliente**		
P2	**4,7,11**	**Memoria de Diseño del Sistema**		
P3	**5,8,12**	**Implementación del Sistema**		
P1, P2 y P3	**2 – 12**	***Backlog* del Proyecto**		
P1 y P2	**4 y 7**	**Informe de Inspección de Proyecto A de otro grupo**	**Informe de Inspección de Proyecto B de otro grupo**	
P3	**14**			**Control Antidoping, Informe Final y Defensa del Proyecto**
A1	**1**	**Informe escrito sobre las cuestiones del panel**		
A2	**2**			
A3	**3**			
A4	**4**	**Informe escrito durante la actividad**		
A5	**4**	**Test a cumplimentar durante la actividad**		
A6	**13**			**Informe Escrito de la Actividad Desarrollada (Sólo debe realizar una de las tres propuestas esa semana)**
A7				
A8				
L1	**2**	**Código del problema resuelto**		
L2	**2**	**Código del problema resuelto**		
L3	**3**	**Fichero StarUML**		
L4	**3**	**Código del problema resuelto**		
L5	**4**	**Fichero StarUML**		
L6	**8**		**Código del problema resuelto**	

Tabla 7: Entregables del Proyecto y del resto de actividades con sus fechas de entrega

En la sección anterior, dentro de las tablas que describen cada actividad ,se ha incluido una celda para sus entregables asociados. A modo de resumen y con el objetivo de clarificar algunas de estas actividades que se realizarán de forma iterativa a lo largo del desarrollo del Proyecto se ha incluido la tabla 7 que sintetiza el calendario de entregables. La columna semana indica el número de semana dentro del periodo lectivo en el que se realizará la actividad y se generará el entregable asociado a la misma. En el caso de P1, P2 y P3 el entregable se refinará en cada una de las iteraciones, habiendo por tanto tres tres semanas diferentes. El backlog se considera un entregable continuo que puede ser supervisado por el profesor en cualquier momento para verificar que los grupos no encuentran dificultades insalvables ni en la metodología SCRUM ni en el desarrollo general del Proyecto.

3.5. Umbrales de evaluación

Para aprobar la asignatura el alumno deberá cubrir un mínimo exigible que consistirá en:

1. No tener más de un número determinado de evaluaciones de tipo Filtro NO APTAS
2. Media aritmética entre todas las actividades de Punto Fijo que supere el 50%.
3. Nota mínima de un 30% en cada actividad de Punto Fijo.
4. Funcionalidad completa de la aplicación constituyente del Proyecto.

En caso de superarse estos mínimos la forma de calcular la nota es indicada en la tabla 8, en la que figuran tanto los métodos de evaluación para cada actividad como el peso que tendrán en la nota final en caso de ser actividades de punto fijo.

Se entiende que el Extra-Bonus es aplicable con carácter general, es decir que cualquier actividad es susceptible de permitir a un alumno o grupo exhibir cualidades, competencias o productos especialmente destacables. Pero el que se pueda aplicar en todas no implica que se deba hacer, pues en ese caso tendríamos un esquema de evaluación paralela. Orientativamente consideramos que:

- El profesor no tiene que sentirse obligado a conceder Extra-Bonus en ninguna actividad concreta.
- El número de Extra-Bonus que se conceden por actividad debería estar (conceptualmente) limitado: uno o dos a lo sumo.

- La puntuación extra que se concede mediante Extra-Bonus debería tener un tope. Tentativamente pondríamos 1,5 puntos para un estudiante realmente excepcional en su forma de abordar, resolver y presentar las actividades.

	ITERACION-1	ITERACION-2	ITERACION-3	TOTAL
P0	15%			75%
P1		25%	35%	
P2				
P3				
Ejercicio Escrito Individual	5%	5%		10%
A1	FILTRO			
A2	FILTRO			
A3	FILTRO	FILTRO		
A4	FILTRO			
A5	FILTRO			
A6			5%	5%
A7				
A8				
L1	FILTRO			
L2	FILTRO			
L3	FILTRO			
L4	FILTRO			
L5	5%			5%
L6		5%		5%
TOTAL	25%	35%	40%	100%

Tabla 8: Tabla de actividasdes con sus tipos de evaluación y contribuciones a la nota final.

3.6. Reforzamiento del carácter cooperativo del curso

No todas las tareas de grupo exigen el mismo grado de cooperación. La naturaleza de las puramente presenciales tiende a colocar la colaboración en el camino del éxito, ya que han sido diseñadas con actores específicos y tareas paralelas. Ejemplos claros de esto son el puzle de patrones GRASP (A5) o el juego de los objetos animados (A4). En el otro

extremo están las actividades nucleares de cada iteración, en las que sus dimensiones se prestan más a que los estudiantes busquen mecanismos compensatorios para los casos de parasitismo que pueden sufrir en sus grupos. Los mecanismos de exigibilidad individual que queremos introducir para estos casos son los siguientes:

- Prestar el máximo cuidado a la descomponibilidad de las actividades más complejas. Los alumnos no deben estar forzados a trabajar permanentemente en grupo, pues ello supondría un dispendio injustificado de recursos. La actividad base debe ofrecer por tanto una alternativa natural para el reparto del trabajo, y al mismo tiempo las subtareas originadas deben ser difíciles de descomponer, de modo que la penalización de la no cooperación sea mayor (al implicar que algún miembro tenga que doblar su carga para compensar). En resumen, prestar atención a la interdependencia positiva basada en el método analítico.
- Detectar de manera temprana los casos de parasitismo crónico. Unas tareas Filtro individuales bien colocadas al principio del curso permiten descartar a las personas que tienen una actitud estructuralmente pasiva, que según nuestra experiencia previa no son muchas pero tienen un potencial dañino considerable. Si esta táctica se dosifica con inteligencia también puede servir para enrolar en el sistema a alumnos con actitud ambigua.
- Finalmente contemplamos para algunas actividades (P0,P1,P2 y P3) la utilización del Método del Representante (ver tablas de actividades). La evaluación sería realizada por uno de los miembros del grupo elegido al azar en el último momento, siendo la nota obtenida por ese alumno la de todo el grupo.

3.7. Recursos

El Proyecto que se plantea pretende contemplar el contexto profesional con el que se encontrará el alumno a la hora de afrontar un proyecto software real. Estas condiciones pueden resumirse en:

1. el Proyecto a desarrollar se sustenta sobre otra versión previamente realizada,
2. la versión previamente realizada está muchas veces incompleta y
3. el desarrollo del Proyecto requiere de un estudio en profundidad de la versión ya realizada.

Teniendo en cuentas estas características, y con el objetivo de simular este contexto, el recurso principal que se ofrecerá a los alumnos para el desarrollo de su Proyecto será

una aplicación inicial ofrecida por lo profesores. Esta tendrá desarrollados de forma pormenorizada aquellos aspectos que los profesores han estimado que es fundamental para la comprensión de la asignatura y dejará sin detallar los aspectos que consideran menos relevantes, dejando más libertad al alumno.

El alumno deberá estudiar el profundidad todos los artefactos ofrecidos en la versión entregada para poder realizar con éxito el desarrollo su Proyecto.

Ello garantiza que se ofrezca al alumno "el ejemplo" antes que "la teoría", de manera que sea el alumno quien realice el proceso de abstracción. A su vez este esquema reducirá la dispersión a la que puede llevar este tipo de proyectos abiertos.

Una vez identificados los artefactos utilizados en la versión, es tarea de alumno y objetivo de la asignatura, el localizar e interiorizar los recursos necesarios para su aprendizaje.

4. Planificación del trabajo del estudiante

En las siguientes tablas se muestran las actividades a realizar por los estudiantes, indicando las horas planificadas para su realización, así como los entregables asociados. Las filas coloreadas en azul corresponden a actividades asociadas a PBL.

4.1. Primera Iteración (Semanas 1 a 5)

Semana y sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
1.1	Presentación de la asignatura, de la metodología docente y del sistema de evaluación.		1,5	--	
1.2	Propuesta de ***A1: Panel del Modelo de Referencia***. Establecimiento de criterios de formación de Grupos	Búsqueda de Información sobre el Proceso Unificado de Software	1,5	3	Informe sobre las cuestiones del Panel
1.3	Presentación del **Proyecto** de la asignatura (**iteración 0**). Estudio de nuevos requisitos (**iteración 1**). Propuesta de ***P0: Id. Req.***	Estudio de requisitos basado en sistemas comerciales de alquiler de casas rurales	1,5	3	Descripción de requisitos (según objetivo de ***P0***)
2.1	Presentación del panel sobre el Modelo de Referencia (***A1***)		1,5	--	
2.2	Problema en diseño de interfaces y capas de software. ***L1: Arquitectura tres niveles***. ***L2: AWT/SWING***	Resolución del problema	1,5	2	Código del problema resuelto

Semana y sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
2.3	***A2: El cliente siempre tiene razón***. Profesor = rol de cliente y alumno = rol de desarrollador	Revisión de los requisitos identificados previamente	1,5	2	Descripción de requisitos modificado según comentarios del cliente.
3.1	***A3: Discusión dirigida*** sobre requisitos del **Proyecto**. Selección de requisitos básicos a desarrollar (**iteración 1**)		1,5	--	
3.2	Presentación por el profesor del **AnReq** de la **iteración 0**. ***L3: StarUML***	Análisis de los requisitos básicos (**iteración 1**)	1,5	2	Informe del **AnReq** desarrollado (**MCU**, **FluEv** y **MoDom**).
3.3	Problema sobre persistencia de datos (a partir de un caso de uso de un dominio particular). ***L4: DB4O***	Resolución del problema	1,5	2	Código del problema resuelto.
4.1	Presentación por el profesor del Diseño de la **iteración 0** (**DSec** y **ClaDis**). ***L5: DSec en StarUML***		1,5	1,5	Informe del Diseño (**DSec** y **ClaDis**).
4.2	***A4:Juego de los objetos animados. A5: Patrones GRASP***	Desarrollo del Diseño de los requisitos Básicos (**iteración 1**)	1,5	1,5	Informes de la actividad
4.3	Prueba escrita individual sobre Análisis de Requisitos.		1,5	1,5	Respuestas ejercicio
5 **(semana agrupada)**	Contraste por pares de la **iteración 1**	Finalizar implementación y memoria final de la **iteración 1**	2	8,5	Informe de Inspección
	Trabajo de implementación en el laboratorio.		3		Memoria final de la **iteración 1**

TOTAL Primera Iteración:	50	Presenciales:	23	No Presenciales:	27

4.2. Segunda Iteración (Semanas 6 a 8)

Semana y sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
6.1	Análisis de los problemas detectados en la **iteración 1** (casos básicos del **Proyecto**)		1,5	--	
6.2	Presentación de la **iteración 2**. Consenso y acotación de las nuevas funcionalidades a desarrollar	Desarrollo del **AnReq** de la **iteración 2**	1,5	5	Informe del **AnReq** de la **iteración 2**
6.3	En caso de ser necesario, refuerzo sobre el **AnReq** y de la persistencia de datos		1,5	--	
7.1	Trabajo en grupo relativo al diseño de la **iteración 2**.	Realización de los **DSec** y **ClaDis** de la **iteración 2**	1,5	5	Informe del Diseño de la **iteración 2**
7.2	***A3: Discusión dirigida*** sobre las soluciones relativas al ejercicio de diseño		1,5	--	
7.3	Prueba escrita individual sobre Diseño	Estudio personal	1,5	2	Respuestas
8.1	Trabajo en grupo relativo a la implementación de la **iteración 2**. Resolución de problemas colectivos.	Finalización de la implementación final de la **iteración 2**	1,5	4	Memoria de la **iteración 2**.
8.2	Contraste por pares de la **iteración 1**		1,5	2	Informe de Inspección

Semana y sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
8.3	Problema sobre el acceso remoto. ***L6:RMI***	Resolución del problema	1,5	1	Código de la solución.

TOTAL Segunda Iteración: **32,5** **Presenciales:** **13,5** **No Presenciales:** **19**

4.3. Tercera Iteración (Semanas 9 a 12)

Semana y Sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
9.1	Problemas encontrados en **iteración 2**. Presentación de la **iteración 3**. Acotación y consenso de nuevas funcionalidades		1,5	1,5	
9.2	Trabajo en grupo sobre requisitos de la **iteración 3** con sesión de **workshop** opcional		--	3	
9.3	Si es necesario, refuerzo sobre problemas de acceso remoto a la lógica de negocio		1,5	1,5	
10.1	Trabajo en grupo sobre Requisitos de la **iteración 3**	Elaboración del informe de requisitos	1,5	1,5	Informe de Requisitos de la **iteración 3**
10.2	Trabajo en grupo sobre Requisitos de la **iteración 3** con sesión de **workshop** opcional		--	3	
10.3	Trabajo en grupo sobre requisitos de la **iteración 3** con sesión de **workshop** opcional		--	3	

Semana y Sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
11.1	Trabajo en grupo sobre Diseño en la **iteración 3**	Elaboración del informe de diseño	1,5	1,5	Informe de Diseño de la **iteración 3**
11.2	Trabajo en grupo sobre diseño de la **iteración 3** con sesión de **workshop** opcional		--	3	
11.3	Trabajo en grupo sobre diseño de la **iteración 3** con sesión de **workshop** opcional		--	3	
12 (semana agrupada)		Implementación de la **iteración 3**	--	9	

TOTAL Tercera Iteración: **36** **Presenciales:** **6** **No Presenciales:** **30**

4.4. Aspectos Complementarios y Finalización del Proyecto (Semanas 13 a 15)

Semana y Sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
13.1	***A6: Concurso de desastres Informáticos***	Elaboración del informe y la presentación	1,5	6	Informes escritos
13.2	***A7: Presentación temas introductorios***	Elaboración del informe y la presentación	1,5		
13.3	***A8: Vendedores del RUP de IBM***	Elaboración del informe y la presentación	1,5		

Semana y Sesión	Actividad Presencial	Actividad No Presencial	Horas Presenciales	Horas No Presenciales	Entregables
14.1	**Workshop** de seguimiento	Trabajo en grupo en el **Proyecto** con posible extensión del **workshop**	1	7	
14.2					
14.3	Control individual sobre el **Proyecto** global	Elaboración del informe final del **Proyecto** incluyendo **las tres iteraciones**	1,5	3	Informe final del **Proyecto**
15.1	Presentación pública del **Proyecto**	Elaboración de la presentación y defensa	0,5	8	Fichero de la presentación
15.2					
15.3					

TOTAL Aspectos Complementarios y Fin Proyecto: **31,5** **Presenciales:** **7,5** **No Presenciales:** **24**

Anexos

Este apartado recoge como anexos un subconjunto de los documentos que actualmente utilizamos en la asignatura. El primero de ellos es el enunciado literal del Proyecto que realizan los alumnos a partir de un desarrollo parcial dado. Hemos incluido también el cuestionario individual que realizan los estudiantes antes de la defensa oral del mismo.

También se incorporan rúbricas específicas de la asignatura para evaluar tareas de análisis y diseño, y otras para la evaluación de las presentaciones orales, incluyendo una de auto-evaluación. El resto son referencias a manuales y libros que resultan de interés para el seguimiento de la asignatura.

Anexo I: Enunciados

A I.1 Enunciado del proyecto entregado a los alumnos (copia literal) en el curso 2011-12

Introducción

El objetivo de esta segunda iteración del Proyecto consiste en analizar, diseñar e implementar nueva funcionalidad en el sistema de información de reservas de casas rurales. En este documento se presenta la información del Proyecto desarrollado hasta el momento, durante la primera iteración del Proyecto.

1. Descripción del sistema de información
2. Modelo de Casos de Uso
3. Modelo del Dominio
4. Flujos de Eventos
5. Diagramas de Secuencia
6. Implementación e instrucciones para poner en marcha la aplicación.
7. Planificación y documentación a entregar
8. Nueva funcionalidad a añadir al sistema actual

Descripción del sistema de información

Se desea desarrollar una aplicación de gestión de reservas de casas rurales, que sirva tanto para los propietarios de las mismas como a sus posibles clientes.

Los propietarios de las casas rurales pueden darlas de alta en el sistema, proporcionando los siguientes datos: población donde se encuentra, número de dormitorios, baños, cocinas y comedores así como el número de plazas de garaje. Además, se puede incluir una descripción general de la casa. Las normas sobre casas rurales obligan a que por lo menos haya 1 cocina, 3 habitaciones y 2 baños. Por supuesto, los propietarios también podrán dar de baja en el sistema a las casas rurales.

Las casas rurales se alquilan por paquetes de días y cada propietario los organiza como quiere. Por ejemplo, en agosto se alquila por quincenas (del 1 al 15 y del 16 al 31), el puente del 1 de mayo entero (del 28 de abril al 1 de mayo), en julio por semanas, en noviembre por días, etc. Cada paquete tiene su precio.

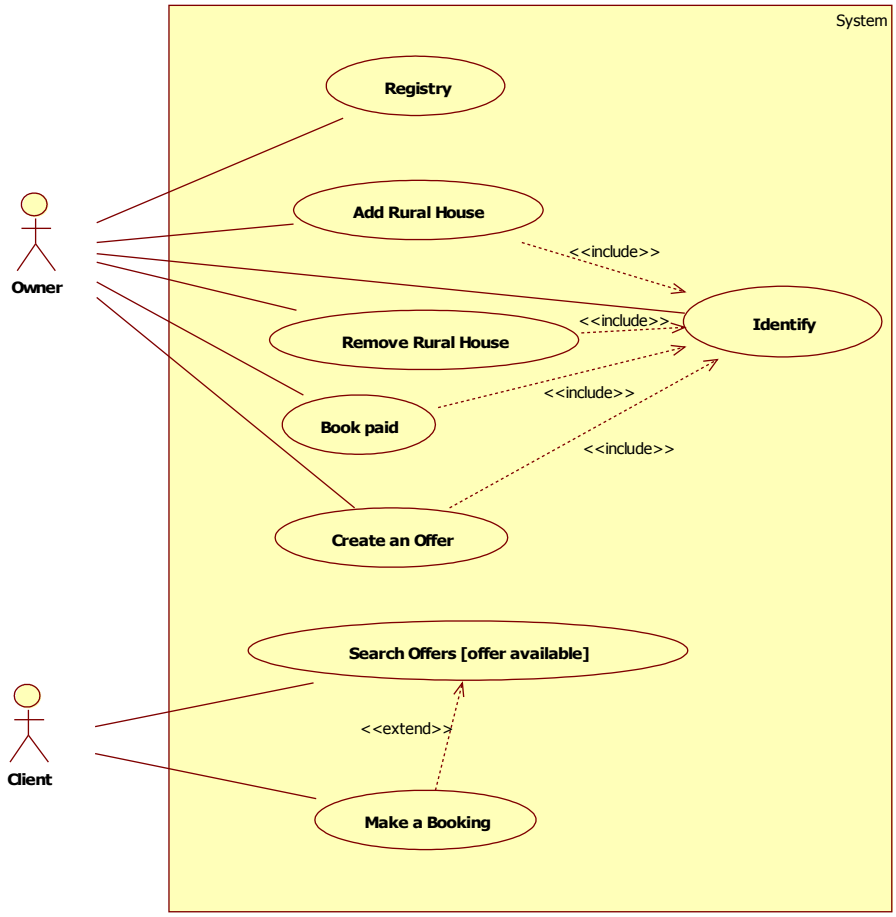
Para no perder clientes potenciales, no es necesario estar registrado en el sistema para consultar por casas rurales o hacer reservas. Sin embargo, para insertar casas rurales sí que hay que estar registrado en el sistema.

El sistema permite realizar búsquedas de las casas rurales por población. En ese caso se obtendrá una lista de las casas rurales de la población, donde podrá seleccionar una de ellas para ver las características de la casa. Si el usuario conoce el código de la casa rural, el sistema le dará la opción de ir directamente a conocer sus características.

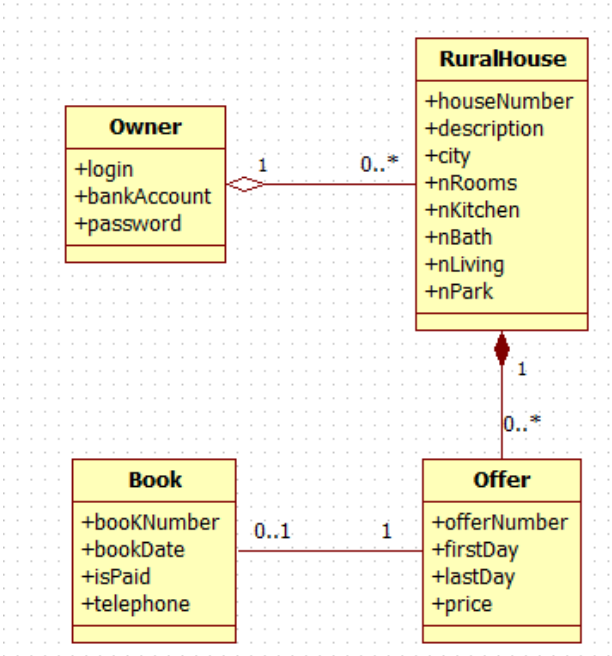
Para ver los paquetes de días disponibles de una casa, el usuario debe proporcionar el código de la casa rural, el día de entrada y la cantidad de noches. Dichos paquetes no se pueden partir, esto es, la casa se podría reservar el conjunto de días completo y no un subconjunto de los mismos. Si no hay un paquete exacto para los días solicitados por el usuario, el sistema devolverá ofertas que cumplen parcialmente la petición del usuario. Por ejemplo, si los paquetes posibles son: 5/5-7/5, 10/5-14/5 y 17/5-20/5, y el usuario quisiera reservar 15 días a partir del 3/5, el sistema devolvería los siguientes paquetes: 5/5-7/5, 10/5-14/5.

Para reservar una determinada casa habrá que indicar los siguientes datos: código de la casa, día de entrada y el número de noches que desea alquilar. El sistema comprobará si el alojamiento está libre en esas fechas y si lo está, se le mostrará un número de cuenta del propietario y una cantidad de dinero para que el usuario ingrese como fianza, y se le pedirá al usuario que introduzca un número de teléfono. Cuando el propietario compruebe más adelante que se ha realizado el pago, se pondrá la reserva como pagada.

Modelo de Casos de Uso



Modelo del dominio



Flujos de Eventos

CASO DE USO: Identify
1. El propietario introduce el nombre y la clave
2. El sistema, si existe un propietario con ese nombre y clave, le deja entrar

Flujo de eventos alternativo:
1.- Si no hay propietario con ese nombre. Fin.
2.- Si no coincide la clave para ese nombre. Fin.

CASO DE USO: Registry
1. El propietario introduce el núnero de cuenta, el nombre y la clave.
2. El sistema crea un propietario con dichos datos.
3. El sistema comunica al propietario que ha creado la cuenta.

CASO DE USO: Add Rural House
1. El propietario proporciona la población, número de habitaciones, baños, cocinas, comedores y aparcamientos, y si lo desea, una descripción de la casa.
2. El sistema crea una casa rural nueva y se la asigna el propietario.

Flujo de eventos alternativo:
1.- Si el número de cocinas es mejor que 1. Fin.
2.- Si el número de habitaciones es mejor que 3. Fin.
3.- Si el número de baños es mejor que 2. Fin.

CASO DE USO: Remove Rural House
1. El sistema muestra todas las casas rurales del propietario.
2. El propietario escoge una casa.
3. El sistema borra todas las reservas y paquetes de esa casa. Si hubiera una reserva, se le comunica al propietario.
4. El sistema borra la casa rural.

CASO DE USO: Book paid
1. El sistema presenta todas las casas rurales del propietario.
2. El propietario escoge una casa.
3. El sistema presenta al propietario todas las reservas sin pagar para esa casa rural.
4. El propietario escoge una reserva.
5. El sistema pone el estado "pagado" a dicha reserva
Flujo de eventos alternativo:
1.- Si existe una reserva caducada, se pregunta al propietario si desea eliminarla, indicándole el número de teléfono de quien hizo la reserva. Continuar.

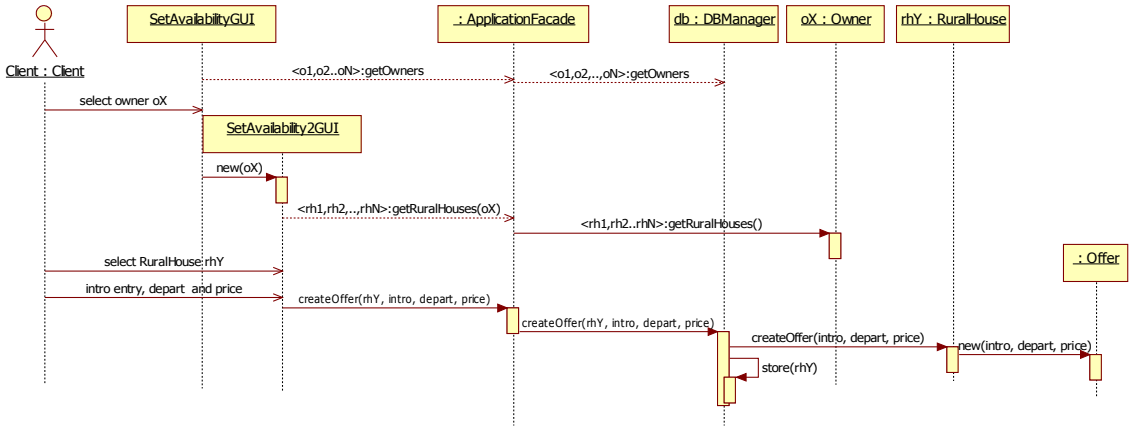
CASO DE USO: Search Offers
1. El cliente proporciona un nombre de población.
2. El sistema busca las casas de esa población y devuelve una lista de las mismas.
3. El cliente escoge una casa rural.
4. El cliente proporciona la fecha de entrada y número de días.
5. El sistema devuelve los paquetes que cumplen esas condiciones y ofrece la posibilidad de realizar la reserva. Si no hubiera paquete apropiado, se le hace saber al cliente y se le muestran paquetes de días cercanos.

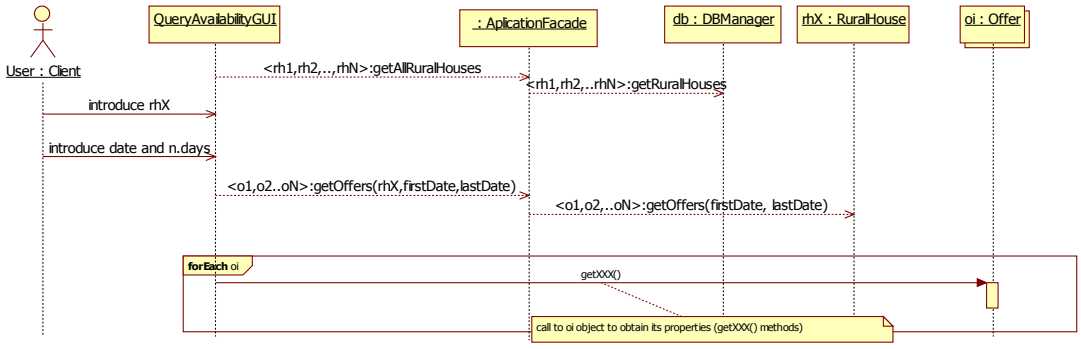
CASO DE USO: Make a Booking
1. El cliente proporciona el código de la casa, el día de entrada y cantidad de días.
2. El sistema comprueba que hay paquetes libres para esos días.
3. Si hay paquete libre para esas fechas, el sistema muestra el número de cuenta corriente del propietario, solicita un número de teléfono al cliente y crea la reserva con esas datos.
4. Si no hay reserva libre, se lo comunica al cliente.

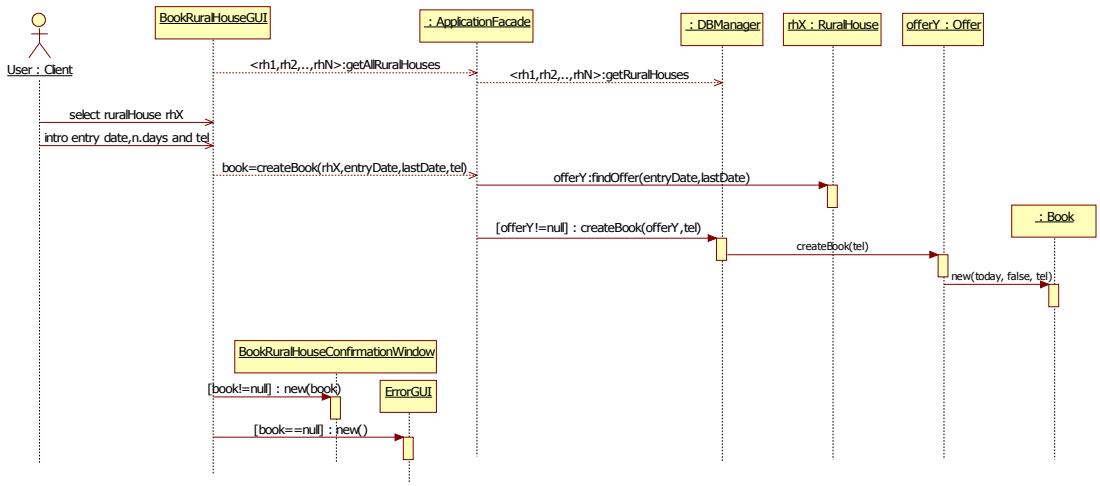
Flujo de eventos alternativo:
1.- Si no hay una casa rural con ese código. Fin.

Diagramas de secuencia de 3 casos de uso

La aplicación está diseñada utilizando una arquitectura en 3 niveles. Se utiliza AWT/SWING para definir las interfaces gráficas, el acceso a la lógica del negocio situada en otro ordenador remoto se realiza por medio de RMI, y para el nivel de datos, se utiliza la base de datos orientada a objetos db4o. Para agrupar todas las operaciones de la lógica del negocio se ha aplicado un patrón de diseño GRASP, y por ello se ha definido una clase CONTROLADOR, con todas las operaciones. A continuación presentamos los diagramas de secuencia de 3 casos de uso.







Implementación e instrucciones para poner en marcha la aplicación.

En el fichero (Proyecto.zip) se encuentran los paquetes necesarios para poner en marcha la aplicación. La aplicación está organizada en 6 paquetes:

- gui: clases con las interfaces gráficas
- bussinessLogic: clases con la lógica del negocio
- domain: clases que definen el modelo del dominio
- dataAccess: clases para proporcionar la persistencia
- configuration: paquete que guarda los parámetros de configuración
- exception: paquete donde se encuentran las excepciones

Nota: en la implementación proporcionada no se ha tenido en cuenta el flujo alternativo correspondiente al caso de uso "Create an Offer". Si alguien se anima a implementarlo, que lo haga y lo añada a la documentación presentada.

Hay que seguir los siguientes pasos para ejecutar dicha aplicación.

1. Descargar el fichero Proyecto.rar del sitio Moodle de la asignatura y descomprimirlo en C:\. Esto hará que el código y los ficheros necesarios se copien en C:\Proyecto. Si no se hace así, en los siguientes pasos, habrá que asegurarse de que los ficheros con la política de seguridad, la base de datos DB4o y las librerías DB4o tienen los nombres correctos.

2. Importar el proyecto Java (File->Import->General-> "Existing projects into workspace" y seleccionar el directorio C:\Proyecto.
3. Comprobar que las librerías necesarias están accesibles. Para ello posicionaros encima del nombre del proyecto y con el botón derecho del ratón seleccionar "Properties". Seleccionar "Java Build Path" en el menú de la izquierda que aparece en la pantalla y en la opción "Libraries" a la derecha de la pantalla pulsar "Add external Jar". Las librerías que tenéis que cargar las podéis encontrar en el fichero proporcionado:
   a) Librería JCalendar, en la carpeta AdditionalLibraries.
   b) Todas las clases de la librería Db4o, esto es, todos los ficheros jar que se encuentran ene l directorio ObjectManager-7.14/lib.
4. Configurar la aplicación. Hay que configurar las variables de la clase Config que se encuentra en el paquete configuration:
   a) El atributo javaPolicyPath, con la dirección en la que se encuentra el fichero java.policy (proporcionado en Proyecto.rar).
   b) El atributo db4oFilename, con el nombre del fichero y carpeta donde se dejará la base de datos. La carpeta que se proporcione debe existir (o crearse) en el sistema.
   c) El atributo dataBaseOpenMode, para definir el modo en que queremos que se abra la base de datos. Hay dos posibilidades: (1) initialize, para que se inicialice la base de datos con unos valores por defecto, y (2) open, para que se abra con los valores que contenga la base de datos. Usando este último modo, si la base de datos no existe, se crea una nueva.
5. Lanzar el servidor RMI. "Run" la clase businessLogic.RemoteServer.
6. Lanzar la interfaz gráfica. Run la clase gui.StartWindow. En este clase se puede configurar el atributo "isLocal". Si se configura "isLocal" a true, la lógica del negocio y la interfaz gráfica están en la misma máquina, y se accederá como una clase local. Sin embargo, si "isLocal" es false, entonces quiere decir que la lógica del negocio y la interfaz gráfica de presentación se encuentran en distintas máquinas y que se accederá por medio de RMI.

Planificación

El nuevo proyecto consistirá en añadir nueva funcionalidad al sistema actual. Dicha ampliación del Proyecto se desarrollará durante el transcurso del cuatrimestre en el que se imparte la asignatura. El último día para entregar el Proyecto es el 11 de mayo. El 11 de mayo de 10:45 a 12:15 tendrá lugar un control escrito en el aula sobre el Proyecto. Las defensas individuales del Proyecto se realizarán la semana del 14 al 18 de mayo.

La documentación del Proyecto que deberá entregarse será la siguiente:

1. Problemas encontrados. Cuáles han sido los problemas encontrados más importantes y cómo los habéis solucionado.
2. Modelo de casos de uso con la nueva funcionalidad.
3. Diagramas de secuencia en UML de la nueva funcionalidad
4. Modelo del dominio con los nuevos elementos.
5. Código fuente en formato electrónico. No hay que imprimirlo.
6. Ficheros con la documentación en Javadoc (formato.html). No hay que imprimirlos.
7. Recuento de las horas invertidas en el desarrollo del Proyecto, indicando la actividad
   a) Lectura de la documentación.
   b) Puesta en marcha de la aplicación.
   c) Extensión del MCU.
   d) Extensión del Mod. Dominio.
   e) Diagramas de secuencia nuevos
   f) Implementación en interfaces gráficos
   g) Implementación de lógica del negocio
   h) Añadir el tiempo total.
8. Conclusiones personales del trabajo realizado

Nueva funcionalidad a añadir al sistema actual

La asociación de casas rurales desea enriquecer su aplicación para que los propietarios de las casas rurales puedan incluir qué tipo de actividades pueden realizarse en cada casa rural. Esto quiere decir que es posible que haya casas rurales que no tengan posibilidad alguna de realizar ninguna actividad, mientras que otras pueden ofertar una amplia variedad de actividades. Cada propietario de cada casa rural será el encargado de indicar qué actividades oferta cada casa rural. Sin embargo, hay que tener en cuenta que una actividad como el senderismo guiado puede ser ofertada por diferentes casas pero en condiciones diferentes (p. ej. en precio o fechas).

El sistema debe permitir dar de alta a actividades. Una actividad se caracteriza por un identificador único, un nombre y un tipo de actividad (agua, aire o tierra) y no está asociada a ninguna casa rural.

También debe permitir asignar a una casa rural la lista de actividades que se pueden desarrollar. Es decir una actividad puede ofertarse en varias casas rurales, y una casa rural puede ofertar diferentes actividades. Para cada actividad asignada a una casa rural deberá especificarse: el precio, el cupo, la fecha de inicio, la fecha de finalización y una descripción detallada de la actividad concreta en esa casa rural.

Será posible consultar para cada actividad qué casas rurales la ofrecen. Esta consulta podrá verse enriquecida opcionalmente por parámetros adicionales tales como fechas, precio y cupo. Además se indicará por qué criterio se deben ordenar los resultados.

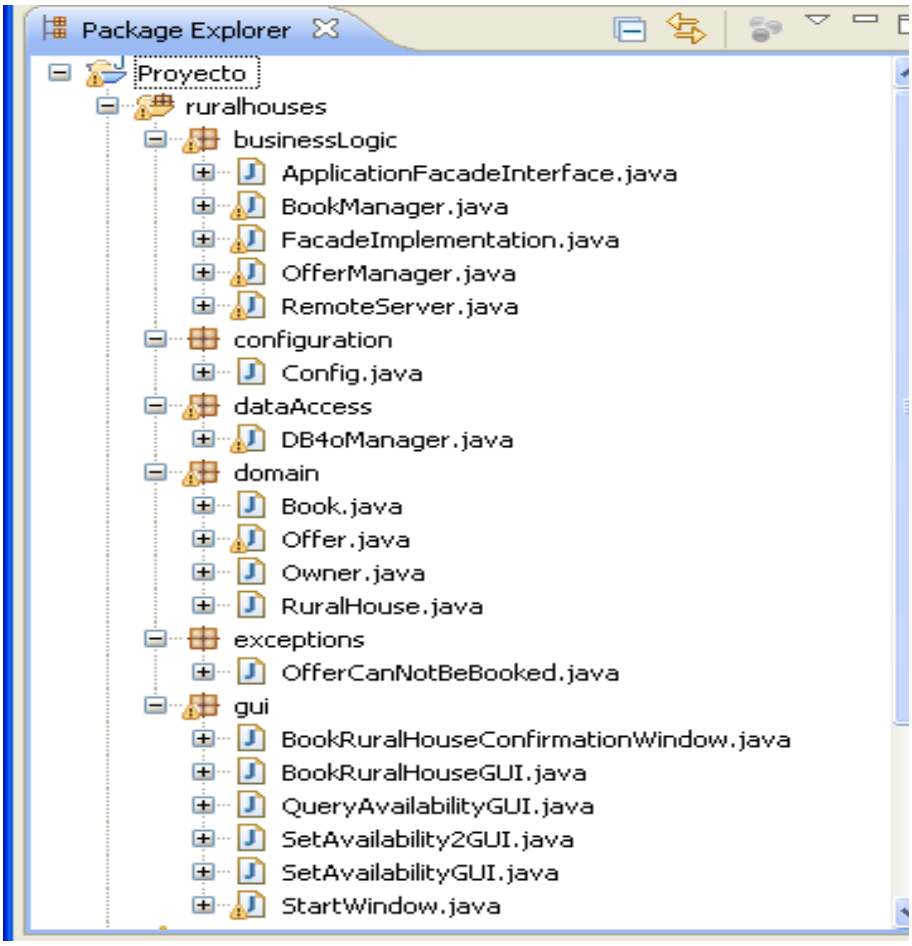
El procedimiento recomendado para la nueva funcionalidad será el siguiente: los propietarios dispondrán la posibilidad de dar de alta una nueva actividad, pero ojo sin estar asignada a ninguna casa rural. Posteriormente el propietario de la casa rural entrará en su casa rural y podrá optar a realizar la operación asignar actividad. Esta operación consultará las actividades disponibles y se mostrarán. Una vez elegida la actividad se rellenarán los datos necesarios (indicados anteriormente). Se debe ofrecer la posibilidad de dar de alta una nueva actividad si en el momento de buscar actividades para una casa rural concreta esta no existe.

A I.2 Cuestionario de evaluación individual del Proyecto.

Este cuestionario está pensado para detectar casos de parasitismo. El alumno debe haber tomado parte de forma significativa en la elaboración del proyecto para poder superarlo.

INGENIERIA DEL SOFTWARE (2º). GRADO EN INGENIERIA INFORMATICA
CONTROL PROYECTO ASIGNATURA

La siguiente figura muestra los ficheros fuente que contenía el proyecto en el momento de la entrega inicial:

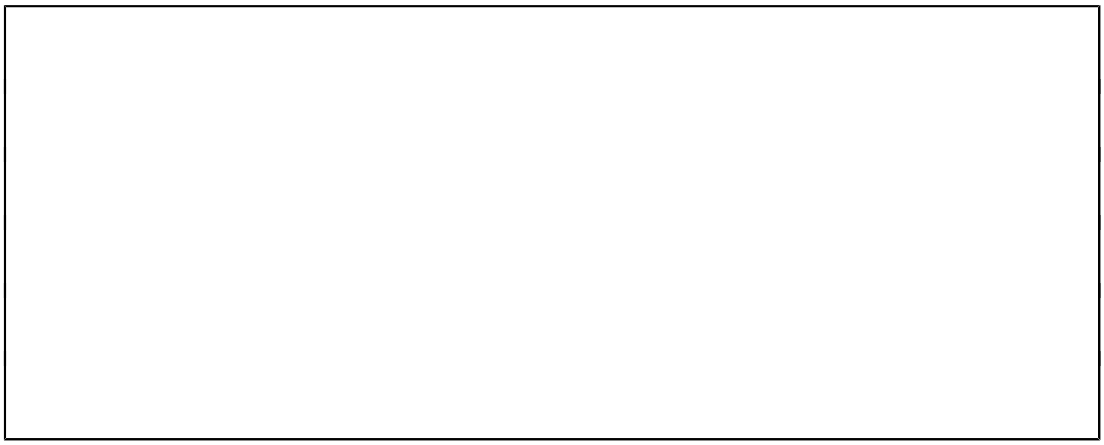


Responde a las siguientes preguntas, respetando el espacio dejado para su respuesta.

A) Como sabéis, la aplicación se puede lanzar bien utilizando un servidor RMI o prescindiendo de él. Di exactamente para cada uno de ambos casos qué variable deberás modificar y qué fichero/ficheros (y en su caso en qué orden) deberás ejecutar.

B) Indica qué ficheros has tenido que modificar para realizar el proyecto. Describe brevemente en qué ha consistido dicha modificación

C) Indica qué ficheros has añadido al proyecto y en qué paquetes. Describe brevemente para qué se han utilizado dichos ficheros.



Anexo II: Rúbricas:

Incluimos ejemplos de rúbricas adecuadas para valorar elcontenido de nuestra asignatura (casos de uso, modelo del dominio y diagramas de secuencia).

Rúbricas para los casos de uso

Nota: Los puntos máximos posibles (de acuerdo a donde se acomode mejor la entrada) se indican entre corchetes [ ].

**Hacer cosas extra (que están fuera de lo contemplado en las rúbricas) puede dar puntos extra.

	Escala			
Categoría	Bueno	Regular	Malo	Nulo
Estructura [40]	Los casos se reportan de acuerdo a la estructura planteada: nombre, descripción, actores involucrados y caso de uso relacionado (si aplica). **[40]**	Los casos reportan la mayoría de los campos solicitados. **[25]**	No se sigue la estructura en ningún aspecto; acaso se menciona el nombre del caso de uso. **[10]**	
Contenidos (caso individual) [40]	El caso representa claramente formas de utilizar el software (verbo + objeto directo); la descripción cumple con su objetivo. **[40]**	Los casos no son completamente entendibles; su nombre no se indica con verbo y objeto, pero la descripción clarifica de lo que trata el caso y esto es correcto. **[25]**	Los casos conforman palabras aisladas que la descripción no clarifica. **[10]**	No se subió la entrada. **[0]**
Completitud [15]	Se detectaron todos los casos pertinentes. **[15]**	Se detectaron la mayoría de los casos pertinentes (falta alguno que claramente	Faltan muchos casos importantes. **[5]**	

Rúbricas para el diseño con clases

	Escala			
Categoría	Bueno	Regular	Malo	Nulo
Contenido: clases (clase individual) [40]	Se reporta: el nombre de la clase, el rol que juega dentro del sistema, su visibilidad, sus atributos y sus métodos. El nombre de la clase es un sustantivo y empieza con mayúscula. La información reportada es detallada y coherente. Globalmente, se detectaron todas las clases importantes y para cada clase se detectaron todos los atributos y métodos importantes. **[40]**	Se reportan sólo algunos de los elementos solicitados. La información reportada no está lo suficientemente detallada y/o no es completamente coherente. El nombre de la clase es un sustantivo. Globalmente, falta alguna clase importante para el sistema y/o faltan atributos y métodos importantes para alguna clase. **[20]**	Básicamente no se reporta nada más que el nombre de la clase. La información reportada es incoherente y/o escueta. El nombre de la clase es un verbo o cualquier otro tipo de palabra. Globalmente, faltan muchas clases importantes para el sistema y/o faltaron atributos y métodos importantes para muchas clases. **[10]**	No se subió la entrada. **[0]**
Contenido: atributos (atributo individual) [25]	El nombre del atributo es un sustantivo o adjetivo y expresa claramente una propiedad de la clase. Se reporta su visibilidad	El nombre del atributo es un verbo u otro tipo de palabra, pero respeta la noción de propiedad. Se reporta su visibilidad	El nombre del atributo es un verbo u otro tipo de palabra. Se reporta su visibilidad como pública o no se	No se reportaron atributos. **[0]**

Ingeniería de Software I

Nombre del docente: _____

Número de equipo: _____

Rúbrica:
Diagramas de Secuencia
Fecha de creación: **17 de octubre de 2011**

A1: _____ A2: _____ A3: _____ A4: _____ (A1: Alumno1, A2: Alumno2, A3: Alumno3, A4: Alumno4)

CATEGORIA	2	1	0	A1	A2	A3	A4
Nombre y diseño	Los diagramas de secuencia incluyen su nombre y la distribución de los objetos, mensajes y activación es ordenada, lo que permite que sean claros de comprender.	Los diagramas de secuencia incluyen su nombre pero la distribución de los objetos, mensajes y activación es desordenada, lo que complica que sean claros de comprender.	Sólo algunos diagramas de secuencia incluyen su nombre y la distribución de los objetos, mensajes y activación es desordenada, lo que complica que sean claros de comprender.				
Objetos	Los diagramas de secuencia identifican perfectamente los objetos involucrados en cada uno de ellos.	Sólo algunos diagramas de secuencia identifican perfectamente los objetos involucrados en cada uno de ellos.	Los diagramas de secuencia no identifican claramente los objetos involucrados en cada uno de ellos.				
Línea de tiempo y Activación	La línea de tiempo de los diagramas de secuencia incluye la activación de cada uno de los objetos durante el lapso de tiempo que corresponde.	La activación no se incluye en la línea de tiempo de cada uno de los objetos durante el lapso que corresponde en los diagramas de secuencia.	No existe activación en la línea de tiempo de los objetos identificados en los diagramas de secuencia.				
Mensajes	Los mensajes agregados en los diagramas de secuencia, están colocados y numerados correctamente.	Los mensajes agregados en algunos de los diagramas de secuencia, están colocados y numerados correctamente.	Los mensajes agregados en los diagramas de secuencia no están colocados ni numerados correctamente.				
Número de diagramas	El equipo entregó de 10 a 13 diagramas de secuencia.	El equipo entregó de 6 a 9 diagramas de secuencia.	El equipo entregó menos de 6 diagramas de secuencia.				
			Total:				

El siguiente libro presenta rúbricas a ser consideradas para la presentación escrita de trabajos (pp. 193 en adelante):

"Aprendizaje Basado en Competencias". A. Villa y M. Poblete. Ediciones Mensajero. Universidad de Deusto, 2007.

Por último, se incluyen rúbricas para la presentación oral del trabajos. La primera de ellas está pensada para autoevaluación de los estudiantes y la segunda para evaluación por parte del docente.

Rúbrica para la Auto-evaluación Presentación Oral

	4	3	2	1
Dominé ampliamente el tema				
Apliqué la información (ofrecí ejemplos)				
Introduje bien el tema				
Seguí un orden o una estructura definida				
Las transiciones de un tema a otro fueron lógicas				
Hice un uso adecuado del tiempo				
Demostré seguridad				
Me expresé con corrección				
La información que presenté se entiende				
Pude aclarar dudas o conceptos				
Usé un vocabulario adecuado y variado				
Presenté la información de forma creativa				
Mi volumen de voz fue lo suficientemente alto para ser escuchado por todos los miembros de la audiencia a través de toda la presentación.				
Tuve buena postura, estuve relajado y seguro de mi mismo. Establecí contacto visual con todos en el salón durante la presentación.				
Mis expresiones faciales y lenguaje corporal generaron un fuerte interés y entusiasmo sobre el tema				
Utilicé elementos visuales tales como tablas, ilustraciones y gráficas. Las imágenes fueron relevantes al tema, tenían el tamaño adecuado, fueron de buena calidad y aumentaron el interés del lector.				
Uitlicé fonts fáciles de leer. El uso de itálicas, negritas y sangría facilitó la lectura del texto. El fondo y los colores utilizados facilitaron la lectura del texto.				
Total:				

Describa lo más que le gustó de su presentación:

¿Cómo podría mejorar la misma?

	Rúbrica Presentación Oral			
Criterio	**Nivel 4**	**Nivel 3**	**Nivel 2**	**Nivel 1**
Volumen de voz	El volumen es lo suficientemente alto para ser escuchado por todos los miembros de la audiencia a través de toda la presentación.	El volumen es lo suficientemente alto para ser escuchado por todos los miembros de la audiencia al menos 90% del tiempo.	El volumen es lo suficientemente alto para ser escuchado por todos los miembros de la audiencia al menos el 60% del tiempo.	El volumen con frecuencia es muy débil para ser escuchado por todos los miembros de la audiencia.
Postura del cuerpo y contacto visual	Siempre tiene buena postura y se proyecta seguro de sí mismo. Establece contacto visual con todos en el salón durante la presentación.	Casi siempre tiene buena postura y establece contacto visual con todos en el salón durante la presentación.	Algunas veces tiene buena postura y establece contacto visual.	Tiene mala postura y/o no mira a las personas durante la presentación.
Habla claramente	Habla claramente y distintivamente todo el tiempo (100-95%).	Habla claramente y distintivamente casi todo el tiempo. (80 – 95 %)	Habla claramente y distintivamente la mayor parte (70-85%) del tiempo.	A menudo habla entre dientes o no se le puede entender.
Conocimiento del tema	Demuestra un conocimiento completo del tema.	Demuestra un buen conocimiento del tema.	Demuestra un buen conocimiento de partes del tema.	No parece conocer muy bien el tema.
Contestar preguntas	El estudiante puede con precisión contestar casi todas las preguntas planteadas sobre	El estudiante puede con precisión contestar la mayoría de las preguntas planteadas sobre	El estudiante puede con precisión contestar unas pocas preguntas planteadas sobre el tema por sus	El estudiante no puede contestar las preguntas planteadas sobre el tema por sus compañeros de clase.

	el tema por sus compañeros de clase.	el tema por sus compañeros de clase.	compañeros de clase.	
Entusiasmo	Expresiones faciales y lenguaje corporal generan un fuerte interés y entusiasmo sobre el tema en otros.	Expresiones faciales y lenguaje corporal algunas veces generan un fuerte interés y entusiasmo sobre el tema en otros.	Expresiones faciales y lenguaje corporal son usados para tratar de generar entusiasmo, pero parecen ser fingidos.	Muy poco uso de expresiones faciales o lenguaje corporal. No genera mucho interés en la forma de presentar el tema.
Uso del tiempo	Utiliza el tiempo adecuadamente y logra discutir todos los aspectos de su trabajo.	Utiliza el tiempo adecuadamente pero al final tiene que cubrir algunos tópicos con prisa	Confronta problemas menores en el uso del tiempo (termina muy pronto o no logra terminar su presentación el tiempo asignado	Confronta problemas mayores en el uso del tiempo (termina muy pronto o no logra terminar su presentación el tiempo asignado
Uso de gráficas, tablas e imágenes	Incluye elementos visuales tales como tablas, ilustraciones y gráficas. Las imágenes son relevantes al tema, tienen el tamaño adecuado, son de buena calidad y aumentan el interés del lector.	Incluye elementos visuales tales como tablas, ilustraciones y gráficas. Las imágenes son poco relevantes al tema y no tienen el tamaño adecuado.	Los elementos visuales son pobres y no abonan a la presentación. Las imágenes son seleccionadas al azar, son de pobre calidad y distraen al lector.	No incluye elementos visuales.
Organización	Se presenta la información de forma lógica e interesante que la audiencia puede seguir.	Se presenta la información utilizando una secuencia lógica que la audiencia puede seguir.	La audiencia tiene dificultades siguiendo la presentación porque se brinca de un tema a otro.	La audiencia no puede entender la presentación debido a que no sigue un orden adecuado

Errores gramaticales y "typos"	La presentación no tiene errores gramaticales	La presentación tiene no más de dos errores gramaticales.	La presentación tiene tres errores gramaticales.	La presentación tiene cuatro o más errores.
Elementos del texto	Los fonts son fáciles de leer y el tamaño de letra varía apropiadamente en los encabezamientos y el texto. Uso de itálicas, negritas y sangría facilita la lectura del texto. El fondo y los colores utilizados facilitan la lectura del texto.	A veces los fonts son fáciles de leer pero en algunos slides, fonts, itálicas, negritas, párrafos, colores y fondos obscuros, dificulta la lectura.	Se dificulta la lectura general de la presentación con párrafos muy largos, fonts diferentes y fondos obscuros.	El texto es extremadamente difícil de leer con largos bloques de texto y tamaños de letra muy pequeños, inapropiado contraste de colores.
Creatividad	Presenta el material creativamente y de forma espontánea	Hay algún tipo de originalidad, con buena variedad de texto y gráficas.	Poca o ninguna variación; poca originalidad e interpretación	Repetitivo, con poca o ninguna variedad

Anexo III: Referencias

A III. 1 Manuales:

Manual de SCRUM:

http://jeffsutherland.com/scrumhandbook.pdf

Guia de SCRUM en castellano:

http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20ES.pdf#view=fit

A III.2 Bibliografía básica:

INGENIERÍA DEL SOFTWARE: UN ENFOQUE PRÁCTICO. *Roger S. Pressman.* MacGraw-Hill, 2001. 5ª Edición (referencia general sobre Ingeniería del Software).

UML Y PATRONES: INTRODUCCIÓN AL ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS. *Craig Larman.* Prentice-Hall, 2003 (sobre UML y patrones GRASP).

A III.3 Bibliografía profundización:

EL PROCESO UNIFICADO DE DESARROLLO DEL SOFTWARE. Ivar Jacobson, Grady Booch y James Rumbaugh. Pearson-Addison Wesley, 1999 (sobre UML y proceso unificado de desarrollo de software).

CONSTRUCCIÓN DE SOFTWARE ORIENTADO A OBJETOS. Bertrand Meyer. Prentice-Hall, 1998 (sobre Orientación a Objetos).

Anexo IV: Glosario

AnReq: Análisis de Requisitos. Disciplina cuyo objeto es traducir las necesidades del cliente a un lenguaje que sea lo suficientemente preciso para poder acometer el diseño de la aplicación sin ambigüedades y, a ser posible, sin dejar de ser comprensible para el cliente).

AWT/SWING: *Abstract Window Toolkit*. Bibliotecas de clases Java que permiten implementar gráficos, interfaz de usuario y sistema de ventanas para las aplicaciones.

Caso de uso: Descripción de una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un usuario o proceso externo. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

ClaDis: Clases de Diseño. Parte de la representación de los datos que no se deriva del dominio del problema, sino de las necesidades del diseño. Vendría a ser la parte del modelo de datos que no corresponden a entidades del mundo real y también se llaman utilidades.

db4o: *DataBase for Objects*. Plataforma para construir bases de datos a partir de objetos. Es muy sencilla de usar y minimiza el esfuerzo cuando se utiliza diseño de software orientado a objetos (hoy en día la solución más sólida y extendida con diferencia). Tiene licencia GPL para uso no comercial.

DCla: Diagrama de Clases. Tipo de diagrama usado en UML que describe la estructura de un sistema mostrando sus clases y las relaciones entre ellas.

DCU: Diagrama de casos de uso. Tipo de diagrama usado en UML que describe qué casos de uso tiene un sistema software, qué actores externos (usuarios o

procesos) son necesarios para que se materialicen y cómo se relacionan entre sí.

Diseño: Disciplina cuyo objeto es traducir los resultados del análisis de un sistema software -en nuestro contexto, del Análisis de Requisitos (AnReq)- a una especificación de los objetos y las acciones que estos deben ejecutar para completar las funcionalidades descritas en el análisis, de manera que dicha especificación sea traducible a código con un esfuerzo razonable.

DSec: Diagrama de secuencia. Tipo de diagrama usado en UML que describe cómo y en qué orden interaccionan los objetos en un sistema. Se usa para describir el comportamiento de las distintas clases de un sistema software para solventar una funcionalidad concreta, típicamente en la fase de Diseño.

FluEv: Flujo de Eventos. Descripción del comportamiento de un sistema en términos de eventos. En nuestro contexto son utilizados en el Análisis de Requerimientos para documentar casos de uso, por lo que suelen constar fundamentalmente de acciones externas entre los usuarios y el sistema.

GRASP: *General Responsibility Assignment Software Patterns*. Patrones que proporcionan auxilio a problemas muy generales que se presentan en el diseño orientado a objetos. Indican criterios para, dado un comportamiento o tarea, elegir qué objetos de los existentes deben incorporarlo, o si no cuándo resulta una buena práctica construir una utilidad nueva para ello.

Iteración: Acto de repetir un proceso con el objetivo de aproximarse a una meta. En Ingeniería del Software cada iteración produce una serie de artefactos o documentos que son utilizados en la siguiente. Las metodologías ágiles prescriben que el artefacto básico y sustancial de toda iteración debe ser código ejecutable.

MCU: Modelo de Casos de Uso. Descripción de los requerimientos de un sistema software en términos de casos de uso, sus diagramas (DCU) y los correspondientes flujos de eventos (FluEv).

MoDom: Modelo del Dominio. Descripción estática de los objetos que forman parte del dominio sobre el que debe operar un sistema software. En nuestro contexto lo representamos mediante Diagramas de Clases que sólo contienen los objetos del dominio. Más tarde, en fase de diseño, pueden enriquecerse con las utilidades incorporadas (ClaDis).

RMI: *Remote Method Invocation*. Mecanismo simple que permite que una clase Java ejecutándose en un equipo invoque un método de otra clase

ejecutándose en un equipo remoto. Funciona como servicio TCP y requiere el uso de Java en los dos extremos de la conexión.

RUP: *Rational Unified Process*. Versión comercial y completa del Proceso Unificado (UP), desarrollada por Rational, más tarde absorbida por IBM. Incluye una potente plataforma software para la generar, gestionar y validar los modelos producidos.

StarUML: Herramienta de código abierto para crear, mantener y documentar los diagramas UML relacionados con un proyecto software. Sólo existe versión para Windows.

UML: *Unified Modeling Language*. Lenguaje estandarizado para cubrir las necesidades de especificación y documentación de todas las disciplinas involucradas en el desarrollo del software orientado a objetos.

UP: *Unified Process*. Conjunto de metodologías de desarrollo de software que funcionan como estándar *de facto* para el análisis, implementación y documentación de sistemas orientados a objetos. Es una versión restringida y de dominio público del RUP.

PROGRAMA ERAGIN III

GRUPO DE INGENIERIA DEL SOFTWARE:

ALFREDO GOÑI

JESÚS IBÁÑEZ

JON ITURRIOZ

JOSÉ ÁNGEL VADILLO

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

FACULTAD DE INFORMÁTICA

GUÍA DEL ESTUDIANTE

Índice

1. Planteamiento del Proyecto

Acabas de llegar a la importante empresa de software Sinking Soft. Una de las primeras cosas que te sucederá es que te caerán "marrones": trabajos que han quedado sin terminar y que hay que completar para que cumplan todas las especificaciones. Inicialmente se te darán mal documentados, pero progresivamente empezarás a ver que otros trabajos vienen acompañados de información útil (artefactos) que simplifican mucho tu tarea. Si consigues superar este estadio inicial con éxito tendrás un encargo de mayor responsabilidad (el Proyecto, que también será un trabajo dejado a medias por otros compañeros) y formarás un equipo, pero siempre teniendo en cuenta que tu actuación está a prueba y que hasta que no completes dicho proyecto no se te ofrecerán responsabilidades reales ni condiciones laborales mejores.

La pregunta que debes hacerte durante este periodo de prueba y formación es la siguiente:

¿Tú crees que vas a durar mucho en Sinking Soft?

1.1. Escenario del Proyecto

"Los habitantes de Villatripas de Arriba han mantenido el aspecto de su pueblo y restaurado sus viviendas para alquilarlas como casas rurales. Sin embargo el negocio no marcha bien: casi todos los turistas se hospedan en Villatripas de Abajo, que además de ser mucho más feo, está habitado por imbéciles, como todo el mundo sabe.

La razón parece estar en que en Villatripas de Abajo tienen una web estupenda que enseguida capta a los clientes potenciales dándoles información útil y atractiva y guiándoles en su elección. Sin embargo, la web de Villatripas de Arriba tiene enlaces muertos, no permite meter información individualizada por cada casa, no obliga al cliente a introducir un adelanto para reservar la casa (con lo que muchas reservas no se cumplen), y además no verifica que las casas cumplan con los requisitos de la Dirección General de Turismo Rural (número mínimo de baños, existencia de cocina, etc...) por lo que han tenido que hacer frente a denuncias.

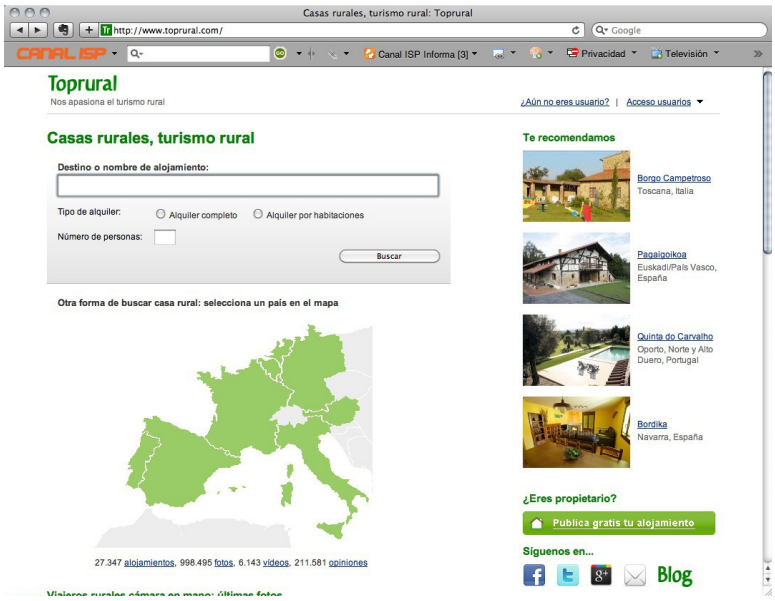
Don Manolo Pillaste, alcalde de Villatripas de Arriba, ha decidido renovar la web, pero como la otra vez le engañaron, quiere estar seguro, antes de hacer el encargo, de que le instalan todas las funcionalidades. Sinking Soft es una de las candidatas y tiene que

construir y presentar una descripción del servicio a desarrollar. Como no parece un cliente importante (la verdad es que sólo han aceptado un presupuesto miserable) la empresa se lo va a encargar tu grupo. Si salís adelante y satisfacéis al cliente se os prorrogará el contrato otros diecisiete días. Si no, se os echará a la calle y no se os devolverá la fianza que se os ha cobrado en concepto de uso del papel higiénico de la empresa.

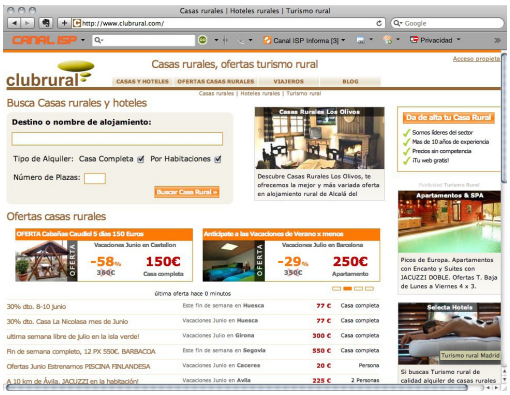
1.2. Primeros pasos (el primer marrón)

Lo primero que tienes que hacer es tener una mínima familiaridad con el problema. Existen en el mercado muchas aplicaciones web que permiten gestionar el alquiler de casas rurales, tanto para clientes como para los propietarios de las mismas, así que empieza a buscarte la vida. Una búsqueda en Google del texto "casas rurales" te devolverá infinidad de resultados entre los que se encontrarán:

- http://www.toprural.com/



- http://www.clubrural.com/ o http://www.homeaway.es/



El Proyecto que un grupo de novatos como tú va a desarrollar es una aplicación de gestión de casas rurales que nazca con el objetivo de superar a las actualmente existentes (que harían el papel de Villatripas de Abajo) en al menos alguna característica. No se espera que el producto final quede en explotación, pero sí que obtengas como resultado un prototipo que sea funcional y que muestre las bondades de la aplicación propuesta.

Pero antes de formar el grupo queremos comprobar que sabes en qué te estás metiendo. Tu primera actividad individual consistirá en una reflexión de los requisitos del proyecto, teniendo en cuenta no sólo las especificaciones y artefactos recibidos sino también algunos ejemplos reales como los comentados anteriormente. Si lo haces bien te meteremos en un grupo del proyecto y tendrás ocasión de hacer más méritos.

2. Metodología y desarrollo del proyecto

2.1. Desarrollo iterativo

Tras la actividad inicial individual ("el primer marrón") se os presentará el *enunciado inicial del Proyecto* se os presentará y formaréis los grupos. Junto con el enunciado se os entregará artefactos (programa y documentación) correspondientes a una iteración ya desarrollada del proyecto (**Iteración 0**).

Después tendréis que realizar tres iteraciones:

- Primera iteración: semanas 1-5
- Segunda iteración: semanas 6-8
- Tercera iteración: semanas 9-12

Todas las iteraciones tienen la misma estructura con tres actividades: **Captura de Requisitos**, **Diseño** e **Implementación**. La metodología que usaremos se inspira en el juego del rugby: en cada iteración se pasara el balón hacia adelante para avanzar lo más posible. Para comenzar la siguiente se hace un pase hacia atrás para buscar una mejor posición de arranque. El objetivo es, naturalmente, conseguir un avance más profundo.

Estas tres **Actividades de desarrollo del Proyecto** serán, por tanto, recurrentes y deberán revisarse y completarse en cada una de las tres iteraciones. Para ello tendréis que buscar documentación e inspiraros en el material que se os ha entregado para conocer y aplicar los aspectos de cada fase, comenzando por los más básicos y profundizando a medida que se complican las características del producto a desarrollar.

El desarrollo iterado pretende también gestionar el cambio, inherente a todo proyecto informático. Al principio tanto el grupo de trabajo como el cliente tienen ideas más imprecisas sobre lo que se quiere o puede obtener como producto. Pero a medida que se avanza, tanto el grupo (que domina mejor la metodología) como el cliente (que ve cómo toma forma la aplicación) se van dando cuenta de más posibilidades, nuevos requisitos, mejoras en la presentación o en la eficiencia, etc..., y estos cambios son incorporados como metas adicionales para la siguiente iteración.

2.2. *Metodología de trabajo en grupo*

Tu grupo estará formado por ***3 personas***. Es importante, debido a la metodología a seguir (SCRUM), que todos los miembros del grupo tengáis unas horas comunes libres para poder realizar las reuniones y este será uno de los criterios para su formación. Inicialmente se os dejará a vuestro criterio el establecimiento del grupo, y solo intervendrá el profesor si se observan conflictos e incompatibilidades a la hora de crearse estos (alumnos descolgados, número de alumnos no múltiplo de tres, etc.).

En este sentido no imponemos muchas restricciones, pero sí es importante conocer *a priori* las reglas de juego y lo que supone para el grupo que un integrante no realice las actividades en plazo e intensidad requeridos. La realización de pruebas de evaluación a integrantes del grupo elegidos al azar (método del representante, cuya calificación es la que obtendrá el grupo) o la calificación de todo el grupo con la media de las calificaciones individuales son algunas prácticas que se efectuarán a modo de control de la exigibilidad individual y debéis conocer su existencia y considerarlas a la hora de formar el grupo.

Para que desarrolléis vuestras habilidades básicas de trabajo en grupo vamos a utilizar una metodología sencilla de entender y de aplicar: *SCRUM.* Esta metodología es ampliamente utilizada en metodologías ágiles de desarrollo del software, por su naturaleza de proceso, en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar en grupo de manera colaborativa y obtener el mejor resultado posible de un proyecto.

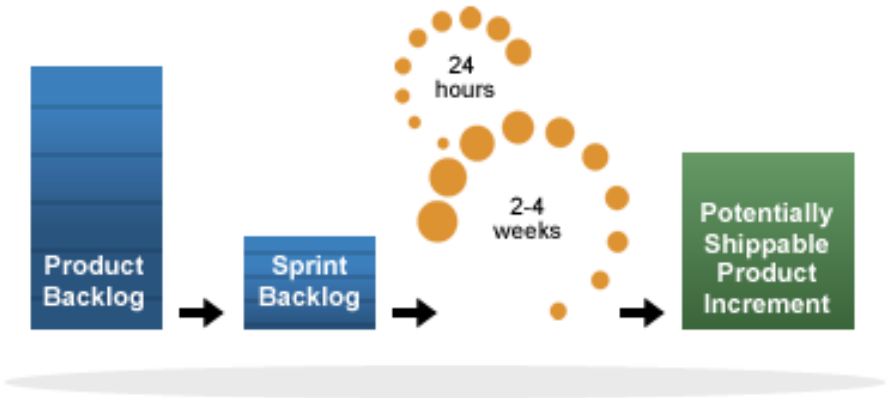


Figura 1: El proceso SCRUM (fuente http://pkab.wordpress.com/2008/07/11/scrum)

La metodología está basada en roles que permiten distribuir las responsabilidades y tareas de una forma relativamente natural:

1. El **Dueño del proyecto**. En un proyecto real sería el cliente, pero en nuestro caso tendrá que ser desempeñado por el profesor.

2. El *Scrum Master* o **Responsable del proyecto**. En cada iteración sería uno de los componentes del grupo.
3. El *Scrum Team* o **Equipo del proyecto**. Lo formarían los otros dos miembros del grupo.

En SCRUM un proyecto se ejecuta en bloques temporales cortos y fijos llamados **sprints**. La noción de *sprint* es una forma particular de ver las iteraciones de un proyecto, y el nombre únicamente recalca la importancia del tiempo y la distancia fija a una meta. Como comprobaréis en la asignatura, cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

Cada una de las iteraciones de vuestro Proyecto se planificará, de acuerdo con la metodología, mediante tres acciones principales, cada una de las cuáles implicará una serie de reuniones.

- **Selección de requisitos** (una sesión de clase). El cliente presentará al equipo la lista de requisitos más o menos priorizada del producto o proyecto. El equipo preguntará al cliente las dudas que surjan y seleccionará los requisitos más prioritarios que se compromete a completar en la iteración. En nuestro caso los profesores en nuestro rol de clientes marcaremos las pautas de la selección de requisitos para cada iteración, sobre todo en la iteración inicial.
- **Planificación de la iteración**. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo os distribuiréis las tareas. Esta planificación se recogerá en un **backlog** del Proyecto. El volumen de trabajo está pensado para tres personas, de modo que un reparto inadecuado sólo os producirá perjuicios y retrasos. El profesor puede elegir al azar al alumno que debe explicar el estado del *backlog*, cómo se ha realizado el trabajo hasta un momento dado y cuáles son los próximos pasos. Aunque cada miembro del grupo se centre en sus tareas, no puede desentenderse de lo que hacen los compañeros, porque al final debe estar en condiciones de explicar el estado del desarrollo del proyecto. Además, si se demostrara que sólo conoce su parte, eso querría decir que no es dueño del proyecto y que está funcionando como un trabajador subcontratado. El jefe de Personal de Sinking Software decidiría inmediatamente que su trabajo es de poca cualificación y que su sueldo es demasiado elevado para ello.

- **Ejecución de la iteración**. Para ello SCRUM propone una reunión diaria de sincronización (15 minutos máximo). Cada miembro del equipo inspeccionará el trabajo que el resto esté realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. En la reunión cada miembro del equipo debe responder a tres preguntas:

  1. ¿Qué he hecho desde la última reunión de sincronización?
  2. ¿Qué voy a hacer a partir de este momento?
  3. ¿Qué impedimentos tengo o voy a tener?

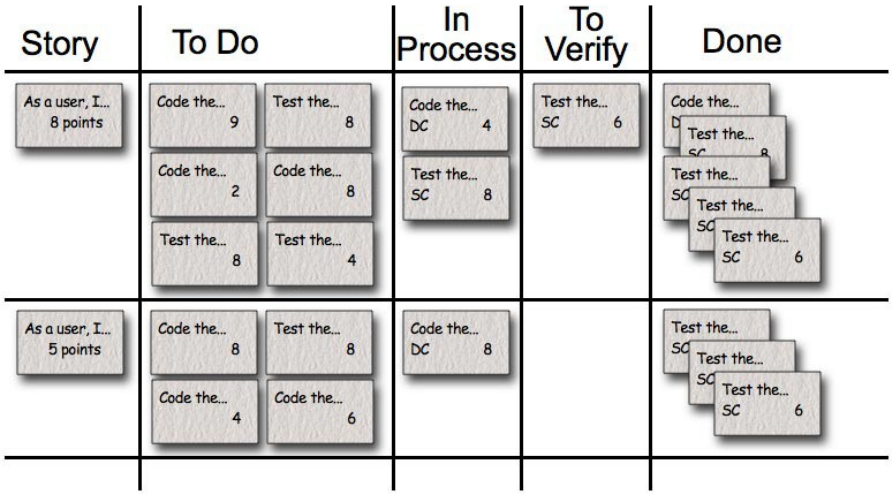


Figura 2:. Ejemplo de *backlog (fuente: http://www.mountaingoatsoftware.com/scrum/task-boards)*

Somos conscientes de las dificultades que tenéis para realizar una reunión diaria, y teniendo en cuenta vuestra dedicación real debería ser suficiente con dos o tres reuniones semanales. Además el profesor fomentará estas reuniones en parte del horario de nuestras sesiones lectivas, lo que le permite monitorizar el proceso y exigir al grupo informes con las respuestas de cada miembro del grupo a las tres preguntas planteadas anteriormente.

El *backlog* puede ser una estructura muy compleja que sirva como elemento de gestión global. El mínimo que exigiremos será una lista de tareas de grano suficientemente fino como para describir de manera clara las responsabilidades y tareas del grupo y de cada uno de sus miembros. La lista surge durante la planificación de la iteración, las tareas son las que el equipo define como necesarias para conseguir el objetivo, identificándose el responsable de hacer el trabajo y estimándose el esfuerzo requerido para completarla. Por último, el *backlog* debe reflejar claramente el avance

diario del proyecto, así como los riesgos y problemas identificados sin exigir procesos complejos de gestión. Es también una herramienta de soporte para la comunicación directa del equipo y promueve la **reflexión crítica del grupo**. Debe contener accesos o enlaces a los artefactos relacionados con cada tarea completada o en curso. El *backlog* debe también ser accesible para el profesor como herramienta fundamental de seguimiento del Proyecto. Para su soporte físico del hay varias posibilidades a vuestro alcance:

- Hojas de cálculo compartidas en Google Docs, DropBox u otra aplicación en la nube.
- Pizarra física o pared (con pegatinas simulando las tareas), de la que se pueden extraer y publicar fotos.
- Aplicaciones de Gestión de Proyectos que incorporan SCRUM.

En resumen SCRUM, por su propia naturaleza, fomenta el **compromiso** conjunto y la **colaboración** entre los miembros del equipo. La **transparencia** entre todos es fundamental para poder entender la situación real del proyecto y así poder hacer las mejores adaptaciones que permitan conseguir el objetivo común. Por ello, su implantación como marco de trabajo en el desarrollo del proyecto creemos que potencia decisivamente la adquisición de **habilidades interpersonales y de trabajo en grupo**, tan necesarias en el entorno profesional de la Ingeniería Informática.

2.3. Preguntas guía

La metodología SCRUM está pensada como herramienta en entornos eminentemente profesionales. Aunque estemos simulando uno de dichos entornos (la empresa Sinking Soft) no debemos olvidarnos de que todavía estamos en el aula y de que muchas veces no tendremos los conocimientos mínimos para abordar los problemas que se nos planteen, así que tendremos que compatibilizar "lo que hay que hacer" con "lo que hay que aprender", porque sin lo segundo lo primero será imposible. En todo caso no será el profesor el que decida "lo que hay que aprender", y los grupos han de ser autónomos para tomar ese tipo de decisiones. Lo que sí se os proporciona es una serie de cuestiones que deberíais plantearos al inicio de cualquier reunión del grupo (excluyendo las de sincronización) con respecto a la tarea en curso, y que denominamos preguntas-guía:

1. ¿Qué es exactamente lo que debemos hacer? (especificación)
2. ¿Qué es lo que no tenemos por qué hacer? (alcance)

3. De las cosas que sabemos, ¿cuáles pueden ser útiles para resolver la tarea? (recursos previos)
4. ¿Qué deberíamos saber hacer y no sabemos? (objetivos de aprendizaje)
5. ¿Existe algún procedimiento más o menos estándar para hacerlo? ¿Cuál es? ¿Dónde hay información sobre el mismo? (técnicas)
6. En caso de haber varios procedimientos, ¿cuál es el más sencillo? (limitación de recursos)
7. ¿Hay en la versión incompleta del Proyecto que nos han entregado algo que nos pueda servir como modelo para plantear, resolver o documentar el trabajo restante? (inspiración).

Adicionalmente tenemos otra batería de preguntas guía orientadas a las fases de desarrollo del Proyecto e irán planteándose dependiendo del momento en la que nos encontremos. Contemplamos cuatro fases:

1. Identificación de los requisitos. ¿Cuáles son las funcionalidades que va a ofrecer nuestro Proyecto?
2. Captura de requisitos. ¿Qué elementos creemos que hay que identificar en la captura de requisitos? ¿Existen procedimientos regulados y eficientes para capturarlos sin errores?
3. Diseño del sistema. Una vez capturados los requisitos, ¿Cómo diseñaremos cada caso de uso? ¿Utilizaremos algún paradigma concreto? ¿Existe algún tipo de artefacto específico para el diseño? ¿Hay algún principio organizativo que facilite el diseño? ¿Existen patrones de diseño ya establecidos? ¿Cómo vamos a realizar la persistencia de los objetos?
4. Implementación del sistema. Una vez diseñados los casos de uso, ¿En qué lenguaje de programación vamos a implementarlo? ¿Que tecnología vamos a utilizar para implementar cada nivel? ¿Qué Base de Datos vamos a utilizar para realizar la programación distribuida (cliente/servidor)?

2.4. Entregables del Proyecto

Tras la inspección de cada iteración el grupo deberá realizar una **Memoria** correspondiente a cada una de las tres Actividades. Estas tres memorias deberán estar enlazadas desde el *backlog*.

Dado que al final de cada iteración el grupo debe tener una aplicación operativa, aunque quizás incompleta, debe ser posible hacer una presentación de control para ver que se cumplen los objetivos. Inicialmente el papel de controlador será desempeñado por otro equipo. Al final de las iteraciones 1 y 2 tendremos una sesión de contraste entre grupos, a realizar en el laboratorio. Cada grupo tendrá como tarea asistir a la defensa del Proyecto de otro grupo y detectar posibles errores, inconsistencias o fallos de funcionamiento. Como conclusión debe escribir un **Informe de Inspección** que se entregará al profesor y deberá ser enlazable desde el *backlog* del grupo auditado, junto con una versión "congelada" del Proyecto para pruebas ulteriores.

Atención porque el Informe puede tener consecuencias para los dos grupos implicados. Si un fallo grave es detectado e incluido en el Informe pero no es corregido por el grupo inspeccionado, este sufrirá una penalización adicional que en casos extremos puede implicar la anulación del Proyecto. Pero si un fallo grave no es detectado y se propaga a iteraciones posteriores la responsabilidad será compartida entre el grupo inspeccionado y el inspector.

El papel de controlador tras la tercera iteración lo hará el profesor. Cada grupo deberá realizar una **Defensa del Proyecto** ante la clase.

<table>
<tr><th></th><th>ITERACIÓN 1</th><th>ITERACIÓN 2</th><th>ITERACIÓN 3</th></tr>
<tr><td>Semana/s</td><td colspan="3">ENTREGABLES</td></tr>
<tr><td>1</td><td>Tabla de requisitos identificados para el proyecto</td><td></td><td></td></tr>
<tr><td>2,6,10</td><td colspan="3">Memoria de requisitos validados por el cliente</td></tr>
<tr><td>4,7,11</td><td colspan="3">Memoria de Diseño del Sistema</td></tr>
<tr><td>5,8,12</td><td colspan="3">Implementación del Sistema</td></tr>
<tr><td>2 – 12</td><td colspan="3">Backlog del Proyecto</td></tr>
<tr><td>4 y 7</td><td>Informe de Inspección de Proyecto A de otro grupo</td><td>Informe de Inspección de Proyecto B de otro grupo</td><td></td></tr>
<tr><td>14</td><td></td><td></td><td>Control Antidoping, Informe Final y Defensa del Proyecto</td></tr>
</table>

Tabla 1: Calendario de Entregables asociados a cada iteración

Finalmente tendréis que realizar un control para confirmar el nivel de exigibilidad individual adecuado en el desarrollo del Proyecto (**Control Antidoping**).

La tabla 1 sintetiza el calendario de entregables a presentar a lo largo del Proyecto. Se indica el número de semana dentro del periodo lectivo en el que se realizará la actividad y se generará el entregable asociado a la misma.

El backlog se considera un entregable continuo y puede ser utilizado en cualquier momento por el profesor, sobre todo para detectar que algún grupo pueda estar experimentando dificultades, pero también para ver si se sigue la metodología con aprovechamiento.

3. Evaluación

Lo primero de lo que tienes que ser consciente es de que el Proyecto es, con mucho, la actividad principal de la asignatura, tanto en Evaluación Continua como en la Ordinaria. Los pasos, entregables y plazos cambiarán, pero es imprescindible realizar el proyecto para superar la asignatura. En este documento nos centraremos (como hemos hecho hasta ahora) en las condiciones aplicables a la Evaluación Continua.

3.1. Valoración cuantitativa

En la tabla 2 se muestra la importancia cuantitativa que tiene el desarrollo del Proyecto en la evaluación de la asignatura, ya que su evaluación supondrá el 75% de la nota final. También es acorde con la dedicación en horas que habrá que dedicar a su desarrollo: un total de 119 horas de las 150 que corresponderían a una asignatura de 6 créditos.

PROYECTO	Presenciales	No presenciales	Total	% Nota
Primera Iteración	15,5	20	35,5	15
Segunda Iteración	10,5	16	26,5	25
Tercera Iteración	6	30	36	35
Finalización del Proyecto	3	18	21	
TOTAL PROYECTO	**35**	**84**	**119**	**75**

Tabla 2: Carga horaria para el alumno en actividades PBL y peso previsto en la evaluación.

Para aprobar la asignatura es un requisito que la aplicación resultado del proyecto tenga funcione correctamente en todas las iteraciones, y que proporcione una funcionalidad completa para los casos de uso implementados.

3.2. Ayudas a la auto y coevaluación: rúbricas y guías

Aquí se ha indicado el peso de cada iteración en la nota final, pero no debemos olvidar que en cada una de las tres notas se incorporan varios entregables. Al objeto de que puedas hacer estimaciones sobre la valoración que tendrán tus entregables se te proporcionarán **Rúbricas** en las que se explicarán los aspectos a valorar. Como ejemplo incluimos aquí dos utilizadas para valorar los informes de Requisitos y Diseño:

Rúbricas para los casos de uso

Nota: Los puntos máximos posibles (de acuerdo a donde se acomode mejor la entrada) se indican entre corchetes [ ].

**Hacer cosas extra (que están fuera de lo contemplado en las rúbricas) puede dar puntos extra.

	Escala			
Categoría	Bueno	Regular	Malo	Nulo
Estructura [40]	Los casos se reportan de acuerdo a la estructura planteada: nombre, descripción, actores involucrados y caso de uso relacionado (si aplica). **[40]**	Los casos reportan la mayoría de los campos solicitados. **[25]**	No se sigue la estructura en ningún aspecto; acaso se menciona el nombre del caso de uso. **[10]**	
Contenidos (caso individual) [40]	El caso representa claramente formas de utilizar el software (verbo + objeto directo); la descripción cumple con su objetivo. **[40]**	Los casos no son completamente entendibles; su nombre no se indica con verbo y objeto, pero la descripción clarifica de lo que trata el caso y esto es correcto. **[25]**	Los casos conforman palabras aisladas que la descripción no clarifica. **[10]**	No se subió la entrada. **[0]**
Completitud [15]	Se detectaron todos los casos pertinentes. **[15]**	Se detectaron la mayoría de los casos pertinentes (falta alguno que claramente	Faltan muchos casos importantes. **[5]**	

Figura 3: Rúbrica para la evaluación de los Casos de Uso

Rúbricas para el diseño con clases

	Escala			
Categoría	Bueno	Regular	Malo	Nulo
Contenido: clases (clase individual) [40]	Se reporta: el nombre de la clase, el rol que juega dentro del sistema, su visibilidad, sus atributos y sus métodos. El nombre de la clase es un sustantivo y empieza con mayúscula. La información reportada es detallada y coherente. Globalmente, se detectaron todas las clases importantes y para cada clase se detectaron todos los atributos y métodos importantes. **[40]**	Se reportan sólo algunos de los elementos solicitados. La información reportada no está lo suficientemente detallada y/o no es completamente coherente. El nombre de la clase es un sustantivo. Globalmente, falta alguna clase importante para el sistema y/o faltan atributos y métodos importantes para alguna clase. **[20]**	Básicamente no se reporta nada más que el nombre de la clase. La información reportada es incoherente y/o escueta. El nombre de la clase es un verbo o cualquier otro tipo de palabra. Globalmente, faltan muchas clases importantes para el sistema y/o faltaron atributos y métodos importantes para muchas clases. **[10]**	No se subió la entrada. **[0]**
Contenido: atributos (atributo individual) [25]	El nombre del atributo es un sustantivo o adjetivo y expresa claramente una propiedad de la clase. Se reporta su visibilidad	El nombre del atributo es un verbo u otro tipo de palabra, pero respeta la noción de propiedad. Se reporta su visibilidad	El nombre del atributo es un verbo u otro tipo de palabra. Se reporta su visibilidad como pública o no se	No se reportaron atributos. **[0]**

Figura 4: Rúbrica para la evaluación de los Diagramas de Clases

Ingeniería de Software I

Nombre del docente: _____

Número de equipo: _____

Rúbrica:
Diagramas de Secuencia
Fecha de creación: **17 de octubre de 2011**

A1: _____ A2: _____ A3: _____ A4: _____ (A1: Alumno1, A2: Alumno2, A3: Alumno3, A4: Alumno4)

CATEGORIA	2	1	0	A1	A2	A3	A4
Nombre y diseño	Los diagramas de secuencia incluyen su nombre y la distribución de los objetos, mensajes y activación es ordenada, lo que permite que sean claros de comprender.	Los diagramas de secuencia incluyen su nombre pero la distribución de los objetos, mensajes y activación es desordenada, lo que complica que sean claros de comprender.	Sólo algunos diagramas de secuencia incluyen su nombre y la distribución de los objetos, mensajes y activación es desordenada, lo que complica que sean claros de comprender.				
Objetos	Los diagramas de secuencia identifican perfectamente los objetos involucrados en cada uno de ellos.	Sólo algunos diagramas de secuencia identifican perfectamente los objetos involucrados en cada uno de ellos.	Los diagramas de secuencia no identifican claramente los objetos involucrados en cada uno de ellos.				
Línea de tiempo y Activación	La línea de tiempo de los diagramas de secuencia incluye la activación de cada uno de los objetos durante el lapso de tiempo que corresponde.	La activación no se incluye en la línea de tiempo de cada uno de los objetos durante el lapso que corresponde en los diagramas de secuencia.	No existe activación en la línea de tiempo de los objetos identificados en los diagramas de secuencia.				
Mensajes	Los mensajes agregados en los diagramas de secuencia, están colocados y numerados correctamente.	Los mensajes agregados en algunos de los diagramas de secuencia, están colocados y numerados correctamente.	Los mensajes agregados en los diagramas de secuencia no están colocados ni numerados correctamente.				
Número de diagramas	El equipo entregó de 10 a 13 diagramas de secuencia.	El equipo entregó de 6 a 9 diagramas de secuencia.	El equipo entregó menos de 6 diagramas de secuencia.				
			Total:				

Figura 5: Rúbrica para la evaluación de los Diagramas de Secuencia.

Por otro lado, para la realización de los Informes de Inspección los grupos recibirán un **Guión con su lista de comprobación**. Se valorará especialmente si el grupo es capaz de ampliar este documento basándose en su propia experiencia y reflexiones sobre el Proyecto.

4. Objetivos de aprendizaje:

Todo proyecto de ingeniería requiere de un proceso sistemático y riguroso si deseamos obtener un producto de calidad. Además tiene que venir acompañado por diferentes modelos y herramientas que permitan diseñar el producto de la manera más adecuada.

El objetivo de este Proyecto consiste en estudiar los procesos, modelos y herramientas más extendidos en el área de la Ingenieria del Software para el desarrollo de aplicaciones, y poner en práctica en un proyecto concreto. El objetivo del proyecto consiste en realizar una aplicación que siga los estándares actuales para el desarrollo de software. Un vez realizado el Proyecto, deberías ser capaz de:

- Distinguir y conocer las distintas etapas del desarrollo del software, y muy especialmente las de Análisis de Requisitos, Diseño, Implementación y Pruebas.
- Seguir con aprovechamiento dichas etapas (es decir utilizarlas de manera eficiente y creativa, como una ayuda y no como un corsé).
- Gestionar la evolución de un proyecto describiéndolo de forma que el equipo de trabajo comprenda en cada momento el punto en que se encuentra.
- Dominar los principios y disciplinas del Proceso Unificado del desarrollo de Software (UP).
- Manejar los conceptos y principales diagramas del Lenguaje Unificado de Modelado (UML) en su aplicación a la Ingeniería del Software.
- Identificar los requisitos de un sistema propuesto a partir de las especificaciones (QUÉ QUIERE) de un determinado cliente.
- Obtener un diseño a partir de los requisitos del sistema (CÓMO SE HACE).
- Diseñar y construir aplicaciones mediante arquitecturas de tres capas (presentación, lógica de negocio y gestión de datos), utilizando con criterio sus principios y dominando técnicas específicas para ello.

Además de estas competencias de tipo general el Proyecto está concebido para adquirir soltura en el desempeño de habilidades metodológicas concretas que sirven de soporte a las anteriores:

- Aplicar conceptos básicos de patrones en el diseño de una solución (GRASP).

- Utilizar con soltura Diagramas de Casos de Uso y Flujos de Eventos para modelar los requisitos capturados.
- Utilizar con soltura y precisión los Diagramas de Clases para construir un Modelo del Dominio que incluya todos los objetos necesarios para cumplir con las especificaciones del cliente, así como las utilidades necesarias para diseñar la aplicación.
- Utilizar con soltura Diagramas de Diseño para modelar la cooperación entre objetos de las diferentes clases.

Finalmente el Proyecto te permitirá adquirir experiencia en el manejo de herramientas concretas. Hemos seleccionado algunas que son fáciles de conseguir, instalar, usar y mantener, pero se puede fácilmente dar el salto a otras plataformas análogas que puedas necesitar en tu futuro académico o profesional:

- Manejar con soltura el uso del entorno de programación (ECLIPSE) para el desarrollo y pruebas de un sistema software.
- Manejar la herramienta de modelado StarUML para documentar y expresar de forma visual los resultados de las distintas fases del UP.
- Utilizar una infraestructura gráfica (AWT/SWING) para dotar a las aplicaciones de una interfaz de usuario orientada a las necesidades del usuario.
- Utilizar un un sistema de gestión de bases de datos (db4o) para implementar la capa de persistencia de una aplicación.
- Utilizar una infraestructura de ejecución distribuida (RMI) para dotar a las aplicaciones de una arquitectura cliente/servidor.
- Utilizar el lenguaje de programación Java para acometer todas las tareas de implementación necesarias en el proyecto.

Por tanto el proyecto cubre prácticamente todos los objetivos formativos de la asignatura de Ingeniería del Software.

5. Recursos y materiales

5.1. Equipamiento

Las reuniones de SCRUM se harán, en la medida de lo posible, en el laboratorio, a efectos de que los grupos podráis disponer de recursos online para buscar información y de soporte para actualizar el *backlog* y generar nuevos documentos. De todas formas sería muy positivo si cada equipo dispusiera al menos de un **Ordenador portátil** para traer regularmente a clase, puesto que en las sesiones de aula también se desarrollarán actividades para las que disponer de equipamiento puede ser muy beneficioso.

5.2. Textos disponibles en la biblioteca

CRAIG LARMAN: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall. Este texto está estructurado como un proyecto software, en forma de iteraciones. Contiene explicaciones muy intuitivas y sus capítulos, secciones y apartados son claros y breves. Aparte de los ejemplos sencillos desarrolla dos pequeños proyectos cuya descripción inicial conviene leer si se quieren entender los ejemplos de más enjundia. Se recomienda la tercera edición (2004), aunque desafortunadamente la traducción al castellano sólo ha llegado a la segunda.

ROGER S PRESSMAN: Software Engineering: A Practitioner's Approach, McGraw-Hill. Este libro, cuya primera edición es nada menos que de 1982, se ha ido reinventando a lo largo de los años y manteniéndose como referencia fundamental. Su cobertura es mucho más amplia que el programa de la asignatura, por lo que está recomendado únicamente como lectura de referencia. La 7ª edición es de 2010, y está disponible en inglés y castellano.

5.3. Recursos online

Recursos de IS de Roger Pressman http://www.rspa.com/spi/: El autor del libro de referencia fundamental mantiene una web muy rica en recursos, incluyendo especificaciones, ejemplos, componentes software, herramientas y *checklists*.

SCRUM http://geeks.ms/blogs/jorge/archive/2007/05/09/explicando-scrum-a-mi-abuela.aspx: En este blog tenéis una descripción clara y sencilla de todos los aspectos de SCRUM que nos interesan par el desarrollo del Proyecto.

Modelado ágil http://www.agilemodeling.com/: Esta completa web mantenida por **Scott Ambler**, uno de los padres del agilismo, cumple dos funciones. Por un lado tiene ejemplos de modelado usando UP, por lo que es útil para aprender a usar diagramas UML. Por otro está claramente orientada a las metodologías ágiles, por lo que incluye muchos consejos para evitar el sobremodelado y la pérdida de tiempo en exceso de documentación.

Desarrollo ágil http://www.proyectosagiles.org/: En esta web hay una descripción más detallada de diversas técnicas ágiles para desarrollo ágil, incluyendo el propio SCRUM pero también con técnicas para moderar las reuniones, estimar el coste de las tareas, etc...

Aforismos sobre IS http://www.multicians.org/thvv/proverbs.html: Se han dicho muchas cosas ingeniosas sobre nuestra disciplina. Para relajarse y sonreír un poco, pero también para fijar algunas buenas ideas.

5.4. Recursos de programación

En las siguientes direcciones puedes descargarte los recursos necesarios para instalarte las distintas herramientas de desarrollo, así como su documentación oficial.

Tutoriales de Java SE http://docs.oracle.com/javase/tutorial/index.html: Este recurso contiene muchísima información, muy clara y con ejemplos. Especialmente indicado cuando empiezas con una tecnología nueva, como puede ser el diseño de interfaces en AWT/SWING o la programación distribuida mediante RMI.

Entorno ECLIPSE http://www.eclipse.org/downloads/: Ya tendrás familiaridad con este entorno de desarrollo. Si necesitas instalarlo y tienes dudas de qué distribución elegir puedes probar con la versión **INDIGO** del Eclipse **IDE for Java Developers**,. La ayuda que acompaña al programa es muy completa, y el protal contiene tutoriales y otro material de ayuda.

StarUML http://staruml.sourceforge.net/en/download.php: Usaremos este programa para diseñar modelos en UML, fundamentalmente los Diagramas de Casos de Uso y los de Secuancia. Ten en cuenta que sólo funciona en Windows.

DB4O http://www.db4o.com/DownloadNow.aspx: Este sencillo SGBD orientado a objetos será una herramienta de uso cotidiano. El propio portal contiene manuales de referencia y un tutorial.