baliabideak
material de aprendizaje

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Aprendizaje Basado en Proyectos para la asignatura Ingeniería del Software

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

Cuaderno del estudiante

IKD baliabideak 6 (2013)

ÍNDICE

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?

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.

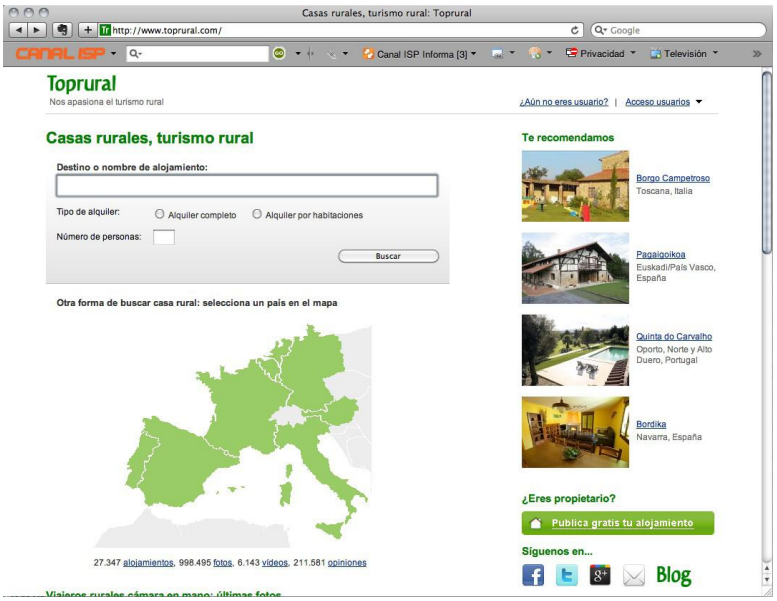
La razón parece estar en que en Villatripas de Abajo tienen una web estupenda que enseguida capta a los clientes potenciales dándoles información útil y atractiva y guiándoles en su elección. Sin embargo, la web de Villatripas de Arriba tiene enlaces muertos, no permite meter información individualizada 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.

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.

METODOLOGÍA Y DESARROLLO DEL PROYECTO

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.

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.

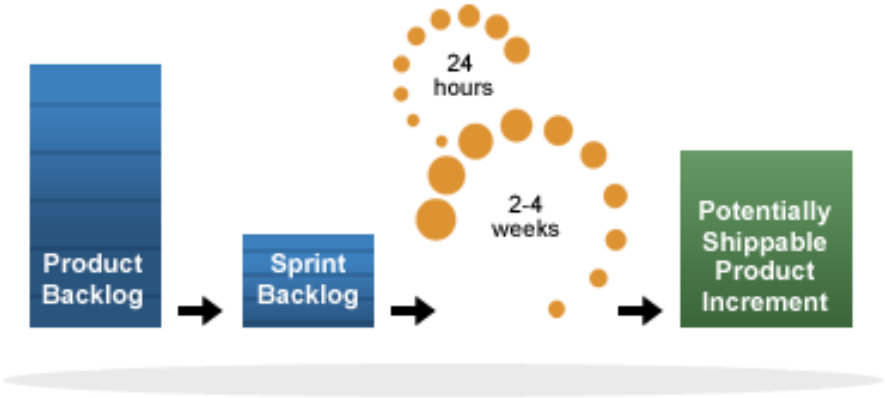


Figura 1: El proceso SCRUM (fuente http://pkab.wordpress.com/2008/07/11/scrum)

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?

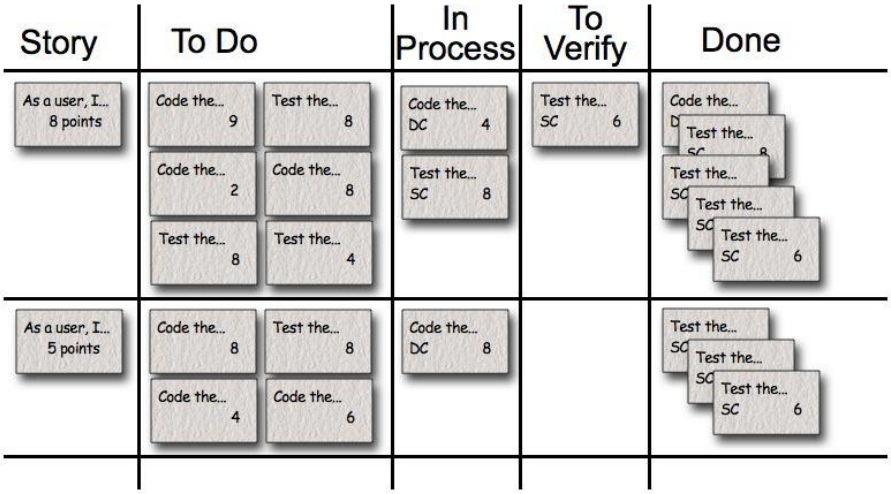


Figura 2:. Ejemplo de *backlog (fuente: http://www.mountaingoatsoftware.com/scrum/task-boards)*

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.

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)?

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.

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

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.

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.

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 funcione correctamente en todas las iteraciones, y que proporcione una funcionalidad completa para los casos de uso implementados.

Ayudas a la auto y coevaluación: rúbricas y guías

8. 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.

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.

RECURSOS Y MATERIALES

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.

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.

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.

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.

Goñi, Ibáñez, Iturrioz y Vadillo, I. (2013). Aprendizaje Basado en Proyectos para la asignatura Ingeniería del Software– IKD baliabideak 6 -http://cvb.ehu.es/ikd-baliabideak/ik/goni-sarriguren-06-2013-ik.pdf