Actas XVIII JENUI 2012, Ciudad Real, 10-13 de julio 2012
I.S.B.N. 10: 84-615-7157-6 | I.S.B.N. 13:978-84-615-7157-4
Páginas 25-32

Método para el aprendizaje de entornos y lenguajes de programación basado en prototipado ágil

Ramón Hervás Lucas
Departamento de Tecnologías y Sistemas de Información
Universidad de Castilla – La Mancha
Paseo de la Universidad s/n, 13071 Ciudad Real, España
Ramon.hlucas@uclm.es

Resumen

Este artículo describe un método de enseñanza para el aprendizaje de un lenguaje de programación y entornos de desarrollo asociados. Este procedimiento de aprendizaje está basado en el desarrollo de un proyecto de cierta complejidad a partir de prototipos incrementales de baja complejidad. El método se inspira en principios del proceso de Desarrollo Rápido de Aplicaciones y de la metodología Desarrollo Ágil de Software. Para alcanzar un efectivo aprendizaje se emplean técnicas como programación en pareja, integración continua, refactorización, etc.
Los alumnos, de tercer curso de ingeniería informática, cuentan con una base previa en metodología de la programación orientada a objetos y con un dominio medio de algún otro lenguaje de programación. Este método evita el estudio exhaustivo de los elementos y estructuras del lenguaje de programación, así como de su API. El aprendizaje se basa en el desarrollo guiado y supervisado de prototipos de complejidad incremental, quedando en manos de los grupos alumnos el perfeccionamiento de los prototipos y su integración en un proyecto único. En todo el proceso se introduce el uso de herramientas CASE para la definición y seguimiento de requisitos, el diseño rápido y su ágil desarrollo y prueba. La experiencia se ha desarrollado en la asignatura Herramientas y Entornos de programación durante 3 cursos en la Escuela Superior de Informática de Ciudad Real de las Universidad de Castilla-La Mancha.

Summary

This paper describes an educational method for programming languages and tools. The learning process is based on the development of a project integrating incremental simple prototypes. The method combines principles of rapid application development (RAD) and Agile Software Development methodology. Some learning techniques are used such as pair programming, continuous integration, and refactoring.
The students of computer engineering (undergraduates in their third-year), had knowledge in object-oriented methodology and they mastered some other programming language. This method avoids the exhaustive study of the elements and structures of the programming language and its APIs. The learning process is based on the supervised development of prototypes with incremental complexity. Students have to improve these prototypes and integrate them into a whole project. Throughout the process we introduce the use of CASE tools for defining and monitoring requirements, the rapid design and agile development and testing. The experience has been developed in the course Programming Tools and Environments during three years at the School of Computer Science in Ciudad Real (University of Castilla-La Mancha, Spain)

Palabras clave

Entornos y Lenguajes de Programación, Métodos de enseñanza-aprendizaje, Desarrollo ágil de aplicaciones, Desarrollo orientado a prototipos.

1. Introducción

La sociedad en general y la Informática en particular requieren que sus profesionales demuestren capacidades como iniciativa, creatividad, trabajo en equipo y competencias autodidácticas. Los paradigmas y métodos de aprendizaje tradicionales suponen limitaciones en el aprendizaje significativo y en el desarrollo de las capacidades anteriormente resaltadas.

Precisamente la puesta en marcha del Espacio Europeo de Educación Superior (EEES) propone mejoras en este sentido y está cambiando el modelo de enseñanza-aprendizaje. Se está dando protagonismo al aprendizaje autónomo, receptivo e innovador, frente a un método centrado en la enseñanza sin retroalimentación ejemplificado en las clásicas clases magistrales [1]. El modelo pasa de estar centrado en el emisor o profesor a estar centrado en el receptor o alumno .

El aprendizaje basado en proyectos (ABP), como metodología activa, puede facilitar un aprendizaje más dinámico, dónde el alumno se enfrenta a un problema cercano a lo que profesionalmente le espera y además desarrolla una forma de aprendizaje vicario y colaborativo mediante el trabajo en grupo.

Centrándonos en la enseñanza de la Informática y particularmente en la ingeniería del software y los lenguajes de programación, existen notorias diferencias en las metodologías tradicionales del mundo profesional con los intereses y objetivos en la enseñanza de lenguajes y entornos de programación. Estas metodologías tradicionales se centran fuertemente en un control exhaustivo del proceso, las actividades desarrolladas, roles involucrados, su planificación y los entregables o artefactos generados. La propia complejidad del proceso, no sólo puede condicionar el propio aprendizaje de las herramientas y lenguajes por falta de tiempo, sino que además puede limitar la creatividad y la consciencia de equipo al condicionar la toma de decisiones.

En este sentido las metodologías llamadas ágiles [2] se focalizan en el factor humano más que en el propio proceso. Este tipo de metodologías de desarrollo de software se realiza en iteraciones muy cortas y con gran interacción entre los involucrados (en nuestro caso los miembros de los grupos de alumnos entre ellos y con el propio profesor). Las metodologías ágiles han demostrado su eficacia en proyectos con requisitos cambiantes o con tiempo de desarrollo reducidos [3]. Estas dos características son propias también en el aprendizaje de nuevos entornos y lenguajes de programación, dónde los requisitos sufren continuos reajustes debido a la inexperiencia del alumnado y a que el tiempo es muy limitado. Las metodologías ágiles ya se ha planteado como técnica para el aprendizaje de la Informática, particularmente para Ingeniería del Software [3] y para programación [4] pero, a diferencia de esta propuesta, desde un punto de vista más conceptual y menos aplicado.

El presente artículo está organizado en seis secciones. Tras esta introducción, la segunda sección describe los fundamentos metodológicos del método docente presentado. El tercer capítulo describe el procedimiento propuesto y denominado *prototipado ágil*. El cuarto capítulo describe el contexto en el que se ha experimentado. Finalmente, la sección quinta analiza los resultados obtenidos y la sexta describe las conclusiones del trabajo.

2. Marco Metodológico

El presenta artículo describe un método de enseñanza-aprendizaje autodenominado prototipado ágil. Este método no pretende ser una técnica de desarrollo software y su finalidad es puramente el aprendizaje efectivo de lenguajes y entornos de programación. El método se inspira en diversas metodologías ágiles de ingeniería del software así como en el método de aprendizaje basado en proyectos. Más concretamente, los fundamentos de la propuesta son:

- Aprendizaje basado en Proyectos (APB): Se trata de un tipo de metodología activa [5] dónde se fomenta el trabajo grupal en la resolución de problemas reales. El APB pretende un aprendizaje significativo y multidisciplinar incentivando competencias como la actividad investigadora, la capacidad de autoaprendizaje, el trabajo cooperativo y la capacidad de autocrítica. Como metodología activa, es especialmente efectiva para introducir al alumno en su propio proceso de enseñanza [6], aprendiendo a aprender y favoreciendo la asimilación de contenidos.
- Metodologías ágiles de desarrollo: se trata de una serie de métodos que surgen como alternativa a los métodos tradicionales de desarrollo software basados en procesos. Estas metodologías se sustentan en el Manifiesto Ágil[1] que principalmente apuesta por el factor humano y las interacciones de equipo frente al proceso, el desarrollo funcional frente a una concisa documentación, la colaboración activa

[1] http://agilemanifesto.org

con el cliente y la respuesta y adaptación a los cambios frente a un plan estricto.

- Desarrollo rápido de aplicaciones (RAD): proceso de desarrollo software de carácter iterativo y orientado a prototipos incrementales. Propone el incremento de la productividad mediante el uso de herramientas CASE y entornos integrados. Pretende resolver problemas como la tardía obtención de resultados aptos para el cliente y el tiempo de desarrollo elevado.
- SCRUM: Se trata de una metodología de desarrollo ágil. Se organiza el proceso en etapas (*sprints)* de corta duración y la supervisión, análisis y toma de decisiones se realiza en base a lo obtenido en cada *sprint.*

3. Método propuesto: Prototipado Ágil

3.1 Descripción general del método

El método de prototipado ágil para enseñanza de entornos y lenguajes de programación tiene como eje central un aprendizaje basado en proyectos. Los alumnos se organizan en grupos reducidos (dos o tres personas) y tras un periodo corto de introducción a los conceptos fundamentales del lenguaje y entorno de programación, se inicia el desarrollo de un proyecto de complejidad considerable en relación a la didáctica y los conocimientos previos. El proyecto lo determinan los alumnos dentro de los objetivos de la asignatura. Este tipo de proyecto fomenta el autoaprendizaje de conceptos propios de la asignatura así como de competencias transversales. Sin embargo, la propia complejidad del proyecto y la inexperiencia podrían resultar contraproducentes si la curva de aprendizaje es demasiado pronunciada. Por esta razón, se incorporan al proceso aspectos del desarrollo ágil de aplicaciones. Concretamente, aunque se detallará en apartados posteriores, se establecen etapas que pueden ser de dos tipos:

- Etapas de prototipado: el alumno realiza en una sesión práctica un prototipo que resuelva una funcionalidad concreta a incorporar en el proyecto final, que además recoge y es parte fundamental de los contenidos de la asignatura.
- Etapas de seguimiento: cada ciertas etapas de prototipado, se realiza una defensa del estado del proyecto que permite la interacción profesor-alumno y que permite una reconsideración de los requisitos así como una detección temprana de errores o desviaciones en los objetivos prefijados.

En estas etapas interviene un concepto importante en el aprendizaje basado en proyectos, el concepto de entregable. Tantos las etapas de prototipado como las de seguimiento conllevan un entregable, centrado en aspectos funcionales, evitando invertir tiempo en una documentación exhaustiva.

3.2 Proceso detallado

Tomando la nomenclatura y principios de la metodología ágil de desarrollo software SCRUM, se definen los siguientes roles:

- Scrum master (supervisor): este rol lo encarna el equipo docente y sus principales responsabilidades son las de seguimiento y asesoramiento.
- Scrum Team (equipo): se trata del grupo de alumnos y su principal responsabilidad es el desarrollo del propio proyecto.
- Product Owner (cliente): en proyectos software profesionales se trata del cliente. En este método es un rol que representan alternativamente docente y alumnos ya que los requisitos del proyectos son un consenso entre los objetivos de la asignatura y el proyecto que han elegido desarrollar los alumnos.

Durante todo el proceso de desarrollo se generan una serie de prototipos que son de dos tipos:

- Prototipos inestables (PI): se desarrollan en sesiones presenciales con carácter de seminario práctico y su desarrollo está directamente orientado por el profesor. El objetivo es la enseñanza de fundamentos críticos del contenido de la asignatura. Estos modelos parciales y ejecutables no tienen porqué corresponder directamente con partes del proyecto final, su finalidad es favorecer el razonamiento sobre los modelos o componentes que se implementarán finalmente en el proyecto.

- Prototipos estables (PE): se trata de prototipos incrementales desarrollados a partir de los conocimientos obtenidos de los prototipos inestables y se construyen de forma incremental hasta obtener el proyecto final.

El proceso comienza con el lanzamiento del proyecto que consiste en unas directrices básicas sobre los objetivos esperados y su alcance. A partir de ahí, alumnos y profesor, alternando el rol de cliente, llegan a un consenso sobre los requisitos del proyecto, siempre desde una perspectiva flexible abierta a futuros cambios.

Las siguientes iteraciones alternan seminarios prácticos para el desarrollo de prototipos inestables y el desarrollo e integración de prototipos estables. Particularmente las iteraciones de prototipos inestables conllevan un entregable que servirá al profesor para comprobar la adecuada adquisición y asimilación de los contenidos fundamentales. Con cierta frecuencia, se propone una actividad de supervisión del estado del proyecto.

Finalmente, como última iteración, se produce la entrega y defensa del proyecto final. La Figura 1 muestra en detalle todo este proceso.

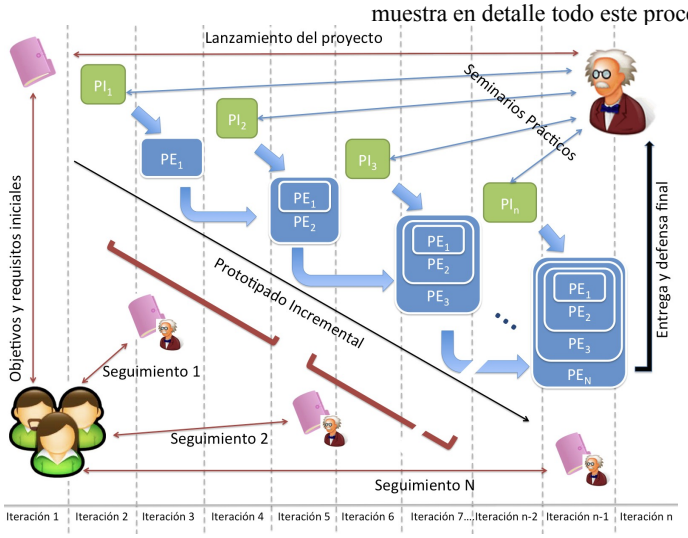


Figura 1. Proceso del método de prototipado ágil.

3.3 Metáforas y técnicas de trabajo.

En el desarrollo de este método de enseñanza aprendizaje se pueden resaltar una serie de metáforas y técnicas de trabajo de las que son partícipes los alumnos durante el proceso, muchas de ellas comunes a las diversas metodologías ágiles:

- Esfuerzo semanal: Se toma la semana como unidad de trabajo y los alumnos tienen claro qué se espera de ellos durante cada semana y el estado esperado del proyecto en todo el proceso. Aún así, siendo consciente de las oscilaciones habituales en las cargas de trabajo de los alumnos, es aconsejable cierta flexibilidad.
- Programación en parejas: Los alumnos desarrollan parte del proyecto de forma conjunta lo que incentiva el aprendizaje vicario, el análisis crítico y la reducción en la tasa de errores.
- Refactorización e integración continua: se trata de una actividad habitual en la integración progresiva de los prototipos al proyecto. Ello conlleva la restructuración y mejora del código implementado con anterioridad a la incorporación de nuevas funcionalidades.
- Desarrollo orientado a pruebas: Los alumnos conocen las pruebas unitarias que deben superar los prototipos en cada etapa del proceso.

- Principio de parsimonia: Se debe desarrollar la solución más simple que cumpla los requisitos, evitando añadir funcionalidades que nunca serán requeridas.
- Entregables pequeños: Los alumnos presentan los prototipos inestables con cierta frecuencia para fomentar la retroalimentación fluida alumno-profesor.

4. Experiencia docente

El método presentado se ha experimentado durante los tres últimos cursos, en la asignatura Herramientas y Entorno de Programación de la Ingeniería informática de las Escuela Superior de Informática de Ciudad Real perteneciente a la UCLM. Se trata de una asignatura de carácter optativo cuyo contenido práctico se centra en el aprendizaje de la tecnología .NET y el lenguaje de programación C#. Se fomenta además, el uso intensivo de herramientas CASE para aumentar la productividad. El número de alumnos matriculados en estos años ha sido respectivamente 43, 33 y 34.

El proyecto a desarrollar ha sido en todos los casos un juego a elección de los propios grupos de alumnos, grupos de entre 2 y 3 personas. Los requisitos iniciales que todo proyecto debía tener son el uso de interfaces de usuario avanzadas (usando tecnología WPF), acceso a base de datos y arquitectura cliente-servidor mediante el uso de *Sockets* para habilitar la funcionalidad multijugador.

Las etapas o *sprints* tienen una duración de una semana y se establecen un total de 16 etapas para completar el proceso, correspondientes a las semanas de un cuatrimestre. Tras la primera etapa de lanzamiento del proyecto y previo consenso de los requisitos, comienzan durante semanas alternas las etapas de prototipos inestables (sesiones presenciales prácticas) y el desarrollo e integración de los prototipos estables en el proyecto (trabajo autónomo). Sólo los primeros conllevan la entrega y supervisión del prototipo.

Sesión práctica	Sprint
PI1: Fundamentos lenguaje C# (Gestor de jugadores)	2
PI2: Fundamentos de WPF (Visor Multimedia)	4
PI3: Animaciones y dinamismo (Simon Dice)	6
PI4: Bases de Datos – ADO.NET (Trivial)	8
PI5: Controles de Usuario (Mascota Virtual)	10
PI6: Comunicación Cliente-Servidor (Chat)	12
PI7: Sistemas multi-hilos (Chat n-usuarios)	14

Tabla 1. Sesiones prácticas de la asignatura.

La Tabla 1 describe las distintas sesiones prácticas que corresponden a cada iteración de desarrollo de prototipos inestables. Cada uno de ellos aborda cada fundamento principal de la asignatura respecto a la Tecnología .NET.

Como se ha comentado en la descripción del método, cada cierto número de etapas se deber realizar un seguimiento del estado del proyecto. Concretamente se ha realizado cada cuatro etapas, lo que corresponde aproximadamente a una vez al mes. Para ello, los alumnos realizan breves presentaciones del estado de su proyecto y se debate con el conjunto de la clase la idoneidad de las decisiones tomadas. La Tabla 2 describe las tareas esperadas en cada una de las etapas.

Los entregables periódicos y el resto de actividades sólo se desarrollan y evalúan durante el desarrollo del curso. Los alumnos que no realizan o realizan parcialmente las actividades y se presentan a la convocatoria extraordinaria se centran en el prototipo final pero se les aconseja que sigan el método de prototipado ágil y hagan uso de las tutoría para su seguimiento.

Etapa	Tareas
1	Definición del juego y sus requisitos
2	Desarrollo y entrega del PI1
3	Desarrollo del prototipo estable 1
4	Desarrollo y entrega del PI2
5	Desarrollo e integración del PE2 Defensa del prototipo estable 2
6	Desarrollo y entrega del PI3
7	Desarrollo e integración del PE3
8	Desarrollo y entrega del PI4
9	Desarrollo e integración del PE4 Defensa del prototipo estable 4
10	Desarrollo y entrega del PI5
11	Desarrollo e integración del PE5
12	Desarrollo y entrega del prototipo inestable 6
13	Desarrollo e integración del PE6 Defensa del prototipo estable 6
14	Desarrollo y entrega del PI7
15	Desarrollo e integración del PI7
16	Entrega y Defensa Final

Tabla 2. Etapas semanales del proyecto

5. Resultados

Tras la experiencia de los últimos años en la aplicación de este método, podemos evaluar los resultados desde diversas perspectivas:

5.1 Calificaciones

La propia naturaleza del método de *prototipado ágil* exige al alumno un esfuerzo continuado en la asignatura. Aquellos alumnos que realizan las tareas esperadas en cada una de las iteraciones generalmente alcanzan holgadamente los objetivos de la asignatura. Además, las tareas de supervisión favorecen que se detecten de forma temprana desviaciones en los objetivos o incumplimientos de éstos. De forma general, los alumnos completan los prototipos inestables en la etapa correspondiente, pero algunos acumulan retraso en el desarrollo e integración de los prototipos estables. Como resultado un porcentaje reseñable de alumnos posponen la finalización del proyecto a la convocatoria extraordinaria. En el último año este efecto se ha reducido al penalizar en un 20% de la calificación la entrega del proyecto en la convocatoria extraordinaria.

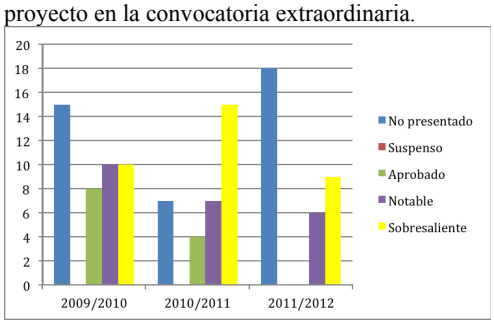


Figura 2. Calificaciones de los cursos 2009/10, 2010/11 y 2011/2012

La Figura 2 muestra los resultados en la calificación de la asignatura en los cursos 2009/2010, 2010/2011, 2011/2012. Cabe destacar que el curso 2011/2012 no está completado ya que resta la convocatoria extraordinaria, pero los resultados estimados, comparados con años anteriores indican una mejora en las calificaciones. Las notas medias de estos cursos son, cronológicamente, 7.61, 8.52 y 8.87

5.2 Proyectos desarrollados

Los proyectos desarrollados, en términos generales, cumplen adecuadamente los objetivos de la asignatura, así como alcanzan niveles de calidad reseñables en aspectos transversales como el diseño gráfico o la usabilidad. Sin embargo estos aspectos no han sido valorados formalmente más allá de la propia percepción del docente.

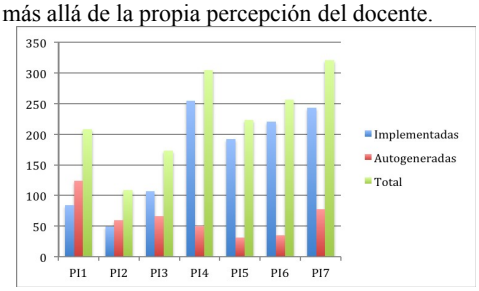


Figura 3. Líneas de código de los prototipos inestables

Una característica cuantificable es el número de líneas de código de los proyectos y su comparación con las líneas de código de los prototipos inestables desarrollados. Es una comparación significativa ya que los prototipos inestables corresponden a los fundamentos prácticos aportados por el docente y a partir de ellos es tarea de los alumnos asimilarlos, generalizarlos, ampliarlos e integrarlos en el proyecto. Concretamente las líneas de código del conjunto de prototipos inestables son 1595 líneas, de las cuales 1151 son implementadas y 444 son generadas por el entorno, correspondientes a las interfaces de usuario.

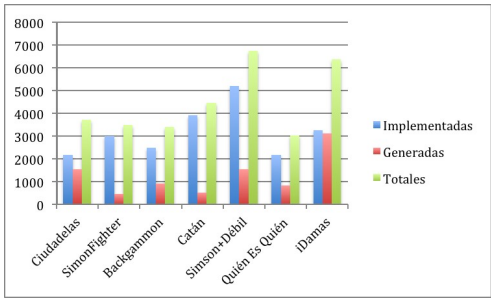


Figura 4. Líneas de código de los proyectos del curso 2011/2012

Centrándonos en los proyectos realizados en el curso 2011/2012, la media de líneas de código totales por proyecto es de 4460, de las cuales son

3279 implementadas y 1280 generadas por el entorno. La Figura 4 y el Anexo 1 contienen información detallada al respecto.

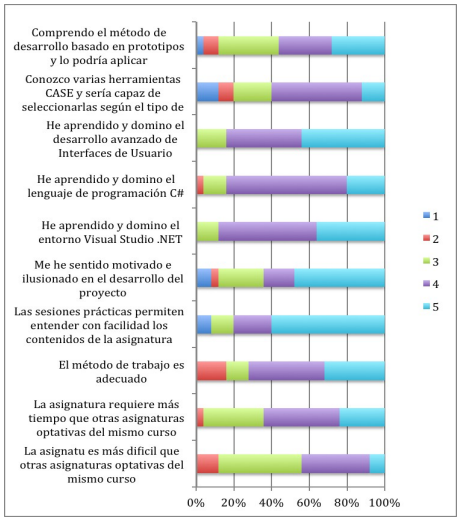


Figura 5. Cuestionario de valoración.

5.3 Cuestionarios

Al finalizar el periodo docente de la asignatura los alumnos han rellenado unos formularios de valoración consistente en 10 ítems. La evaluación se realiza utilizando una escala Likert de 5 puntos, correspondiendo el 5 a "absolutamente de acuerdo" y el 1 a "absolutamente en desacuerdo" con la afirmación. La figura 5 muestra los ítems y los resultados obtenidos. En términos generales resaltan lo interesante del proyecto y el método de desarrollo, la posibilidad de un mayor aprendizaje y el carácter práctico y cercano al desarrollo software profesional. Sin embargo, como punto negativo, señalan la necesidad de un gran esfuerzo para alcanzar los objetivos de cada etapa.

Por otro lado, ciñéndonos a las encuestas oficiales de la UCLM, la asignatura y su equipo docente, obtuvo en el curso 2010/2011 una valoración de 2.18 sobre 3, siendo la media de la carrera 1.82 y la media en la universidad 1.93.

6. Conclusiones

El método *prototipado ágil,* una vez experimentado y evaluado, fomenta el aprendizaje significativo y es acorde a los principios del EEES. Particularmente, fomenta el desarrollo de competencias clave como el trabajo en equipo, la autodidáctica y el análisis crítico. De forma más detallada podemos destacar que ayuda a alcanzar los siguientes objetivos:

- Objetivos conceptuales: Conocer metodologías ágiles de desarrollos, la autonomía en el aprendizaje de lenguajes y entornos de programación y el uso de herramientas de productividad.
- Objetivos procedimentales: planificación flexible de proyectos, desarrollo mediante prototipado incremental y trabajo en equipo
- Objetivos actitudinales: Planificación del esfuerzo, consciencia de la responsabilidad, capacidad de comunicación, habilidades de investigación y análisis crítico.

Los resultados obtenidos tras las distintas evaluaciones del método han sido muy satisfactorias. Sin embargo se han detectado deficiencias que deben ser abordadas en el futuro, principalmente que un número significativo del alumnado posponen la entrega a la convocatoria extraordinaria al no completar las etapas finales del proyecto, que corresponden a una mayor carga de trabajo en el conjunto de asignaturas cursadas.

7. Referencias

Prieto, N. et.al. *Uso de Metodologías Activas en la Implantación de IIP en el Grado en Informática de la UPV*. Jornadas de Enseñanza Universitaria de la Informática. Sevilla, 2011.

Beck, K. *Extreme Programming Explained. Embrace Change*. Pearson Education, 1999.

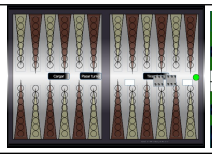
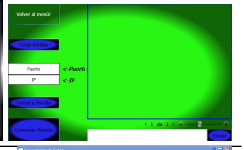
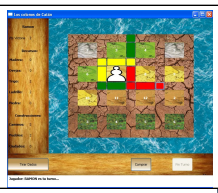
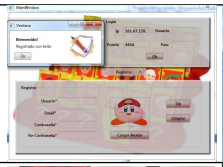
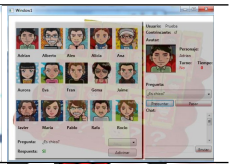
Pérez, J., Yagüe, A., Díaz, J., Alonso, S. *Un Primer Paso a la Agilidad: Retrospectivas para el Aprendizaje de la Ingeniería del SW.* Agile-Spain. Castellón, España, 2011.

Pardo, A., Estévez-Ayres, I., Basanta-Val, P., Fuentes-Lorenzo, D. *Programación en C con aprendizaje activo, evaluación continua y trabajo en equipo: caso de estudio*. Jornadas de Enseñanza Universitaria de la Informática. Santiago de Compostela, España, 2010.

Silberman, M. *Active Learning: 101 Strategies to teach any subject*. Allyn & Bacon, 1996.

Solomon G. *Proyect-based Learning: a Primer*. Technology and Learning, 23:6, 2003.

ANEXO 1: PROYECTOS DESARROLLADOS EN CURSO 2011/2012

Título	Alum-nos	Herramientas	Líneas de Código	Capturas
Ciudadelas	3	Visual Studio Expression Blend Requisite Pro MySQL Admin Visual UML	Implementadas: 2188 Generadas: 1541 Totales 3729	 
SSFighter	3	GatherSpace Ms Visio Visual Studio Expression Blend SandCastle	Implementadas: 3020 Generadas: 467 Totales 3487	 
Backgammon	3	Visual Studio Expression Blend Db4Free SandCastle RationalRose	Implementadas: 2482 Generadas: 918 Totales 3400	 
Colonos de Catán	2	Visual Studio Expression Blend SandCastle Gantter Umbrello Gliffy	Implementadas: 3923 Generadas: 535 Totales 4458	 
El Simpson más Débil	2	Visual Studio Expression Blend SandCastle GatherSpace MagicDraw	Implementadas: 5201 Generadas: 1553 Totales 6754	 
Quién es Quién	2	Visual Studio Expression Blend SandCastle MySQL-WorkBench Visual paradigm	Implementadas: 2186 Generadas: 843 Totales 3029	 
iDamas	2	Visual Studio Expression Blend SandCastle Gantter Visual Paradigm	Implementadas: 3257 Generadas: 3108 Totales 6365