Elaboración de material educativo para la formación de profesionales en desarrollo de software

Edgar E. Casasola

Universidad de Costa Rica, Escuela de Ciencias de la Computación e Informática
San Pedro, Costa Rica 2060
ecasasol@ecci.ucr.ac.cr

Abstract

This paper documents the first country wide experience related to collaboration between academy, industry and government, towards the development of educational material used to improve the quality of software development professionals. This effort is part of the educational component of the PROSOFTWARE project. The goal of the project is to strength the Costa Rican software development enterprises. This paper describes the context, background, methodology and results. It resumes the results obtained after finishing the first of a series of two courses on computer programming. This paper could be useful for those individuals or organizations interested on course design and implementation towards the education of well formed software development professionals.

Keywords: course planning, teaching and learning support environments, teaching methodology

Resumen

Este artículo documenta la primera experiencia a nivel nacional de colaboración entre academia, industria y gobierno asociada a la elaboración de un material educativo para el mejoramiento de la formación de profesionales en desarrollo de software. Este esfuerzo corresponde a resultados concretos del componente educativo del proyecto PRO-SOFTWARE. Proyecto creado con el fin de fortalecer a la industria de desarrollo de software en Costa Rica. Se describen el contexto y antecedentes del trabajo, la metodología de desarrollo, y la descripción de los resultados obtenidos durante la elaboración exitosa del primer material correspondiente a una serie de dos cursos de programación. Este artículo puede ser de interés para aquellos individuos u organizaciones interesados en la sistematización de cursos para la formación de profesionales idóneos para el desarrollo de software.

Palabras clave: planes de estudio, ambientes de apoyo a la enseñanza, metodología de enseñanza

1. Introducción

El presente trabajo es parte de los resultados obtenidos luego de desarrollar un material interinstitucional para impartir el primer curso de programación de computadoras. El material está diseñado para ser utilizado por los principales centros de enseñanza a nivel universitario en Costa Rica. Este curso es un producto directo del componente educativo del proyecto PROSOFTWARE, una iniciativa de la Cámara de Productores de Software de Costa Rica CAPROSOFT [4]. CAPROSOFT, fundada en 1998, es un consorcio sin fines de lucro formado por 60 compañías de desarrollo en este sector. Una de sus primeras iniciativas fue crear PROSOFTWARE, un proyecto conjunto entre la industria del software, el gobierno y las universidades para mejorar la competitividad de sus miembros mediante el mejoramiento de la calidad.

El proyecto PROSOFTWARE está constituido por tres componentes principales tal y como se puede observar en la Figura 1.

Este trabajo se ubica en el componente educacional del proyecto. Este componente persigue "la formación de profesionales idóneos", donde la idoneidad del recurso humano es vista como "la capacidad y habilidad de los graduados para desempeñarse de manera óptima de acuerdo con las necesidades y demandas del entorno" [7].

Dentro del contexto del componente educacional se llevaron a cabo tanto un Estudio de Oferta Demanda de Profesionales en el sector de Desarrollo de Software, como un Estudio para el Fortalecimiento de los Centros de Enseñanza y Actualización Curricular. El estudio de oferta y demanda de profesionales en el sector software se llevó a cabo entre octubre del 2000 y junio del 2001 con la participación de 97 empresas desarrolladoras de software de 124 identificadas, 53 organizaciones de otros sectores, y 123 profesionales. Y en octubre del 2003 se concluyó el Estudio de Perfiles Profesionales y Académicos. Este estudio, dividido en fases, implicaba la

identificación de perfiles de desempeño ocupacional, perfiles académicos-profesionales y finalmente la elaboración de recomendaciones curriculares con diseño de mallas curriculares propuestas según el perfil. [7]

Con el fin de promover aún más el fortalecimiento de los centros de enseñanza a nivel superior se procedió a identificar los cursos considerados medulares o cursos pertenecientes a un eje común para todos los perfiles. Se determinó que el primer curso de programación es fundamental para la formación de la mayoría de los profesionales en la industria del software y se tomó la decisión de contratar un coordinador de contenido para encargarse de la sistematización de este primer curso y coordinar esfuerzos entre profesores universitarios de todo el país.

Este documento presenta los resultados de la culminación exitosa del diseño del primer curso de Programación, el cual forma parte de una serie de dos. La versión oficial y validada del primer curso se utilizará oficialmente a partir de julio del 2004. Se espera contar con la primera versión del segundo curso para agosto del 2004.

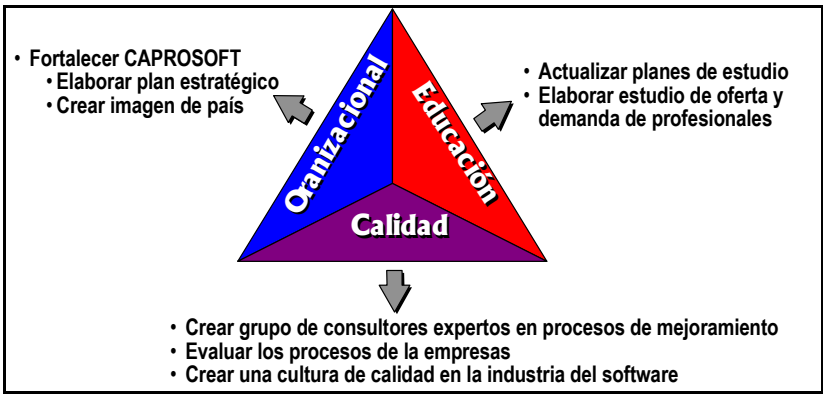

Figura 1. Los tres componentes principales del proyecto PRO-SOFTWARE [6].

2. *Metodología utilizada*

El trabajo de gestión de esta etapa del proyecto inició en paralelo con la presentación de los resultados preliminares obtenidos como parte de los estudios presentados por Mata y Matarrita [7]. Este trabajo de gestión se llevó a cabo ante los representantes de los centros de estudio participantes. En este momento se solicitó la participación de profesores y se propuso la estructura de trabajo que se muestra en la Figura 2.

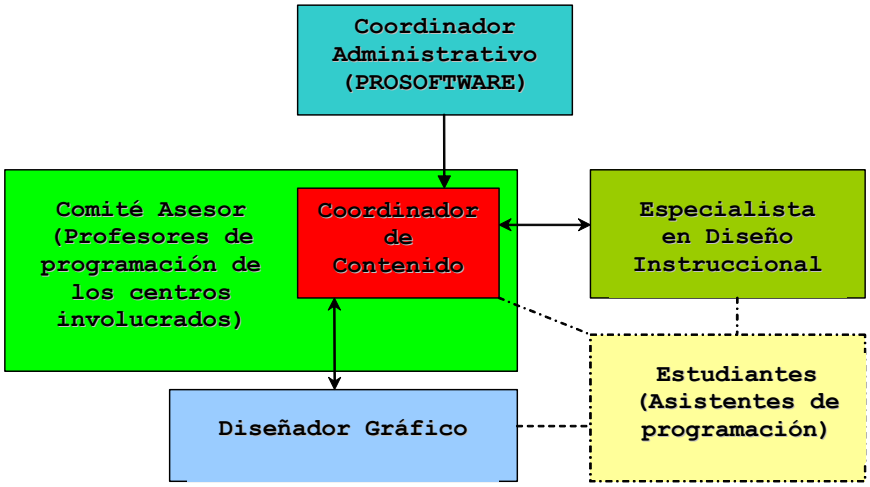

Figura 2. Conformación del equipo de trabajo.

Como se muestra en la Figura 2, el comité asesor estaría formado de profesores de programación representantes de cada centro, su trabajo estaría dirigido por el coordinador de contenido con el fin de definir las características deseables para un curso que fuera de utilidad para todos los involucrados. Todos los aspectos administrativos los manejaría un representante del proyecto Pro-Software dejando los aspectos relacionados al

contenido del curso bajo la responsabilidad del coordinador respectivo. El grupo de profesores fue entonces el generador de las características deseables del curso, utilizando los dos insumos básicos que fueron los planes y experiencia de los docentes de todos los centros y el conocimiento de los perfiles académicos y profesionales y de las recomendaciones curriculares propuestas en los estudios previos.

El trabajo se llevó a cabo bajo la modalidad de grupos focales, donde el coordinador se encargaba de propiciar una discusión que estimulaba a los participantes a compartir sus percepciones e ideas sobre un determinado tópico lo cual genera información a fondo sobre los temas en cuestión. Esta técnica propició motivar e identificar las características y necesidades o problemáticas comunes entre los diferentes centros.

La primera etapa del trabajo con el comité asesor fue la discusión y análisis de los planes de estudio actuales de los diferentes centros. Cabe mencionar que la principal preocupación se orientó hacia la selección del paradigma de programación y el lenguaje que se utilizarían.

En sesiones posteriores se discutió una propuesta de plan de curso y la discusión giró alrededor de los objetivos del mismo, contenidos, la modalidad de curso y las características que debía cumplir un material que fuera útil para todos los centros participantes.

Con el insumo anterior se preparó la primera versión del material y se llevó a cabo un taller de presentación, análisis y retroalimentación con el grupo asesor y un profesor invitado extra por centro. En este taller se resaltaron los aspectos positivos y negativos de la primera versión del material, y dichos insumos vinieron a propiciar la base para la elaboración de la versión 2 del material que incorpora todas las mejoras discutidas.

En la siguiente sección se presentan los resultados obtenidos con ejemplos concretos del curso implementado.

3. Resultados concretos

A. Análisis de los planes de estudio de los diferentes centros

En todos los centros se detectó alguna deficiencia relacionada con el diseño instruccional. En la mayoría de los centros: la sistematización de los cursos no era total, estaba orientada solamente a una lista de contenidos sin especificación del nivel al que se debía tratar cada tema, el perfil de salida del estudiante dependía en gran medida de la "interpretación" que haga el profesor del plan existente. En la mayoría de los casos simplemente se confiaba en la experiencia de los profesores asignados para impartir los cursos.

El primer resultado positivo fue resaltar la importancia de la sistematización de los cursos, y que esto llevaría a la minimización del esfuerzo en cuanto a planeamiento, programación de ejemplos y ejercicios. Por otra parte el aseguramiento de un nivel mínimo de calidad, y la posibilidad de que el estudiante avance a su propio ritmo.

Otro punto importante fue la definición de las habilidades fundamentales que se desea que el estudiante desarrolle para llegar a ser un buen profesional. Y la conclusión de que aspectos que no son centrales para el primer curso de programación pero que son importantes para la formación de un profesional en desarrollo de software deben incorporarse en forma temprana como "ejes transversales" cuyo abordaje se incrementará conforme el estudiante avance en su plan de estudios. Por ejemplo el conocimiento de la importancia de la comunicación oral y escrita, trabajo en grupo y la introducción de aspectos éticos relacionados al desarrollo de software se consideraron ejes transversales. Se definió la importancia de contar con pequeños proyectos elaborados en grupos de varios estudiantes con períodos de tres a cuatro semanas de duración, a los cuales se les denomina en el medio como "tareas programadas". El producto específico asociado a la calidad de la documentación, fue la definición de un estándar para la presentación de estas "tareas programadas" y el uso de generadores de documentación de código fuente como el javadoc [9].

B. Importancia de la selección del paradigma de programación

La importancia de la selección del primer paradigma de programación siempre fue tema de discusión, algo que se manifiesta en la gran cantidad de artículos existentes sobre el tema, por ejemplo estudios como el de Zhu, Haibin y Zhou [10]. Por lo tanto, la selección de paradigma de programación, y de los posibles lenguajes para la enseñanza del mismo fueron definidos desde el inicio.

Se tomaron en cuenta las recomendaciones curriculares del conjunto formado por la ACM, IEEE-CS y AIS conocido como La Fuerza Conjunta para el Currículo Computacional o "Joint Task Force for Computing Curricula 2004" [1] y los resultados del estudio de perfiles profesionales y académicos con que se contaba. Este último estudio vino a reforzar la importancia de contar con profesionales que dominaran el desarrollo de software orientado a objetos, modular, siguiendo patrones de diseño y preferiblemente en capas múltiples con interfaz Web. El sector software consideró importante el conocimiento en cuanto a las tecnologías emergentes en el mercado nacional incluyendo .NET de la corporación Microsoft [8] y J2EE de Sun Microsystems [9].

C. Definición de los objetivos del curso

El apoyo del diseñador curricular fue importante para la definición de los objetivos del curso. Se mencionó que el verbo utilizado para la redacción de cada objetivo llevaría implícito el nivel de aprendizaje esperado. Se incorporó al trabajo el uso de la una adaptación de la escala de Bloom [3], presentada en forma de pirámide y la cual se muestra en la Figura 3.

Esta escala indica que un objetivo con uno de los verbos colocados a un nivel de la pirámide implica el cumplimiento de objetivos en cada uno de los niveles inferiores. Por ejemplo el verbo "fundamentar" incluye aprendizaje asociado a los niveles inferiores por ejemplo (ser capaz de identificar, distinguir, ejemplificar y categorizar).

El plan del curso incluye la definición de un objetivo general y el detalle de los objetivos específicos. El objetivo general del curso quedó redactado de la siguiente forma:

> **Al terminar el curso el estudiante será capaz de resolver problemas simples mediante el diseño de algoritmos y desarrollo de programas, aplicando técnicas actuales de desarrollo de software orientado a objetos y considerando criterios de calidad apropiados.**

Como objetivos específicos se espera que al finalizar el curso el estudiante sea capaz de:

- Identificar problemas específicos.
- Representar problemas mediante la utilización de modelos abstractos.
- Comprender los elementos y estructuras básicas presentes en un lenguaje de programación orientado a objetos para implementar programas modulares, claros, simples y generales.
- Aplicar a nivel básico buenas prácticas de construcción de software tales como uso de estándares de documentación, codificación, verificación y validación para el aseguramiento de la calidad.
- Analizar problemas mediante un proceso de descomposición y refinamiento en pasos sucesivos.
- Sintetizar los resultados del proceso de análisis para diseñar algoritmos para la resolución de problemas.
- Aplicar una metodología de resolución de problemas que le permita trabajar con orden y disciplina.
- Descubrir la importancia del desarrollo de buenas prácticas de trabajo en equipo y de comunicación oral para la resolución de problemas.
- Aplicar técnicas apropiadas de comunicación escrita para la documentación de la solución de cada problema.

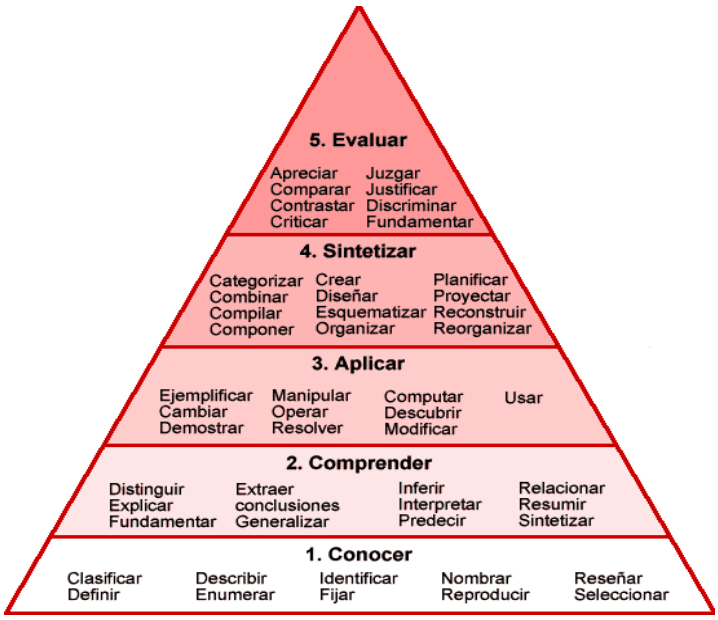

Figura 3. Diagrama que muestra los verbos utilizados para la definición de los objetivos de aprendizaje, organizados según la taxonomía de Bloom.

D. El lenguaje de programación

Java fue seleccionado por múltiples razones. Principalmente por ser un lenguaje independiente de la plataforma tecnológica de cada centro y cuyo uso no obliga a los centros a contar con licencias de alto costo. Otra

característica particular es la que algunos centros de estudio con programas de diplomado no cuentan con el tiempo suficiente dentro de sus programas para enseñar varios lenguajes de programación por lo que con Java pueden utilizar un lenguaje que también tiene utilidad para cursos de ingeniería de software y en el mercado laboral les permitirá el desarrollo de aplicaciones empresariales. Según se mencionó en el estudio previo se considera importante en este medio el conocimiento de tecnologías como la Edición Empresarial de Java 2 o J2EE [9] utilizadas para desarrollo de software en múltiples capas y aplicaciones Web. Se discutió que la mayoría de los estudiantes acostumbran trabajar en sus casas por lo que utilizar herramientas de alto costo implica obligar a los estudiantes a trasladarse a los centros para llevar acabo sus trabajos. Se consideró además que el uso de java evita que los estudiantes desarrollen el hábito de instalar software sin licencia en sus computadores de uso personal alegando que el costo de las licencias está fuera de su alcance, la formación puede ser integral si se presentan alternativas menos costosas para el estudiante.

E. Compilador y ambiente de desarrollo de programas

Al referirse al compilador y ambiente de desarrollo más recomendable se discutió la importancia de que el estudiante sea capaz de distinguir y diferenciar conceptos como lenguaje, compilador y herramienta de desarrollo. Además, que pueda distinguir las fases del proceso de escritura, compilación y ejecución de programas. Se mencionó la importancia de introducir la compilación mediante comandos para comprender los pasos que llevan a cabo las herramientas de desarrollo.

Al inicio del curso el estudiante aprende a distinguir las fases del proceso de creación, compilación y ejecución de un programa tal y como se muestra en la figura 4. En los primeros ejercicios se aprende el proceso de compilación y ejecución paso a paso mediante el uso de comandos como "javac" y "java" pertenecientes a las herramientas de desarrollo en Java o jdk de Sun Microsystems [9]. Por otra parte se introduce el uso de ambientes integrados de desarrollo como el Dr. Java [2] marcando la diferencia entre los conceptos "lenguaje", "compilador" y "ambiente integrado de desarrollo" o "IDE" según sus siglas en inglés. Dr.Java deja de lado toda funcionalidad innecesaria en un curso básico de programación.

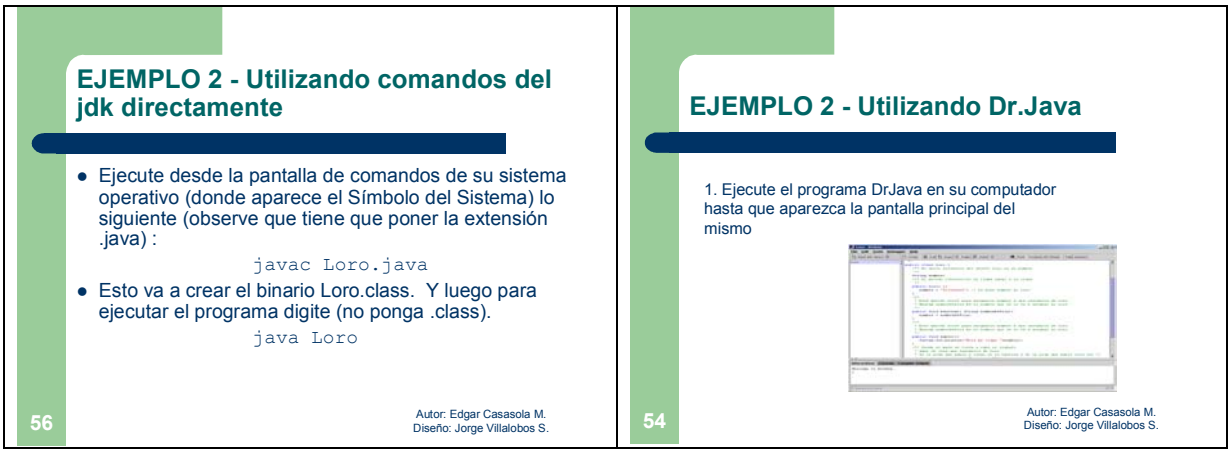

Figura 4. En los primeros ejemplos el estudiante se familiariza con diferentes formas de compilar y ejecutar un programa.

Otra característica importante fue la importancia de que se diera énfasis en los conceptos y no en el lenguaje seleccionado por lo que en todo el curso se mantiene una separación entre el contenido y la implementación en el lenguaje específico. Esto se manifiesta en el material con la separación explícita tal y como se muestra en la figura 5. Note que cada concepto presenta por separado los aspecto de implementación logrando que una posible actualización o expansión del material utilizando otros lenguajes se pueda dar de manera natural, facilitando el mantenimiento del curso.

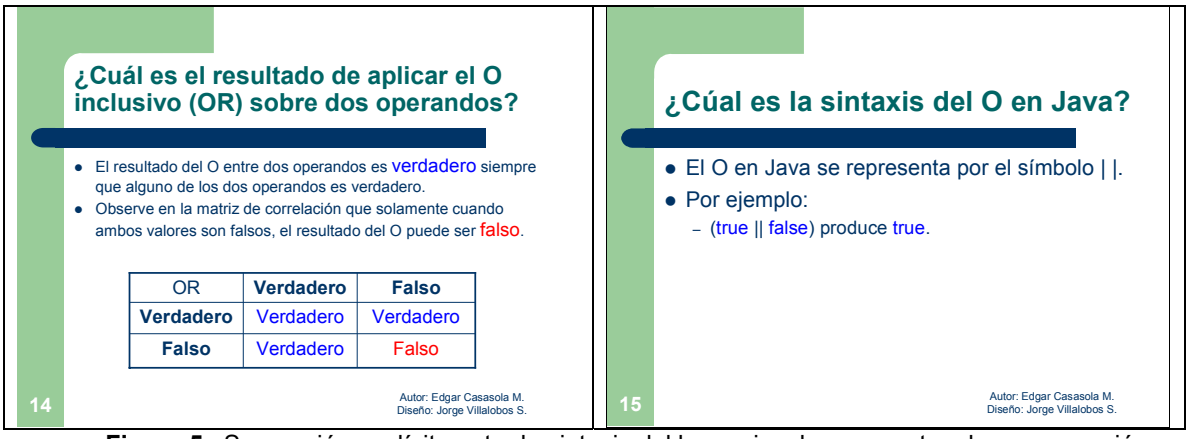

Figura 5. Separación explícita entre la sintaxis del lenguaje y los conceptos de programación en el material que presenta los contenidos del curso.

F. Aspectos curriculares asociados al contenido

Se discutió la importancia de no dar tanta importancia a la cantidad de contenidos como a las habilidades que se desarrollen en el estudiante, además de dar énfasis a la comprensión de los procesos generales llevándolos posteriormente a su representación sintáctica. En este caso se recurrió al uso de animaciones paso a paso como se muestra en la figura 6. Por ejemplo: Visualización de la memoria (Pila, memoria estática y el montículo principal o "Heap") durante la construcción y destrucción de objetos, mecánica de los llamados a métodos, recursividad, recorridos en vectores y recorridos en matrices.

Se discutió si la existencia de cursos con la modalidad de laboratorio para práctica dirigida era apropiado para los cursos de programación o si era más recomendable la creación de cursos bi-modales donde el estudiante pudiera experimentar a su propio ritmo fuera del aula y disponga del instructor para aclarar dudas y profundizar en la resolución de casos y llevando a cabo un rol de tutor que se encarga de aclarar conceptos. Fue importante la intervención del experto en diseño curricular quien fundamentó las virtudes de los modelos de autoaprendizaje, educación a distancia y educación asistida por computador. Por lo tanto se requería de un curso que tendiera más a enfoque menos presencial pero que permitiera el uso del material y contenidos en el aula.

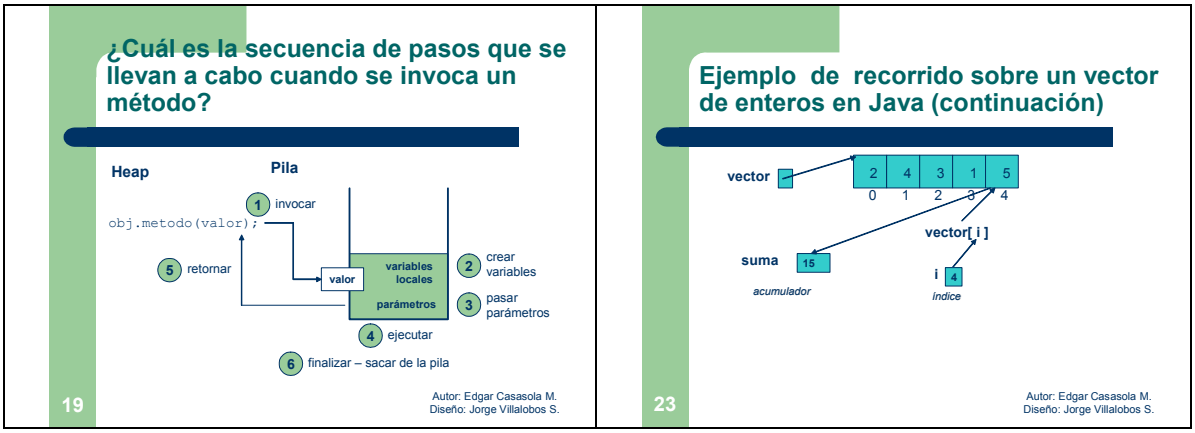

Figura 6. Fue necesario ilustrar procesos conceptualmente importantes mediante el uso de animaciones.

G. Estructura global del material y presentación del curso

Durante las discusiones con el grupo asesor se detectó una tendencia casi generalizada hacia el enfoque magistral y la idea del profesor como generador de conocimiento y el estudiante es un mero receptor de información y no como el eje central del proceso de aprendizaje. Se pensó entonces que para la transición hacia enfoques menos magisteriales el material podría utilizar un enfoque bi-modal con definición de horas presenciales y horas extra clase. El material debía ser suficientemente versátil como para poder ser utilizado en el aula como de manera individual por el estudiante.

El curso mantiene una estructura navegable y permite acceder directamente desde el árbol de la parte izquierda a las partes. El mismo fue desarrollado con una interfaz Web mediante el uso de html y javascript sobre

un modelo de datos reducido que mantuviera las mismas características de presentación a través de diferentes navegadores, lo cual unido al uso de Java ofrece la opción de ser multiplataforma e independiente de la tecnología utilizada en cada centro. Permitiendo su presentación y navegación tanto a través del Web como directamente desde un CD si así se desea.

La Figura 7 muestra en ejemplo de presentación de contenidos. Al lado izquierdo se presenta un árbol con la estructura del curso la cual permite navegar por los diferentes componentes del curso. Al lado derecho aparece un ejemplo de una pantalla de contenido. El estudiante puede repasar contenidos específicos a su propio ritmo. Y el docente puede maximizar la pantalla de presentación en clase. El formato de presentación es apto para ser utilizado en video conferencias.

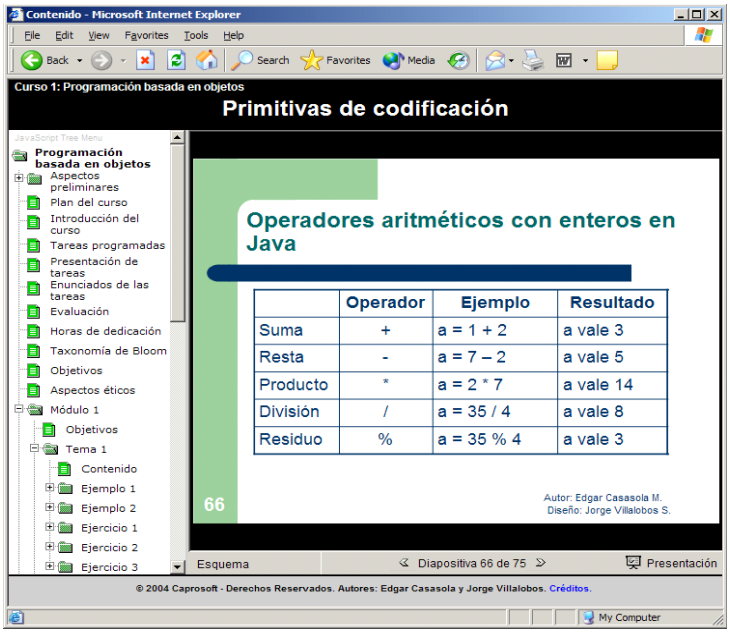

Figura 7. Presentación de los contenidos del curso.

De las sesiones de trabajo con el grupo asesor surgió la opinión generalizada de la necesidad de que el estudiante se mantenga siempre activo en cuanto a programación. El estudiante aprende mejor cuando siente que necesita del conocimiento para lograr algo, como por ejemplo resolver algún problema que le aqueja. De este modo se resaltó la figura de los proyectos o "tareas de programación" donde el estudiante se enfrenta a un problema que debe ser resuelto en grupo a lo largo de tres o cuatro semanas de trabajo y donde la solución del mismo no puede ser visualizada de forma inmediata como en ejercicios pequeños tendientes a reforzar conceptos. Esta modalidad se presta para que los estudiantes puedan desarrollar habilidades "transversales" como el trabajo en equipo y la comunicación oral y escrita, mediante la presentación de reportes y documentación de proyectos. Además permite llevar a cabo proyectos de mayor nivel supervisados por su instructor y es apta para todos los centros. El ejercicio en cuanto al proceso de identificación del problema, análisis del problema, diseño de una solución, implementación de la solución y prueba del mismo es reforzado a mayor escala durante el desarrollo de las tareas programadas.

A diferencia de otros cursos donde todo se centra en el código fuente, en este se presentan los ejemplos y ejercicios separados en las etapas del ciclo de resolución de un problema tal y como se muestra en la figura 8. El estudiante y el profesor cuentan con copias que se diferencian en las instrucciones incluidas. Además la versión del profesor tiene soluciones para todos los ejercicios incluyendo su análisis, diseño, implementación y posibles pruebas. En el caso del estudiante y dependiendo del enunciado del problema puede que en algunas de estas secciones lo que exista sean instrucciones que le sirvan de guía para llevar a cabo esa etapa en forma independiente. Ambas versiones permiten utilizar el material discusión en clases magistrales mediante la maximización de las presentaciones de contenido.

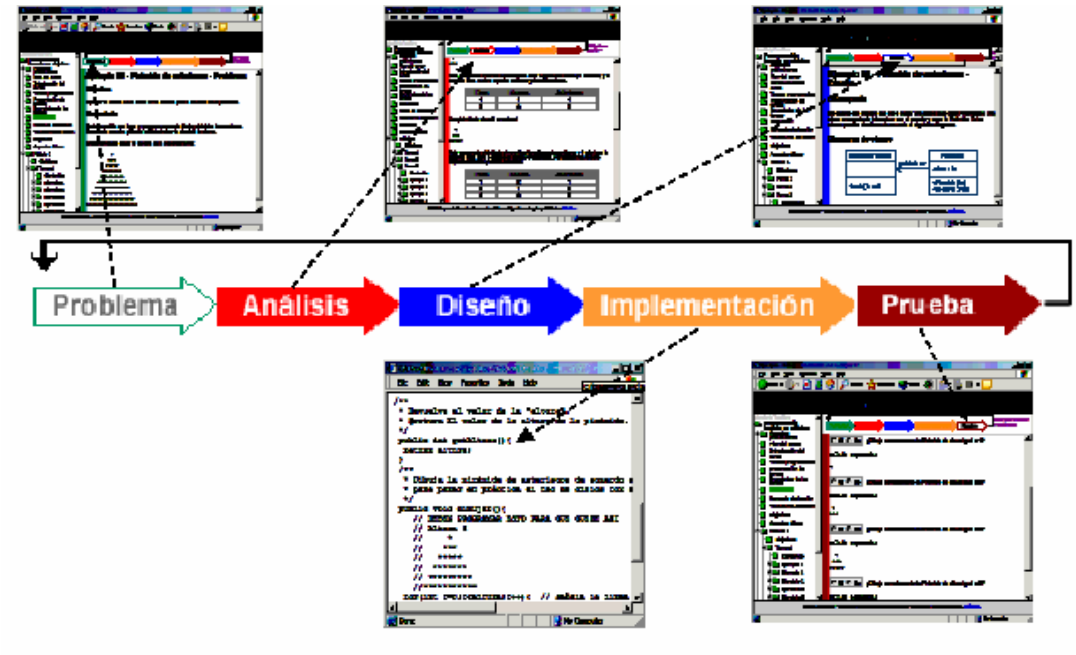

Figura 8. Ejemplos y ejercicios dan énfasis a la resolución de problemas.

De igual forma se hace énfasis en aspectos como la buena especificación y documentación de programas, y en la importancia del diseño de pruebas y el uso de listas de chequeo. Se quiere dejar claro en todo momento que aprender a programar no implica solo generar código, lo cual puede ser traumático para algunos si se trata de llevar a cabo de forma directa, sino ir de la especificación del problema hasta una solución probada siguiendo una metodología apropiada., donde en cada etapa hay un producto que sirve como insumo para la etapa siguiente.

H. Documentación de programas

Finalmente se mencionan dos aspectos o soluciones para el tema de calidad en el desarrollo de software. En este caso nos referimos a documentación y especificación de programas. En este caso la solución propuesta se centró en el la definición de un estándar simple para documentación de tareas y en el uso de especificación del código fuente mediante javadoc para llevar a cabo generación automática de documentación.

Un ejemplo de la documentación generada se puede visualizar en la Figura 9. En el curso se introduce el uso de comentarios dentro de los programas utilizando javadoc para la documentación apropiada de los programas.

I. Sistematización de criterios de evaluación

Por último es importante que tanto el estudiante como el docente tengan claro cuales son los aspectos que se evaluarán y la ponderación o importancia que se asigna a cada uno. La sistematización de este proceso se llevó a cabo mediante plantillas de evaluación con sus respectivas instrucciones de aplicación. Un ejemplo de plantilla de evaluación de tareas se muestra en la figura 9. Se sugieren algunas plantillas de evaluación de tareas cortas para que tanto el instructor como el estudiante tengan claro el énfasis que se da a cada aspecto de las soluciones planteadas. Las plantillas pretenden promover la sistematización de los mecanismos de evaluación con el fin de motivar y capacitar a los nuevos docentes en aspectos relacionados.

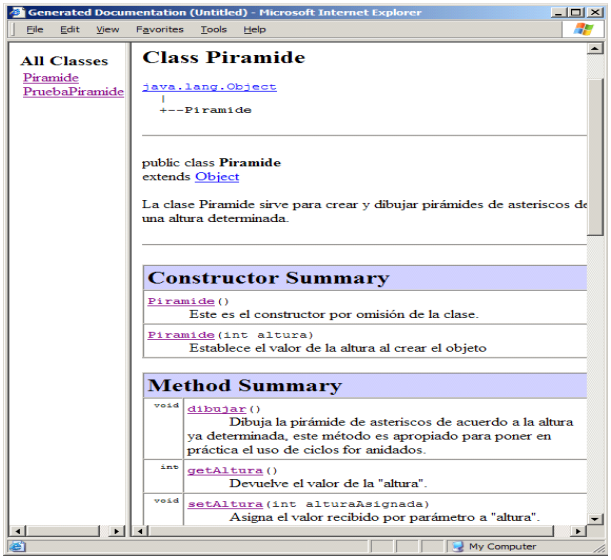

Figura 9. Uso de Javadoc.

4. Conclusiones

En Costa Rica es la primera vez que se sistematiza un curso para la formación de profesionales en desarrollo de software a nivel universitario con colaboración de todos los centros de enseñanza superior del país. El curso permite trazar la línea curricular originada en el estudio de oferta y demanda de profesionales para el desarrollo de software [7], hasta llegar a los contenidos, ejemplos y ejercicios del curso. Todos los ejemplos y ejercicios cuentan con objetivos explícitos que los relacionan con el contenido. El contenido satisface uno o más objetivos del curso lo cual demuestra una alta coherencia en el diseño instruccional del mismo. Todo esto facilita al estudiante y al docente para ubicarse y tener en todo momento claro: ¿que se quiere?, ¿para qué se quiere? y ¿cómo se quiere?

EVALUACION DE TAREA						
	MUY ALTO	ALTO	REGULAR	BAJO	MUY BAJO	NOTA
DOCUMENTACION INTERNA						
Tabulación y anidamiento correcto	8	6	4	2	0	
Nombres significativos	8	6	4	2	0	
Usa comentarios para especificar variables, métodos, parámetros y clases	8	6	4	2	0	
Consistencia en uso de convenciones de programación	6	4	2	1	0	
LOGICA DEL CODIGO FUENTE						
Claro	15	10	6	3	1	
Modular	15	10	6	3	1	
EJECUCION						
Compila y ejecuta sin errores de sintaxis	10	6	3	1	0	
Soluciona correctamente el problema planteado	15	10	6	3	1	
Casos de prueba apropiados	15	12	9	6	3	
FACTOR DE COMPLETITUD	x 1	x 0.8	x 0.6	x 0.4	x 0.2	
TOTAL						

Figura 10. Uso de plantillas de evaluación.

El diseño bi-modal del curso es apto tanto para estudiantes de los centros que utilizan un enfoque de enseñanza a distancia acompañado de tutorías, como para centros con enfoques presenciales. La portabilidad y compatibilidad del material es adecuada ya que no depende de plataformas comerciales para la enseñanza y para trasladarlo tanto en CD como el acceso vía Web. La versatilidad del material permite que el mismo sea utilizado para presentación de material y discusión bajo el enfoque presencial en el aula, como para auto aprendizaje por parte del estudiante.

El uso de las tareas programadas como disparador del trabajo a lo largo de todo el curso es importante para mantener al estudiante activo durante todo el proceso de aprendizaje y para que aplique e integre los conceptos puntuales vistos en el contenido. El curso logra enfocarse en aspecto como resolución de problemas más que enseñanza de un lenguaje de programación. En el mismo se separan claramente los conceptos de los aspectos de implementación en Java por lo que puede extenderse fácilmente con ejemplos en lenguajes como C++ o C# si así se deseara. Razón por la que la obsolescencia del curso será más lenta.

Se espera que el uso apropiado del material y la sistematización de la evaluación ayuden a asegurar un nivel de aprendizaje idóneo, y homogeneidad entre los estudiantes que aprueban el curso. Lo anterior facilitaría el proceso de articulación con los siguientes cursos de la carrera y finalmente formar profesionales idóneos para el medio en que se desempeñan.

5. Trabajo Futuro

Actualmente se está trabajando en la elaboración de un segundo curso de programación como continuación del primero para introducir aspectos como mecanismos de reutilización, programación por eventos, flujos de datos, hilos, y comunicaciones. Se plantea la necesidad de capacitar profesores en cuanto al uso apropiado del paradigma orientado a objetos y la necesidad de completar el esfuerzo mediante la creación de un curso compartido de estructuras de datos y análisis de algoritmos, otros dos de ingeniería de software y otro en diseño de bases de datos para completar el núcleo común. Los profesores participantes de cada centro han abierto un canal de colaboración que se planea utilizar para llevar a cabo al menos un taller anual de discusión para el mejoramiento continuo de la enseñanza de la programación, y se plantea contar con un foro de discusión continua entre centros. Actualmente se discute con los personeros de la Cámara de Productores de Software sobre la posibilidad de financiar un nuevo proyecto tendiente a evaluar a largo plazo el aprovechamiento por parte de los docentes al utilizar el material, y el impacto en el aprendizaje de los estudiantes.

Agradecimientos: MBa. Adolfo Cruz (Director ejecutivo Prosoftware), MSc. Eduardo Araya (Coordinador administrativo Prosoftware), Dr. Francisco Mata, profesores universitarios de las instituciones participantes (CENFOTEC, CUC, CUNA, CUP, ITCR, UNED, UCR, UCR, ULATINA, UNA), Bach. Jorge Villalobos (diseñador gráfico), MSc. Melvin Chaves (diseñador curricular), y a los estudiantes Adriana Blanco y Francisco Villegas.

6. Referencias

[1] Joint Task Force for Computing Curricula. **Computing Curricula 2004-Overview Report.** Association for Computing Machinery (ACM), Association for Information Systems (AIS), y Computer Society (IEEE-CS). 2004. http://www.acm.org/education/curricula.html

[2] Allen, E.; Cartwright, R.; y Stoler, B. DrJava: A lightweight pedagogic environment for Java. En: **33rd ACM Technical Symposium on Computer Science Education (SIGCSE 2002)**, Northern Kentucky, Cincinnati, USA, February 27 - March 3, 2002.

[3] Bloom, B. (1956). **Taxonomy of educational objectives. Handbook I. The cognitive domain**. New York, David McKay & Co., 1956.

[4] Caprosoft . **Estudio de Oferta y Demanda del Recurso Humano en el Sector Software Costarricense.** [página principal WWW], Visitada: Junio 2004. http://www.caprosoft.org/Caprosoft - Proyecto BID.htm

[5] Duke, R.; Salzman, E.; Burmeister,J.; Poon, J.; y Murray, L. Teaching programming to beginners - choosing the language is just the first step. En: **Proceedings of the Australasian Conference on Computing Education. Melbourne,** Australia, 2000.

[6] Jenkins, M. PRO-SOFTWARE: A Government-Industry-Academia Partnership that Worked. En: **17th Conference on Software Engineering Education and Training (CSEET'04).** Norfolk, Virginia. 01 – 03 March, pp. 92-97, 2003

[7] Mata, F.; y Matarrita, R.. **Conclusiones y Recomendaciones del Estudio para el Fortalecimiento de los Centros de Enseñanza y la Actualización Curricular.** CAPROSOFT. San José, Costa Rica. 2003 http://www.caprosoft.org/publicaciones.shtml

[8] Microsoft, **Página web**. Julio 2004. http://www.microsoft.com/

[9] Sun Microsystems. **Página web.** Julio 2004. http://www.sun.com/

[10] Zhu, H.; Zhou, M. Methodology first and language second: a way to teach object-oriented programming. En: **18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications.** Anaheim, California, USA, 2003.