METODOLOGÍA Y HERRAMIENTAS PARA EL APRENDIZAJE DE LA VERIFICACIÓN Y DERIVACIÓN FORMAL DE PROGRAMAS

Inés Jacob[1]

[1]*I.J.(Universidad de Deusto)*
e-mail: ines@eside.deusto.es

RESUMEN: En este artículo se presenta la experiencia en la enseñanza de la asignatura Metodología de la Programación en la Facultad de Ingeniería de la Universidad de Deusto. Los contenidos de esta asignatura tratan la verificación y derivación formal de programas. Se han diseñado actividades en grupo en el aula, se utiliza una red interna como medio para compartir recursos y se cuenta con el correo electrónico como herramienta de comunicación asíncrona.

Se está trabajando además en la elaboración automática de listados de enunciados de problemas y ejercicios resueltos. Se basa en la utilización de la tecnología XML que permitirá, entre otras cosas, que los materiales obtenidos puedan ser accedidos vía web.

1. OBJETIVOS DE LA ASIGNATURA

El planteamiento de objetivos viene orientado por el análisis del lugar que la asignatura ocupa en el plan de estudios en Ingeniería Técnica en Informática de Gestión y en las áreas de conocimiento. Los objetivos elegidos persiguen:

- Cubrir el contenido fijado en el plan de estudios para esta asignatura troncal,
- Completar el itinerario del alumno por las asignaturas del área de Programación y Lenguajes,
- Fijar y afianzar las bases para el aprendizaje de contenidos de disciplinas relacionadas y abordadas en parte en cursos posteriores.
- Dividimos los objetivos, por su nivel de formulación, en generales y específicos. Los objetivos generales son:
- Familiarizarse con la verificación y derivación formal de programas
- Razonar formalmente el diseño recursivo de funciones
- Entender la recursividad y su relación con los programas iterativos

Los objetivos específicos se han elegido en función de los objetivos generales intentando que el alumno vea en ellos la identificación de los logros que ha de conseguir a lo largo de la asignatura. Coinciden además con los aspectos que serán evaluados, lo que facilita al estudiante la identificación de las áreas en las que debe detenerse y al profesor la calificación.

Se enumeran a continuación los objetivos específicos asociados a cada objetivo general (OG). Algunos objetivos pueden ser considerados medios (M) por ser necesarios para la consecución de otros considerados terminales (T).

OG1: Familiarizarse con la verificación y derivación formal de programas

- Especificar formalmente el comportamiento esperado de un programa (M)
- Verificar y derivar las instrucciones: nula, composición secuencial, asignación, alternativa, repetitiva y llamada a función. (M)
- Distinguir los conceptos implicados en la derivación y verificación de programas (M)
- Identificar la necesidad de realizar inmersiones (T)
- Diseñar llamadas iniciales a las funciones inmersoras (T)
- Mejorar la eficiencia de los programas (T)

OG2: Razonar formalmente el diseño recursivo de funciones

- Demostrar que los programas iterativos y recursivos terminan (M)
- Elegir casos base y recursivos (M)
- Derivar programas recursivos (T)
- Obtener recursividad final (T)

OG3: Entender la recursividad y su relación con los programas iterativos

- Pasar a postcondición constante (M)
- Transformar programas recursivos en iterativos (T)
- Derivar programas iterativos (T)

2. CARACTERÍSTICAS DE LA ASIGNATURA

Metodología de la Programación se imparte en la Universidad de Deusto en segundo de Ingeniería Técnica de Informática de Gestión. Se basa en los conocimientos de lógica formal y de programación que el alumno adquiere en primero. El alumno se supone que sabe programar, aunque de forma algo intuitiva, y comprueba si el programa obtenido es "correcto" haciéndolo funcionar (de forma simulada o real).
Los métodos de verificación y derivación formal se basan en el análisis estático de los programas. No se "ve" si los programas funcionan, sino que se "demuestra" matemáticamente que así es.
Algunas características de esta asignatura que la hacen diferente a otras de la misma área son:

- Los programas obtenidos no pueden probarse por ejecución (aunque podrían transformarse para que así fuera no es claro que sea conveniente hacerlo).
- Contradice en cierta forma el modo en el que los estudiantes han aprendido a programar en su primer año de carrera.
- Los métodos que se explican no se aplican habitualmente.
- Exige gran rigor, pulcritud, exactitud, claridad y orden.
- Los temas que componen el programa de la asignatura se apoyan unos sobre otros, de forma que es casi imprescindible que el alumno lleve la materia al día.

3. METODOLOGÍA Y MATERIALES

El diseño de la actividad docente se ve muy afectada por el tamaño de los grupos a los que se imparte la asignatura (más de 100 alumnos por grupo). Las diferentes actividades planteadas pretenden hacer posible la participación activa de los alumnos en el proceso de enseñanza-aprendizaje de forma que tanto el alumno como el profesor tengan ocasión de comprobar los avances realizados y ser conscientes de los errores que se cometen.

Se da especial importancia a potenciar el aprendizaje significativo de los contenidos evitando estudiar de memoria. Por ello se intenta reducir el tiempo dedicado a enunciar contenidos aumentando el dedicado a la experimentación.

a) Lecciones magistrales

En estas sesiones se trata de explicar los conceptos básicos en cada uno de los capítulos y exponer los procedimientos de aplicación de las diferentes técnicas. La explicación se ilustra con la realización de ejemplos y ejercicios. Para que haya cambios de ritmo que faciliten el mantenimiento de la atención del alumno se intenta que la exposición del profesor no ocupe toda la sesión. Algunos autores [3] recomiendan que su duración no supere nunca el 65% de la de la sesión completa. Normalmente esto se consigue solicitando la participación de los alumnos en la realización de ejercicios. Se logra además que

- el alumno ponga en práctica las técnicas aprendidas
- el alumno pueda autoevaluar su evolución
- la clase sea más participativa
- el profesor pueda observar el nivel de aprendizaje de los alumnos, no sólo en cuanto a resultado obtenido sino en cuanto a forma de proceder

En algunas ocasiones los ejercicios propuestos son los aparecidos en exámenes de cursos anteriores. Esto les permite por una parte conocer el tipn de examen con el que serán evaluados y por otra, autoevaluar su proceso de aprendizaje.

Con el fin de facilitar la motivación intrínseca del alumno se le proporcionan apuntes elaborados por el profesor en los que se recoge la explicación de los diferentes capítulos siguiendo la estructura de las clases magistrales impartidas. Los contenidos de estos apuntes pueden ser completados con el texto base de la asignatura y con el resto de la bibliografía recomendada al alumno. Al no tener que ocuparse de tomar apuntes pueden centrarse más en las explicaciones del profesor.

b) Grupos

Las actividades en grupo suponen una alternativa a la tradicional clase magistral. La experiencia con este tipo de actividades nos hace llegar a la conclusión de que el factor sorpresa/novedad es importante, aunque es difícil mantenerlo cuando la actividad se repite en más de un grupo. Esta repetición se hace recomendable por motivos de uniformidad en los diferentes grupos de un mismo curso y asignatura.

Otras ventajas de emplear esta actividad son:

- facilita la autoevaluación del alumno contrastada con otros compañeros
- puede ser un instrumento de evaluación continua a tener en cuenta: el número de trabajos a corregir sería notablemente inferior que con actividades individuales.

- permite al profesor observar a los alumnos en su forma de proceder
- facilita el diálogo entre el profesor y los alumnos.

Para esta asignatura se han diseñado tres actividades definiendo objetivo, materiales necesarios, composición de los grupos, sistema de puesta en común, organización y programación.

La primera actividad grupal pretende que los alumnos entiendan *la utilidad de conocer métodos de programación distintos a los aprendidos en cursos previos*. Sobre un programa sencillo en Pascal del que no se indica qué debe hacer, no tiene comentarios y además tiene un comportamiento erróneo, se plantean una serie de cuestiones a trabajar en grupo por los alumnos: ¿está bien el programa?; ¿qué habéis hecho para averiguarlo?, recoged todas las estrategias seguidas por los integrantes del grupo; si el programa no está bien indicad las razones y cómo lo modificaríais para mejorarlo; escribid el programa que obtenéis entre todos. En la puesta en común oral se recogen las aportaciones de los grupos a través del moderador de cada uno de ellos.

La segunda actividad grupal tiene como objetivo que los alumnos *practiquen la sistematización de lo aprendido y desarrollen la capacidad de extraer las ideas fundamentales*. Se distribuye entre los alumnos un documento con el enunciado para la derivación de una función recursiva y los pasos a dar para resolver el ejercicio. Se pide determinar en qué orden pueden darse estos pasos para derivar una función correcta. Puede haber varias soluciones válidas.

Por último, en la tercera actividad, se solicita a los alumnos la preparación de su propia autoevaluación. Consiste en diseñar un test con las preguntas que los alumnos propongan. Cada grupo elige 4 preguntas de test de las 16 aportadas por sus integrantes, asegurándose de que propone una respuesta correcta y dos incorrectas para cada una de las preguntas. Es importante que las respuestas incorrectas sean funcionales [5], es decir, sean elegidas por más del 5% de la muestra y tengan una correlación negativa con el total.

La puesta en común de esta actividad en grupo consiste en la realización de un test formado por las preguntas propuestas por los alumnos. Las preguntas acertadas suman 1 punto, las falladas restan medio punto y las que se dejan en blanco no tienen puntuación. Este sistema de puntuación se corresponde con el empleado en el examen final y trata de evitar que los alumnos intenten dar la respuesta correcta por adivinación [6].

Las actividades grupales realizadas en esta asignatura se plantean con fines de autoevaluación aunque se podría considerar la posibilidad de tomar en cuenta la puntuación obtenida por el alumno en cada una de ellas matizándola con el resultado del grupo al que pertenece.

c) Utilización de la red interna y el correo electrónico

Los alumnos son atendidos en tutorías presenciales. Estas tutorías están programadas tres horas a la semana. Pueden acudir sin cita previa. También pueden realizar consultas por correo electrónico lo que permite la comunicación asíncrona entre profesor y alumnos.

Este medio también se utiliza para recopilar materiales elaborados por los propios alumnos para ser compartidos con sus compañeros. Este tipo de materiales se ponen a disposición de los alumnos en la red interna de la Facultad.

El profesor deposita también materiales que considera de interés para los alumnos por servir de ayuda para el estudio personal o para trabajar en el aula.

GENERACIÓN AUTOMÁTICA DE EJERCICIOS

Una de las dificultades para el aprendizaje de la verificación y derivación formal de programas y, en concreto, de la asignatura Metodología de la Programación, es la escasez de ejercicios y problemas básicos. Los contenidos de la asignatura son fundamentalmente procedimentales y su aprendizaje se basa en la práctica.

Estamos trabajando en un proyecto para la extracción automática de enunciados de problemas y ejercicios resueltos contenidos en documentos en formato electrónico. Estos documentos recogen ejercicios completamente desarrollados en los que se han aplicado todas las técnicas y procedimientos que constituyen el contenido de la asignatura.

Este trabajo se basa en los resultados preliminares de un proyecto de fin de carrera de Ingeniería en Informática [1].

a) Planteamiento de objetivos

Partiendo de ejercicios completos dispuestos en documentos Word se extraen diferentes listados de ejercicios adecuados para la práctica de las técnicas estudiadas en la asignatura. Llamamos *ejercicio completo* al que partiendo de una especificación de función en lenguaje natural incorpora obtención de la especificación en lenguaje formal, solución recursiva del programa, inmersiones de eficiencia, transformación a recursividad final, obtención de verificación a postcondición constante, paso a iterativo y solución iterativa del programa.

Un programa así desarrollado, además de contener ejemplos de aplicación de estas técnicas, presenta ejemplos de derivación de instrucciones simples como asignaciones, composiciones secuenciales, llamadas a función, alternativas y repetitivas.

El tratamiento adecuado del documento que contiene este ejercicio completo, al que a partir de ahora llamaremos *documento fuente*, permitirá obtener para cada etapa del proceso de aprendizaje relaciones de ejemplos y propuestas de ejercicios. La extracción manual de estos ejemplos y ejercicios resueltos es muy costosa por lo que este proyecto pretende automatizar el proceso.

b) Fases principales del proceso automatizado

La metodología contempla tres fases de procesamiento: (a) transformación del documento fuente en texto puro; (b) etiquetado en XML del texto puro; y (c) diseño e implementación de filtros en XSL.

Transformación del documento fuente en texto puro. La edición de los ejemplos completos se realizó en Word por comodidad del profesor que los preparó. La nomenclatura utilizada se corresponde con el pseudocódigo propuesto por Balcázar [2]. En esta primera fase se transforma el documento fuente en texto puro, tratando adecuadamente los símbolos especiales ($\forall$, $\exists$, α, $\geq$, $\rightarrow$, etc.) para que puedan seguir siendo reconocidos.

Etiquetado del documento en XML. Optamos por el lenguaje XML (Extensible Markup Language) [4] que permite diseñar un conjunto de etiquetas adecuado a la estructura del documento tratado. Además será posible presentar en páginas web la información obtenida. El repertorio de etiquetas permitidas y la estructura del documento quedan recogidas en la DTD (Document Type Definition). La DTD presenta la gramática del pseudocódigo empleado en la

codificación de los programas, lo que permite validar su sintaxis. En esta fase se comprueba que el documento se adapta a la gramática del lenguaje y se etiquetan todos sus componentes.

Diseño e implementación de filtros en XSL. Una vez que el documento ha sido convenientemente etiquetado queda explotarlo de acuerdo a las necesidades de los alumnos. Esto se hace mediante la aplicación de filtros implementados en XSL (Extensible Stylesheet Language). Este lenguaje permite crear hojas de estilo que ofrecen visiones diferentes de un documento XML. Se desarrolla por lo tanto una XSL por cada uno de los diferentes listados a obtener: de asignaciones, de composiciones, secuenciales, etc. Cada uno de estos listados podría obtenerse en dos versiones: enunciados y ejercicios resueltos. En el caso de utilizar navegadores que no permitan la visualización de documentos XML se podría diseñar e implementar un filtro adicional que obtuviese documentos en HTML.

5. CONCLUSIONES

Hemos presentado en este artículo las acciones de mejora para una asignatura del plan de estudios conducente a la obtención del título de Ingeniero Técnico en Informática de Gestión. Se trata de un proceso continuo que pretende mejorar el aprendizaje de los alumnos mediante la incorporación de metodologías variadas y la utilización de tecnologías actuales.

La metodología diseñada para obtener automáticamente listados de ejercicios puede aplicarse a asignaturas en las que las técnicas básicas aprendidas se aplican ejercicios completos siempre y cuando la resolución de los ejercicios está muy estructurada de forma que puede definirse su gramática.

El material didáctico obtenido puede ser visualizado por navegadores de Internet con capacidad para tratar documentos XML mediante hojas de estilos XSL (como MSExplorer5.x y Netescape6.x).

El profesor simplemente deberá desarrollar ejercicios completos dejando que la elaboración de propuestas de ejercicios y ejemplos adecuados a cada etapa del proceso de aprendizaje se automatice.

BIBLIOGRAFÍA

[1] I. Azofra y C. de Prado, 1999. *Etiquetado automático de documentos para la extracción de información*. Proyecto fin de carrera. ESIDE, Universidad de Deusto.
[2] J.L. Balcázar, 1993. *Programación metódica*. McGraw-Hill.
[3] Dide, 1999. *La exposición como técnica didáctica*. Dirección de Investigación y Desarrollo Educativo de la Vicerrectoría Académica del Instituto Tecnológico y de Estudios Superiores de Monterrey. **http://www.sistema.itesm.mx/va/dide/inf-doc/estrategias/exposicion.html**
[4] C. Goldfarb y P. Prescod, 1999. *Manual de XML*. Prentice-Hall.
[5] T.M. Haladyna y S.M. Downing, 1988. *Functional Distractors: Implications for Test-Item and Test Design,* ponencia presentada en el congreso anual de la American Educational Research Association, New Orleans.
[6] P. Morales, 1995. *Las pruebas objetivas*. Cuadernos monográficos del ICE, Universidad de Deusto, Bilbao.