Ingeniería del Software aplicada a un Laboratorio de Introducción a la Programación en Ada

María José García, Pedro Lara, Luis Fernández
Dept. de Programación e Ingeniería del Software
Universidad Europea CEES
28670 Villaviciosa de Odón
e-mail:{pepa, pedro,lufern}@dpris.esi.uem.es

Alberto Díaz
Ing. Técnica en Informática de Sistemas
Centro de Estrudios Superiores Felipe II- UCM
28300 Aranjuez
e-mail: adiaz@cesfelipesegundo.com

Resumen

La metodología aplicada en el primer Laboratorio de Programación al que se enfrentan los alumnos de una carrera de Ingeniería (o Ingeniería Técnica) en Informática es determinante para establecer en ellos unos buenos hábitos ante la programación. En esta ponencia se presenta la solución adoptada en la Escuela de Informática de la Universidad Europea CEES. El enfoque elegido pretende aunar la enseñanza de los fundamentos de la programación y la introducción de conceptos básicos de ingeniería del software para el desarrollo de aplicaciones. También se exponen los criterios de evaluación aplicados, así como la forma de coordinar y gestionar los grupos, a fin de utilizar en todos ellos unos criterios homogéneos.

1. Introducción

Laboratorio de Programación I es una asignatura obligatoria primer curso para las titulaciones de Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas. Esta asignatura cuenta con una dedicación de 6 créditos prácticos durante el segundo cuatrimestre.

Contando desde la implantación del primer curso de los anteriores planes de estudios, el Laboratorio de Programación I lleva impartiéndose ya siete años en la Escuela Superior de Informática. Cada curso han ido surgiendo diversos problemas debidos a la propia idiosincrasia de la asignatura y a su carácter eminentemente práctico.

Por un lado, se ha decidido utilizar una metodología que está basada en los fundamentos de la Ingeniería del Software, de tal forma que se orienta al alumno hacia una forma completa de solucionar los problemas, en lugar de centrarse sólo en la tarea de la implementación en un determinado lenguaje de programación. Así se enlazaría con las posteriores asignatruas

Por otra parte, puesto que ésta es la primera asignatura en la que los alumnos deben construir una solución software completa a un problema dado, se considera fundamental el trabajo individual. Esto implica la necesidad de distribuir a los alumnos en grupos de manera que cada alumno pueda, durante las horas de clase, trabajar con un ordenador. Además los grupos reducidos permiten un mejor seguimiento de la evolución de los alumnos. En nuestro caso los laboratorios están dotados con 30 equipos, por lo que la medida que se suele adoptar es la distribución en grupos de entre 20 y 25 alumnos, cada uno de ellos con un horario y un profesor distinto.

El problema entonces es doble: hay que decidir cómo adoptar el enfoque de la ingeniería del software con alumnos que se enfrentan por primera vez con la implementacion de soluciones en el ordenador y además coordinar a todos los profesores de manera que se sigan unas pautas de actuación unificadas (a fin de ser lo más ecuánimes posibles y de evitar diferencias entre los grupos).

Este curso 2001-2002 el problema se ha incrementado puesto que son ya trece los grupos que se han tenido que formar.

Además es importante guardar una absoluta coherencia entre los contenidos y criterios de esta asignatura y la de Introducción a la Programación, tambien de primer curso y que tiene carácter troncal y anual, prevista como implantación en el plan de estudios como implantación de la troncalidad de Metodología y Tecnología de la

Programación. Su objetivo es proporcionar los contenidos más teóricos sobre programación que se aplicarán después en el Laboratorio de Programación I.

Otro factor que influye en la necesidad de establecer de una manera clara los parámetros de la asignatura es la heterogeneidad de los profesores. Muchos de ellos es la primera vez que imparten esta asignatura, por lo que es importante poder proporcionarles a principio de curso unas guías de actuación:

- Es necesario realizar una planificación previa de la asignatura, incluyendo enunciados, diseños, conjunto de pruebas y fechas de entrega de cada una de las prácticas.
- También es fundamental establecer unos criterios de evaluación claros y únicos.
- Para conseguir evitar incoherencias entre los diversos grupos y también para detectar las posibles copias de prácticas entre alumnos de distintos profesores deben existir mecanismos de coordinación y comunicación entre los profesores de la asignatura.
- Del mismo modo se tiene que facilitar el flujo de información alumno profesor.

El enfoque adoptado para la enseñanza de la programación asume la conveniencia de enseñar primero los principios básicos de la programación procedimental y estructurada (frente a quienes prefieren comenzar directamente con la orientación a objetos). La elección del lenguaje Ada no es casual ya que se pensó en un lenguaje que permita programación procedimental relativamente natural. Pero que, a la vez (frente a lenguajes como Pascal) permita una transición interesante hacia la enseñanza de tipos abstractos de datos (en la asignatura de segundo curso de Estructuras de Datos y de la Información de nuestros planes de estudios) o hacia la orientación a objetos (para asignaturas como Programación Orientada a Objetos).

Por otra parte, la introducción de los conceptos tradicionales de Ingeniería del Software en una asignatura de primer curso permite a los alumnos tener una base sobre la que cimentar posteriormente los conocimientos que adquirirán en las asignaturas plenamente dedicadas a esta materia que se imparten en cursos superiores (dos asignaturas de carácter troncal y una obligatoria que suponen un total de 24 créditos para Ingeniería del Software en la carrera de Ingeniería Informática, 12 créditos (6 troncales y 6 obligatorios) en Ingeniería Técnica en Informática de Gestión y otros 6 créditos obligatorios en Ingeniería Técnica en Informática de Sistemas).

2. Herramientas de trabajo

2.1. Herramientas para los alumnos

Como herramientas de trabajo los alumnos disponen en los laboratorios de máquinas PC pentium 300 MHz. Las prácticas se realizan en el lenguaje de programación ADA, por lo tanto en cada computadora está instalado un compilador (GNAT: GNU Ada Translator), un depurador (GDB: GNU Debugger), y los manuales de referencia del lenguaje (incluyendo el estándar ISO correspondiente [3] y libros típicos como [1] y [5]). Se trabaja también con el editor de desarrollo pcGRASP (Graphical Representations of Algorithms, Structures, and Proceses) y las aplicaciones ofimáticas WORD 98 y Powerpoint para la documentación (memoria) de las prácticas. También disponen de la herramienta Microsoft VISIO para dibujar los diagramas de estructuras.

Para la elección del compilador, depurador y editor que se utilizan en la asignatura se ha tenido en cuenta la necesidad de que los alumnos puedan seguir trabajando en las prácticas fuera del horario lectivo. Por lo tanto se han seleccionado herramientas potentes pero de libre distribución que se ponen a disposición de los alumnos a través de FTP anónimo. También a través del FTP anónimo y la página de la asignatura se entrega a los alumnos, en el momento oportuno, la documentación para la realización de cada práctica.

2.2. Herramientas para los profesores

2.2.1. Herramientas de ayuda a la corrección

A lo largo del proceso de evaluación de los programas entregados por los estudiantes han de realizarse toda una serie de actividades que en gran parte se convierten en una labor tediosa y repetitiva. Entre estas actividades destacan fundamentalmente las siguientes: validación del

ajuste del código entregado al diseño realizado en las fases previas, detección de errores graves en el código y búsqueda de plagios o copias más o menos evidentes.

En un intento de reducir el tiempo dedicado a estas labores surgió la necesidad de llevar a cabo un proyecto de desarrollo que facilitase, automatizase e incluso eliminase en algunos casos cualquier acción manual por parte del docente en relación con los procesos citados anteriormente, particularizado para algunas características de Ada.

Actualmente, dicho proyecto ha dado como fruto un primer entorno piloto que cubre parte de la funcionalidad prevista detectando errores en implementaciones no coincidentes con los diseños entregados (genera un esquema del diseño arquitectónico que se ha implementado incluyendo la modelización de los bucles y número de llamadas a los módulos), encontrando y mostrando errores graves de codificación (utilización inadecuada de las variables) y analizando los datos obtenidos de las comparaciones cruzadas de varios ficheros fuente en busca de indicadores de plagio ([2] y [4]). En particular muestra estadísticas sobre líneas iguales/modificadas/añadidas/borradas, y sobre identificadores iguales/modificados).

Además este entorno permite la compilación y ejecución de cada uno de los ficheros fuente utilizando la entrada/salida estándar o redirigiendo estas a ficheros para unificar las pruebas realizadas por el profesor y comprobar el ajuste con las especificaciones.

Como herramienta de apoyo en la corrección de las prácticas se utiliza además un analizador de código desarrollado como proyecto fin de carrera por un alumno de la Escuela. Este analizador permite realizar tanto análisis estático como dinámico de un determinado código, mostrando tanto grafos de llamadas como de flujo, así como el valor de una serie de métricas definidas por el profesor.

2.2.2. Herramientas de coordinación

Puesto que el número de profesores que imparten la asignatura es bastante elevado, se ha creado la figura de "coordinadores de laboratorio", encargados de tomar ciertas decisiones básicas como son la elección de enunciados de las prácticas de las fechas de entrega de las mismas, y el establecimiento de unas normas comunes y criterios de evaluación uniformes. No obstante, se utiliza una lista de distribución en la que están incluidos todos los profesores de laboratorio así como los coordinadores de la asignatura. De esta manera se facilita la comunicación entre todos, pudiéndose comentar las posibles modificaciones o aclaraciones que, sobre cada práctica, realice cada profesor con sus alumnos. Esto permite una máxima coherencia sin necesidad de costosas reuniones, poco factibles debido a la diversidad de horarios de los profesores.

2.3. Herramientas para la comunicación alumno ↔ profesor

2.3.1. Página Web y repositorio de información de la Asignatura

Como para la mayoría de las asignaturas impartidas en la Escuela de Informática de la Universidad Europea CEES, existe una página web en la que de manera unificada todos los alumnos pueden recoger información general. En este caso en ella recogemos los objetivos generales de la asignatura, el sistema de evaluación, información sobre grupos, profesores y aulas y normas generales para la realización de las prácticas.

Además se proporciona un enlace al repositorio de documentos de la asignatura en el que, además de una plantilla para la entrega de la memoria, se incluye a medida que avanza el curso información específica sobre cada práctica: objetivo, enunciado y fechas de entrega de cada fase. En el momento adecuado, es también allí donde se pone a disposición de los alumnos el diseño arquitectónico en el que tienen que basar su implementación.

2.3.2. Automatización de la entrega de prácticas

Para unificar el modo de entrega de las prácticas eliminando en lo posible el trasiego de papeles y disquetes se ha construido una aplicación a la que se accede a partir de la página web de la asignatura. Mediante un pequeño formulario que el alumno debe rellenar, y que le permite adjuntar los ficheros necesarios, las prácticas son enviadas

por correo electrónico a cada profesor, guardándose registro del momento exacto de entrega.

3. Metodología para la realización de las prácticas

Uno de los objetivos de la asignatura de laboratorio de programación I, aparte de la enseñanza de los fundamentos de la programación, consiste en inculcar en los alumnos el uso de un enfoque de ingeniería del software para el desarrollo de aplicaciones. Durante la asignatura de Introducción a la Programación ya han adquirido los conocimientos necesarios sobre éstas técnicas, en particular, para la fase de diseño se les ha enseñado la metodología de los refinamientos sucesivos, y también las diferentes estrategias algorítmicas básicas [6]. Es también en esa asignatura donde se les presenta la sintaxis del lenguaje ADA, que es en el que se implementarán las soluciones a los problemas propuestos en el laboratorio.

Durante la primera semana de laboratorio se pone en contacto al alumno con el entorno que va a encontrar en la asignatura:

En la primera clase se explica a los alumnos la dinámica del laboratorio, número y tipo de prácticas, metodología y criterios de evaluación.

En la segunda clase se realiza la primera toma de contacto con el laboratorio. En esta clase se le entrega a cada alumno un enunciado y un esbozo de la implementación de una práctica sencilla parecida a algún ejercicio resuelto previamente en la asignatura de Introducción a la Programación. Se trata de que el alumno se familiarice con el entorno de programación que va a utilizar a lo largo del curso, completando la codificación que se le entrega; adicionalmente debe depurar el programa ejemplo, detectando y eliminando diferentes errores introducidos en el código: errores de compilación, mal funcionamiento y utilización inadecuada de variables (errores de ámbito).

A partir de la segunda semana empieza la realización del resto de las prácticas de las que consta la asignatura. Cada una de ellas está centrada en la utilización de alguno de los elementos de programación introducidos en la asignatura de teoría. La primera se dedica a la abstracción procedimental, la segunda a la selección, la repetición y los archivos de texto, la tercera al uso de tipos definidos por el usuario sencillos (enumerados, subrangos y arrays simples), la cuarta a la utilización de arrays y registros, y la última, a archivos binarios y memoria dinámica.

La realización de cada una de las prácticas consistirá en desarrollar cada una de las fases del ciclo de vida clásico: análisis, diseño, implementación y prueba, siguiendo la metodología que se describe a continuación.

3.1. Análisis

La *fase de análisis* se realizará el primer día de clase de cada práctica (normalmente lunes), cada grupo con su profesor. El enunciado se pone a disposición de los alumnos el viernes anterior a través del ftp anónimo, de manera que han tenido tiempo de revisarlo previamente. Se repasa el enunciado, se discuten posibles ambigüedades y se llega a unas especificaciones claras y concisas sobre lo que hay que hacer.

3.2. Diseño

La *fase de diseño* consiste en la elaboración del diseño de datos, arquitectónico y procedimental y de la colección de pruebas a realizar. Los diseños se realizan siguiendo la filosofía de la programación estructurada y el método de los refinamientos sucesivos.

- Diseño de datos: deberá seguir la notación EBNF.
- Diseño arquitectónico: se presentará un diagrama de estructuras completo según la notación clásica [7].
- Diseño procedimental: tendrá que reflejar, al menos, el pseudocódigo del programa principal y de los procedimientos de cierta complejidad.

Una vez que los alumnos hayan entregado la documentación asociada a esta fase, el profesor entregará un modelo de diseño de datos y arquitectónico aproximado (sin especificar exactamente el flujo de datos entre los módulos) junto con el interfaz (formato de presentación de los datos de entrada y salida). Este modelo es el

que deberán seguir los alumnos en las siguientes fases. De nuevo, para evitar papeles innecesarios, el modelo de diseño se pone a disposición de los alumnos en el ftp anónimo de la asignatura.

3.3. Implementación

La *fase de codificación* será la traducción del modelo de la fase de diseño entregado por el profesor al lenguaje Ada, corrigiendo errores con el compilador, y ajustándose al interfaz indicado.

La razón para que codifiquen a partir del diseño del profesor y no del suyo propio es evitar que realicen la implementación y después ajusten el diseño a esa codificación (se ha observado que esto es lo que hacían algunos alumnos en años anteriores, de modo que se ponían a codificar sin detenerse a planificar primero un diseño adecuado)

Para facilitar a los profesores la labor de corrección, de modo que les sea más sencillo identificar los diferentes elementos del código, los alumnos seguirán la siguiente normativa de codificación:

- Al inicio del código, en las primeras líneas del fichero adb, se incluirán los datos del autor (Nombre, nº de expediente, grupo, fecha de inicio de la práctica)
- Todos los módulos deberán estar comentados (¿qué hacen? ¿cómo?), sobre todo aquellas cosas especialmente complicadas.
- Notación para los identificadores:
  1. Identificadores compuestos: separados por guión_bajo
  2. Constantes: todo el identificador en mayúsculas
  3. Variables: empiezan por minúsculas, resto en minúsculas
  4. Funciones o procedimientos: empiezan por mayúsculas, resto en minúsculas

3.4. Pruebas

La *fase de prueba* consistirá en primer lugar en el diseño de un conjunto unificado de pruebas, determinando para cada caso de entrada cuál debería ser el resultado a obtener. Posteriormente se deberá ejecutar el programa en todas aquellas situaciones detectadas como significativas en el diseño de las pruebas, asegurando, a través de ellas, que el programa funciona correctamente y se obtienen los resultados esperados.

Utilizando los medios que proporciona la herramienta desarrollada en esta universidad para capturar la entrada y la salida mediante ficheros de un programa implementado en ADA, será posible probar con una sola ejecución el funcionamiento de las prácticas.

El alumno entregará el resultado de las pruebas diseñadas, y de cualquier otra prueba que se haya realizado (capturadas en ficheros). Deben incluirse resultados finales tanto positivos como negativos. Para los casos en que los resultados no son los esperados, se indicará la posible causa y posibles soluciones (justificar fallos). Los alumnos serán penalizados en la evaluación si entregan un juego de pruebas incompleto, que no detecte errores del código.

3.5. Documentación

Cualquier aplicación software debe estar debidamente documentada. Es esto lo que se les pretende inculcar nuestros futuros programadores. Los alumnos de esta asignatura deben entregar una memoria final (siguiendo un formato específico) que incluya la documentación específica asociada a cada una de las fases anteriores (según el modelo que se les proporciona) y además un manual de usuario contemplando aspectos como a quién va destinado el programa, cómo se ejecuta, qué errores devuelve y cómo se interpretan, requisitos de ejecución, incompatibilidades etc.)

4. Evaluación de las prácticas

En cuanto a la metodología de evaluación, es necesario volver a hacer hincapié en la importancia de la homogeneidad entre los grupos. No hay que olvidar que hay alumnos que están distribuidos en diferentes grupos pero están matriculados en la misma asignatura, y comparten el resto de las clases.

Se ha intentado huir de una tabulación de distintos fallos con una serie de descuentos asociados porque es imposible corregir una práctica basándose en si un fallo descuenta 2 puntos y otro 1. En su lugar se presentan una serie de criterios para guiar en la puntuación de cada

práctica. Pero como siempre ocurre en programación es muy difícil encontrar una forma sistemática de puntuar programas: se necesita tener una visión global del programa y una serie de criterios básicos (y quizás algo de experiencia) para poder asignar una nota.

4.1. Evaluación continua

En esta Universidad se sigue el modelo de evaluación continua. Esto implica la valoración del trabajo del alumno a lo largo de toda la asignatura, pero también la obligación por parte del alumno de una cierta regularidad de resultados.

Las prácticas van incrementando su complejidad a lo largo de la asignatura, siendo la última práctica la más completa.

Al final de la asignatura se realizará un examen que no tendrá nota (si bien es una observación más a disposición del profesor en su evaluación) puesto que sólo es una comprobación de que las prácticas las ha realizado cada alumno.

La calificación de la asignatura vendrá dada por una media ponderada de las notas en cada una de las prácticas (a medida que se incrementa la complejidad de las prácticas, aumenta también su peso en la nota final).

Para aprobar la asignatura se pueden presentar las siguientes posibilidades:

- Alumnos que hayan ido aprobando todas las prácticas: sólo tendrán que presentarse al examen final para comprobar que son autores de la última práctica.
- Alumnos con alguna práctica suspensa antes de la última:
  1. Si no alcanzaron una nota mínima (que se ha establecido en 3 sobre un total de 10) en las prácticas más importantes, pierden la posibilidad de aprobar en junio.
  2. En otro caso: repetirán para junio las prácticas que no hayan llegado a la nota mínima o las no entregadas, presentándose al examen final para comprobar la autoría. En dicho examen se realizarán preguntas sobre ellas (razonamientos, cambios de codificación o de diseño respecto a la anterior versión...).
- En todo caso en la última práctica se debe obtener al menos un 5 para poder aprobar la asignatura.
- En la convocatoria de septiembre se deben entregar tres prácticas (dos de ellas son repetidas del curso) aunque sólo deben repetirse en caso de tenerlas suspensas o no presentadas). El examen de septiembre será personalizado, porque se trata de comprobar la autoría de unas prácticas que se han realizado fuera del periodo de clases y por tanto sin el seguimiento directo de los profesores.

4.2. Criterios de corrección

Como siempre, la consigna es conseguir la mayor homogeneidad en cuanto a los criterios que se aplican para la corrección de cada uno de los aspectos de cada práctica.

Dado que se realizan dos entregas por práctica, debemos valorar tanto la fase de diseño como la de codificación. La proporción será 1/3 la nota de diseño, 1/3 la nota de codificación, 1/3 la nota de pruebas y documentación.

4.2.1. Fase de diseño

- El diseño de datos deberá seguir la filosofía top-down, dejando perfectamente definidas las estructuras de datos adecuadas, incluso aunque aún no sean capaces de codificarlas.
- Reglas que se deben valorar en el diseño arquitectónico:
  1. Diseño estructurado
  2. Seguir la filosofía EPS (entrada-proceso-salida)
  3. Mínimo acoplamiento
  4. Máxima coexión
- El diseño procedimental se debe corresponder con el diseño de datos y arquitectónico entregados, tiene que especificar claramente el interfaz de cada módulo diseñado.
- El diseño de pruebas: Se comparará con el conjunto oficial de pruebas.

4.2.2. Fase de implementación

Existen errores que obviamente conducen al suspenso inmediato:

- Práctica que no compila.
- Implementación diferente del diseño (del profesor)
- No se ajusta a las especificaciones (incluso a las de interfaz)

Hay también errores que conducen al suspenso:

- Mala utilización del ámbito de las variables (utilización de variables globales sin pasarlas por parámetro).
- Mala utilización del ámbito de los módulos (invocación a módulos hermanos).
- Mala estructuración, utilización improcedente de instrucciones de salida (iteraciones de las que se puede salir mediante un exit, procedimientos de los que se sale con un return...)
- Resultados que produzcan error en tiempo de ejecución o que sean contrarios a las especificaciones
- Resultados erróneos (no exactos)

Hay también cuestiones de estilo que pueden ayudar a dilucidar la nota final obtenida en una práctica:

- Hay costumbres que se deben evitar
  - Inclusión innecesaria de sentencias NULL
  - Utilización de bucles definidos en lugar de bucles indefinidos, y viceversa
  - Mala utilización de instrucciones de selección, anidamientos inadecuados, utilización de varias condicionales simples en lugar de una condicional múltiple
  - Funciones que realizan operaciones de entrada/salida de datos
  - Procedimientos que, en realidad, implantan lo que debe ser una función
  - Varias sentencias de retorno dentro de una misma función
- Y algunas que es necesario inculcar
  - Inicialización de variables
  - Uso de identificadores significativos
  - Módulos con una única funcionalidad
  - Comentarios adecuados en el código

4.2.3. Fase de pruebas y documentación

Para facilitar una corrección más cómoda y uniforme, se diseñará un conjunto de pruebas concebido como un fichero de casos de valores de entradas. La idea es que este fichero, que se repartirá a todos los profesores, será la prueba "oficial" que se aplicara para decidir si el programa "funciona" o no.

ADA proporciona una manera sencilla de redireccionar la entrada/salida estándar a ficheros, permitiendo así que se generalicen la utilización de ficheros de prueba. Además, la herramienta de ayuda a la corrección que se ha desarrollado permite automatizar este proceso.

A efectos de evaluación, hay que observar si el programa funciona en todos los casos de prueba (se supone que esto es lo que deberían lograr siempre los alumnos), si falla en algún caso puntual o si ni siquiera permite ejecutar las pruebas (no se respeta el interfaz) o está plagado de fallos.

5. Conclusiones

Se ha presentado una metodología para la realización de prácticas en el laboratorio de programación de primer curso que trata de definir una dinámica que permita homogeneizar los distintos grupos de alumnos de dicha asignatura.

Esta metodología está basada en los conceptos tradicionales de ingeniería del software en el desarrollo de programas, concretamente en las fases del ciclo de vida clásico: análisis, diseño, implementación y prueba.

El hecho de obligar a realizar un diseño previo sin utilizar el ordenador (mejor dicho, utilizándolo como herramienta de apoyo para "mostrar el diseño" pero sin escribir ni una línea de código) potencia la elaboración de una planificación previa en lugar de ponerse a implementar directamente.

Por otro lado, que tengan que implementar a partir de un diseño de otra persona también les acostumbra a adaptarse a la participación de otras personas en el desarrollo de programas.

Se consigue facilitar la corrección de las prácticas a los profesores mediante un conjunto de pruebas oficial, ajustado al interfaz que se les entrega a los alumnos, y que mediante la

redirección de la entrada/salida a fichero permite probar la práctica con una sola ejecución.

Los criterios de evaluación se intentan definir de la manera más clara y concisa posible para evitar la falta de coherencia entre distintos profesores de la misma asignatura, sobre todo entre aquellos que imparten grupos que coinciden juntos en el resto de asignaturas.

Estos criterios están basados en conceptos introducidos en la asignatura de Introducción a la Programación y tratan de fomentar el desarrollo de programas estructurados y bien diseñados.

La metodología adoptada en la asignatura no depende del uso de Ada como lenguaje sino que es exportable a otras asignaturas similares que empleen un lenguaje procedimental.

Referencias

[1] J.G.P. BARNES. *Programming in Ada 95. 2nd Edition* Addison-Wesley, 1998.

[2] P. CLOUGH, *Plagiarism in natural and programming languages: an overview of current tools and technologies*, CS-00-05, Internal Report, Department of Computer Science, The University of Sheffield, junio, 2000.

[3] ISO, *ISO/IEC 8652:1995, Information technology : Programming languages. Ada*, ISO, 1995.

[4] S. A. MENGEL, J. V. ULANS, *A Case Study of the Analysis of Novice Student Programs*, Proceedings of the 12th Conference on Software Engineering Education and Training, 1998.

[5] J. SKANSHOLM. *Ada. From the Beginning. 2*, Addison-Wesley, 1994.

[6] A.B. TUCKER et al., *Fundamentos de informática: Lógica, resolución de problemas, programas y computadoras*, Madrid, McGraw-Hill / Interamericana, 1994.

[7] E. YOURDON, L. CONSTANTINE, *Structured Design*, Prentice-Hall, 1979.