PRÁCTICAS DE TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES EN LA UNIVERSIDAD DE LA LAGUNA

Francisco de Sande, Casiano Rodríguez, Coromoto León

Departamento de Estadística, I. O. y Computación
Universidad de La Laguna
e-mail: {sande, casiano, cleon}@ull.es

Resumen: En este trabajo se presenta una experiencia en la organización de las prácticas de la asignatura Teoría de Autómatas y Lenguajes Formales (TALF) que se imparte en las Ingenierías Técnicas en Informática en el Centro Superior de Informática (CSI) de la Universidad de La Laguna. El trabajo está motivado por el hecho de que si bien es fácil encontrar bibliografía adecuada para esta asignatura, tanto desde el punto de vista de los conceptos que se abordan como del alumnado hacia el que van dirigidos, no conocemos experiencias en las que la atención se centre en un enfoque práctico de estos conceptos.

1.- CONTEXTO

En el CSI de la Universidad de La Laguna se imparten los estudios correspondientes a las Ingenierías Técnicas en Informática de Gestión y de Sistemas así como los de Ingeniero en Informática. En los planes de estudio de la Universidad de La Laguna, la asignatura de TALF figura como troncal para las dos Ingenierías Técnicas, y tiene asignados un total de 9 créditos, repartidos en partes iguales entre teoría y prácticas. La asignatura se imparte a lo largo del primer cuatrimestre del segundo curso.
La distribución lectiva de los 9 créditos de la asignatura se realiza de forma que el alumno recibe 5 horas semanales de clases de aula (3 horas de clases teóricas y 2 horas de clases de problemas) y 1 hora semanal de prácticas en

laboratorio. El principal objetivo de la asignatura es el estudio de modelos formales de máquinas computacionales. Los tópicos que se estudian aparecen en el programa agrupados en cuatro grandes bloques temáticos:

1. Autómatas finitos, expresiones regulares y análisis léxico.
2. Gramáticas independientes del contexto.
3. Lenguajes recursivos.
4. Redes Neuronales formales

Se trata de un temario clásico en asignaturas de este tipo, que recoge las directrices curriculares desarrolladas por instituciones como la ACM y la IEEE-CS.

2.- LAS PRÁCTICAS DE LA ASIGNATURA

La componente práctica de la asignatura se desarrolla mediante la realización por parte de los alumnos de una serie de programas que implementan conceptos que han sido explicados previamente en las clases teóricas. Cada alumno realiza un total de 12-13 prácticas a lo largo de las 15 semanas del cuatrimestre. Las dos primeras semanas de clases se utilizan para introducir el lenguaje de programación en que se desarrollan las prácticas. El lenguaje que se ha elegido para la realización de las prácticas es el C y son varios las razones que han motivado esta elección:

- C es con diferencia el más utilizado de los lenguajes de programación de sistemas.
- Los alumnos de segundo curso ya tienen un bagaje suficiente como programadores en un lenguaje de programación imperativo, Pascal.
- Es el lenguaje natural de programación en el entorno del sistema operativo Unix, plataforma en la que se van a desarrollar las prácticas.
- No hay otras asignaturas en el plan de estudios en las que los alumnos aprendan explícitamente este lenguaje.

En cuanto a la plataforma hardware en la que se realizan las prácticas, también se trata de un entorno nuevo para el alumno. En esta asignatura se les introduce por primera vez en el sistema operativo Unix.

Los objetivos docentes a alcanzar a través de las prácticas de la asignatura se pueden resumir en:

- La consolidación a través de las experiencias de laboratorio de los conceptos adquiridos en las clases teóricas.
- La mejora del dominio de la programación adquirido en el primer curso.

Dentro de este segundo objetivo, en las prácticas se hace hincapié en cuestiones en las que los alumnos no han profundizado suficientemente hasta este momento:

- Estructuración consistente de módulos pequeños para constituir programas mayores.
- El desarrollo de hábitos y técnicas de comprobación de errores y de supervisión del resultado final.
- Escritura de programas fácilmente mejorables.
- Generación de información clara y útil al usuario, que prevea los problemas de la ejecución.

Describiremos a continuación la metodología que se sigue a la hora de realizar las prácticas. A partir de la finalización de las primeras clases de introducción al lenguaje C (que suelen durar de 10 a 12 horas), cada semana se suministra al alumno el enunciado de la práctica que ha de desarrollar, y que ha de ser presentada la semana siguiente. Los enunciados de las prácticas se hacen disponibles al alumno en formato PostScript a través del servidor de ftp del CSI. Además del enunciado de la práctica que ha de realizar, el alumno dispone de programas ejecutables que implementan las funciones que se le pide desarrollar, así como un conjunto de ficheros de prueba para comprobar el correcto funcionamiento de la aplicación que desarrolle. Estos programas y conjuntos de pruebas pretenden ser una ayuda con la cual el alumno puede contrastar el funcionamiento de sus implementaciones. Dependiendo de la complejidad de la aplicación que se pide implementar, a veces se suministra al alumno código que puede formar parte de su implementación. En cualquier caso, en todas las prácticas se orienta al alumno respecto al diseño del programa que ha de desarrollar.

En la realización del trabajo propuesto el alumno puede utilizar cualquier material de referencia. Asimismo se le anima a que discuta los problemas de implementación con otros alumnos del curso, e incluso se alienta la colaboración en la resolución de los problemas que se les presentan. No obstante se les advierte que no está permitida la colaboración en los detalles de implementación. La presentación de las prácticas se realiza siempre de forma individualizada, y cada estudiante es responsable de los resultados y de la implementación que presenta.

Cada alumno tiene asignado un grupo de corrección de prácticas, y cada grupo presenta los resultados alcanzados en el desarrollo en una sesión semanal de corrección de prácticas de una hora de duración. Estas sesiones tienen lugar en el laboratorio frente a un terminal y en las mismas cada alumno es evaluado y se le asigna una calificación. Los criterios de evaluación de las prácticas son los siguientes:

- Se valora en primer lugar el hecho de que la aplicación desarrollada funcione correctamente.
- Se evalúa a través de algunas preguntas si el alumno ha comprendido con precisión los conceptos teóricos implicados en el programa que ha realizado.
- La calidad del código producido: modularidad, documentación, portabilidad.
- Se valora el conjunto de pruebas de entrada que el alumno ha desarrollado para comprobar el correcto funcionamiento de su programa.
- La eficiencia del código que se ha implementado.

A continuación presentamos el conjunto de prácticas que los alumnos realizan en un curso típico.

Módulo 1: Introducción al lenguaje C

Herramientas: Introducción al Unix. El editor vi. El compilador gcc. Práctica 1.1.- Números pedrisco. Práctica 1.2.- La conjetura de Goldbath. Práctica 1.3.- Criptografía. Práctica 1.4.- Diseño de una tabla hash. Práctica 1.5.- Listar todas las palabras de diferentes longitudes presentes en un fichero de texto.

Módulo 2.- Implementación de conjuntos en C

Herramientas: make. Práctica 2.1.- La criba de Eratóstenes. Práctica 2.2.- El problema del cubrimiento de conjuntos.

Módulo 3: Autómatas finitos, expresiones regulares y análisis léxico

Herramientas: lint. Práctica 3.1.- Simulación de un DFA. Práctica 3.2.- Simulación de un NFA. Práctica 3.3.- Conversión de un NFA en un DFA Práctica 3.4.- Minimización del número de estados de un DFA.

Módulo 4: Gramáticas independientes del contexto

Herramientas: Construcción de librerías. Práctica 4.1.- Cálculo de los conjuntos First y Follow para una gramática. Práctica 4.2.- Analizador sintáctico descendente recursivo. Práctica 4.3.- Analizador de programas Batch de DOS. Práctica 4.4.- Analizador de expresiones regulares.

Módulo 5: Lenguajes recursivos

Herramientas: Depuración de programas. gdb. Práctica 5.1.- Simulación de una Máquina de Turing.

Módulo 6: Redes Neuronales Formales

Práctica 6.1.- Simulación de una Red Neuronal.

Los enunciados que se proporciona a los alumnos son autocontenidos, con explicaciones suficientes para que, con los conocimientos adquiridos en las clases teóricas el alumno pueda proceder a su desarrollo.

La experiencia acumulada durante el tiempo que llevamos impartiendo las prácticas de TALF nos permite establecer algunos de los errores que más reiteradamente cometen nuestros alumnos.

- Con frecuencia los alumnos se obsesionan en lo que podríamos llamar "el bajo nivel de la programación" y no ponen en relación la aplicación que han desarrollado con los conceptos aprendidos en teoría. Para evitar este problema, en la evaluación de las prácticas hacemos preguntas a los alumnos acerca del significado conceptual de los resultados que obtienen con sus aplicaciones.
- Los alumnos no están acostumbrados a realizar un análisis previo de la aplicación que van a desarrollar, lo cual suele producir como resultado que sus programas resulten ineficientes, poco claros y mal estructurados. Alentamos a los alumnos a escribir programas modulares y fácilmente modificables. De hecho, en las sesiones de prácticas a veces pedimos a los alumnos que reescriban aquellas partes de su código susceptibles de mejoras sustanciales.
- Los alumnos no dan la importancia necesaria a algunos aspectos de la programación que son cruciales cuando se trabaja en entornos profesionales, realizando grandes aplicaciones y formando parte de equipos de trabajo. En concreto la documentación de sus programas, así como las mínimas interfaces de usuario que han de desarrollar suelen ser deficientes. Desde la primera práctica se deja claro que forma parte de la calidad de un código el que éste vaya acompañado de una documentación útil al usuario que haya de manejar dicho código. A veces realizamos la experiencia de que un alumno trabaje con el código producido por otro para que tomen conciencia de esta problemática.
- Los alumnos carecen de criterios de autoevaluación de los resultados que obtienen a través de sus desarrollos y no diseñan conjuntos de pruebas suficientes y adecuados para comprobar la corrección de los algoritmos que desarrollan. Se suelen conformar con comprobar su programa con los ejemplos que se les han suministrado, lo cual a veces resulta insuficiente. Esta es una de las carencias contra las que más cuesta luchar. El desarrollo de hábitos y técnicas de comprobación de errores es uno de los aspectos más importantes de la programación que no siempre recibe la debida atención. Inculcamos a nuestros alumnos la preocupación por comprobar sus resultados con conjuntos de pruebas suficientemente ricos y adecuados.
- Algunos alumnos se resisten a asumir el entorno de desarrollo Unix y sus herramientas. Desarrollan sus aplicaciones usando compiladores de C sobre MS-DOS y cuando llevan sus programas al laboratorio, éstos

fallan estrepitosamente. Entonces el alumno suele decir aquello de "en mi casa funciona, pero aquí no". La portabilidad es uno de los aspectos en los que ponemos especial énfasis y por ello exigimos la utilización del estándar ANSI C en las implementaciones. Nuestra respuesta a la afirmación anterior por parte de los alumnos suele ser: "un informático debería ser capaz de explicar esa situación". Alentamos a los alumnos a usar también en su casa las mismas plataformas de desarrollo que encuentran en la Facultad.

3.- CONCLUSIONES

Las conclusiones que presentamos a continuación son fruto de la experiencia en la corrección de prácticas, de la observación del trabajo de los alumnos tanto en prácticas como en los exámenes de la asignatura, de las encuestas realizadas a los alumnos y del intercambio de ideas con otros profesores del Departamento.

- La implementación de programas que desarrollan conceptos estudiados previamente en clases teóricas ayuda al alumno a comprender y asimilar mejor dichos conceptos.
- Al finalizar las prácticas de la asignatura, la mayoría de alumnos ha consolidado su experiencia como programador en un nuevo lenguaje de programación y ha aprendido a trabajar en un entorno de programación nuevo utilizando de forma adecuada las herramientas que suministra.
- El número de alumnos que supera satisfactoriamente las prácticas es muy superior al de quienes las suspenden. Hemos de decir además que aquellos alumnos que no superan las prácticas es casi siempre debido a que abandonan la realización de las mismas en cierto momento del curso (frecuentemente antes de la cuarta semana del cuatrimestre).
- En este tipo de prácticas el peor inconveniente es el poco tiempo que el profesor dispone para corregir la práctica de cada alumno. Los grupos de prácticas en laboratorio suelen ser de 15 alumnos por grupo, con lo cual en cada sesión a cada alumno le corresponde un promedio de 4 minutos, que consideramos insuficiente. Una solución a este problema sería incrementar el número de sesiones de laboratorio o también disminuir el número de alumnos en cada grupo.
- Dado que el tener las prácticas aprobadas es un prerequisito para acceder al exámen de teoría, los alumnos solicitan que se asigne un mayor peso en la calificación final al trabajo de prácticas.

- Entendemos que sería positivo para la asignatura cambiar el lenguaje de programación, pasando posiblemente a usar Java como lenguaje de implementación. Por otra parte, entendemos que es fundamental que el alumno estudie explícitamente C en alguna asignatura. Quizás una alternativa podría ser cambiar el lenguaje que se usa en las asignaturas de primer curso, pasando de Pascal a Java.
- Es muy positivo exponer al alumno cuanto antes a entornos de desarrollo más profesionales. Finalmente los alumnos acaban reconociendo las ventajas de trabajar en este tipo de entornos, que inicialmente les resultan menos amigables que los que conocen hasta ese momento.