UNA HERRAMIENTA DE APOYO A LA EVALUACIÓN DE TRABAJOS PRÁCTICOS

José Luis Sánchez Romero[1], Daniel Ruiz Fernández[1]

[1]*Departamento de Tecnología Informática y Computación*
Universidad de Alicante
e-mail: {sanchez, druiz}@dtic.ua.es

RESUMEN: En el presente artículo, se presenta una herramienta que sirve como apoyo a la evaluación de diversos ejercicios prácticos que realizan los alumnos. Se describe el método seguido para la implementación del sistema desarrollado, junto a algunos ejemplos de manejo del mismo, el tipo de resultados que se obtienen y la ventajas derivadas de la utilización de la herramienta.

1.- INTRODUCCIÓN.

En el proceso de enseñanza y aprendizaje de la mayoría de las áreas de la Informática, la aplicación práctica por parte de los alumnos de los conocimientos teóricos adquiridos es de suma importancia. De hecho, en la mayor parte de los estudios universitarios de esta materia, el número de créditos de enseñanza práctica representa un elevado porcentaje de la carga lectiva total [1], [2], [3], [4]. Resulta obvio, por tanto, que la evaluación del trabajo práctico desarrollado por los alumnos representa un componente fundamental en el citado proceso, puesto que, por una parte, estos trabajos permiten al alumno desarrollar destrezas a partir de los fundamentos teóricos y afrontar una amplia casuística de problemas que puede aparecer en el desarrollo de un proyecto informático real; y por otra parte, esta evaluación supone para el docente una retroalimentación que le puede ayudar a modificar sus métodos, con el objetivo de conseguir una mayor calidad en la enseñanza [5], [6].

En muchas ocasiones, los trabajos prácticos que presentan los alumnos consisten en diversos programas que pueden corresponder a problemas puramente relacionados con el software, o bien a simulaciones de implementaciones hardware en relación con el área de la Arquitectura de Computadores.
El análisis de tales programas o códigos fuente constituye una tarea fundamental a la hora de evaluar estos trabajos prácticos realizados por los alumnos [7]. Aunque parte del proceso de desarrollo de estos programas se basa en la aplicación de unas estructuras sintácticas y de unas metodologías de programación preestablecidas, no hay que olvidar el hecho de que es igualmente importante la creatividad y la originalidad, las cuales ayudan a proporcionar distintas soluciones ante un mismo problema. Del posterior estudio de esta diversidad de soluciones, se podrá determinar cuáles presentan mayores ventajas y, en el caso de que alguna de ellas represente una innovación, intentar aplicarlas a otro tipo de problemas.
Resulta indiscutible por tanto que, aunque los alumnos deben compartir ideas y ayudarse mutuamente en la resolución de los problemas que se les plantea, el trabajo individual y personal es el que va a producir una solución definitiva original y va a aportar mayor calidad al proceso de enseñanza y aprendizaje. Dicho trabajo debe ser suficientemente valorado, tanto por el docente como por el propio alumno, quienes deben disponer de algún tipo de mecanismo para medir la originalidad y personalización de los programas entregados.

Con vistas a mejorar la calidad del proceso de enseñanza y aprendizaje y conseguir que los alumnos valoren cada vez más su trabajo personal y la originalidad de sus programas, se ha desarrollado la herramienta SEVAP (Sistema de apoyo a la EVAluación de Prácticas) que intenta establecer una clasificación dentro de un conjunto de trabajos prácticos presentados por los alumnos, proporcionando información sobre las estructuras sintácticas utilizadas, de modo que se ponga de manifiesto la mayor o menor creatividad de dichos trabajos.

2.- DESCRIPCIÓN DE LA HERRAMIENTA

Inicialmente, la herramienta SEVAP ha sido desarrollada para el sistema operativo Windows9x/2000®, aunque no se descarta la posibilidad de su adaptación a otros entornos. Del mismo modo, los programas fuente que la herramienta es capaz de analizar y clasificar deben haber sido desarrollados utilizando el lenguaje de programación C/C++ por ser éste uno de los lenguajes más ampliamente utilizandos en el ámbito docente universitario.

a) Método utilizado

El método utilizado para llevar a cabo la tarea de clasificación de los trabajos prácticos consiste, básicamente, en la realización de un análisis de *clustering* jerárquico [8], [9]; es decir, se trata de construir agrupaciones que, en cada paso de iteración, abarquen un mayor número de objetos analizados, hasta que cada uno de estos objetos quede caracterizado como perteneciente a uno u otro *cluster*. Este análisis puede resultar desglosado en las siguientes fases:

1. Determinación de las características o parámetros asociados a cada uno de los programas fuente. En el caso de la herramienta SICOP, los parámetros que se ha considerado pueden asociarse a cada programa fuente son el número de ocurrencias de los *tokens* siguientes: '*if*', '*while*', '*for*', '=' y ';' .

2. Normalización lineal de los distintos parámetros para cada uno de los programas fuente, mediante la formula

$$x_{ij}' = \frac{x_{ij} - \min_i x_{ij}}{\max_i x_{ij} - \min_i x_{ij}}$$

donde x_{ij} es el valor del parámetro j para el programa i, x_{ij}' es el valor normalizado del parámetro j para el programa i, y donde $\max_i x_{ij}$ y $\min_i x_{ij}$ son, respectivamente, el valor máximo y mínimo del parámetro j para el conjunto de programas analizados.

3. Realización del análisis de *clustering* jerárquico, de forma que se van seleccionando programas de forma individual, cada uno de los cuales pasa a formar parte de un *cluster* ya existente o bien constituye un nuevo *cluster*. Los pasos que se llevan a cabo para obtener las distintas agrupaciones de programas son los siguientes:

   3.1. Obtención de una matriz inicial de distancias entre los distintos programas analizados. Como medida de la distancia, se utiliza la distancia Euclídea, caracterizada por la fórmula

$$d(x_i, x_k) = \sqrt{\sum_j (x_{ij} - x_{kj})^2}$$

donde x_i y x_k son dos programas cualesquiera, y donde x_{ij} es el valor del parámetro j para el programa x_i.

3.2. Agrupación de los dos puntos más próximos que, inicialmente, formarán el primer *cluster*

3.3. Recálculo de las distancias entre los puntos que no pertenecen a ningún *cluster* y la nueva situación de los *clusters*. Hay que tener en cuenta que, en el caso del cálculo de la distancia de un punto a un *cluster*, no es directamente aplicable la fórmula de la distancia Euclídea, sino que se ha de utilizar la fórmula denominada *group average*, que proporciona la distancia entre un punto individual y un conjunto de puntos. Esta distancia viene determinada por la siguiente expresión:

$$d_{xi}(c_j, c_k) = \sqrt{\frac{|c_j|}{|c_j|+|c_k|} d^2(x_i, c_j) + \frac{|c_k|}{|c_j|+|c_k|} d^2(x_i, c_k)}$$

de tal modo que x_i representa un programa cualquiera, c_j y c_k son dos *clusters*, $|\cdot|$ representa el número de programas que conforman cada *cluster*, y $d^2(x_i, c_j)$ identifica el cuadrado de la distancia entre el programa x_i y el cluster c_j. Resulta obvio que, inicialmente, puede utilizarse esta fórmula para hallar la distancia de un punto a un *cluster* formado por dos puntos.

3.4. Selección de la menor distancia entre dos programas o entre un programa y un *cluster*. En el primer caso, los dos programas formarán un nuevo *cluster*; en el segundo caso, el programa pasará a formar parte del *cluster* ya existente.

3.5. Repetición del proceso anterior a partir del paso 3.3, hasta llegar al punto en que todos los programas han pasado a formar parte de algún *cluster*.

b) Ejemplo de utilización de la herramienta

El manejo de la herramienta se detalla a continuación: tras la puesta en ejecución de la misma, se ha de seleccionar la unidad de disco y el directorio que contiene los programas fuente que se desea clasificar, los cuales deben tener la extensión .c o .cc .

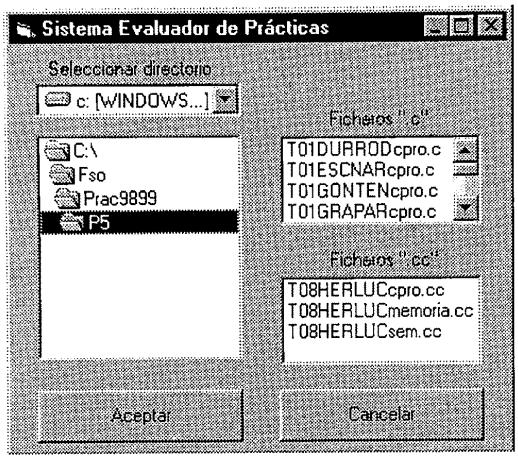

Figura 1. Interfaz del usuario de la herramienta SEVAP

Una vez finalizado el proceso, el usuario recibe un mensaje por pantalla en el que se le informa de que los resultados del análisis han quedado almacenados en el fichero clusters.txt, que se encontrará en el mismo directorio seleccionado en el paso anterior. Este fichero presentará un aspecto semejante al de la Figura 2.

Como se puede observar en la Figura 2, en el fichero de resultados clusters.txt aparecen los distintos *clusters* hallados junto con los programas fuente que forman parte de cada uno de ellos. Además, hay que señalar que, junto a cada programa aparecen, separados por dos puntos, los valores de los distintos parámetros que lo caracterizan, es decir, número de ocurrencias de los *tokens* '*if*', '*while*', '*for*', '=' y ';' , lo cual ayuda a localizar de forma más precisa aquellos programas fuente que, dentro de un *cluster*, presenten una mayor similitud. Así mismo, es preciso destacar que los programas que pertenecen a cada *cluster* aparecen ordenados por su antigüedad de pertenencia al mismo en orden decreciente, lo cual constituye un nuevo factor que se puede tener en cuenta a la hora de intentar localizar aquellos programas fuente más similares.

```
Cluster 1
=============
T10ESOMEcpro.c -->  24: 1: 1: 66: 240
T10SATSEMcpro.c -->  24: 1: 1: 66: 240

Cluster 2
=============
T10GARcpro.c -->  37: 1: 4: 89: 230
T10MARcpro.c -->  37: 1: 4: 89: 230

Cluster 3
=============
T01GRAPARcpro.c -->  42: 2: 2: 107: 257
T06BERPARcpro.c -->  41: 2: 2: 107: 261
T06CANFRUcpro.c -->  33: 2: 2: 103: 279
T08CALcpro.c -->  31: 2: 1: 109: 257
T10BERSORcpro.c -->  49: 1: 3: 118: 302
T10ESOMELcpro.c -->  36: 1: 2: 123: 300
T11MARZORcpro.c -->  47: 2: 1: 104: 229
```

Figura 2. Ejemplo de fichero de resultados de la herramienta SEVAP

Siguiendo con el ejemplo de la Figura 2, observamos que el *cluster* 1 está formado únicamente por dos programas fuente, T10ESOMEcpro.c y T10SATSEMcpro.c , lo cual indica que entre estos dos programas existe un grado de similitud mucho mayor que entre cualquiera de ellos y el resto de programas del conjunto total. Una circunstancia análoga ocurre con respecto al *cluster* 2. En cuanto al *cluster* 3, vemos que está formado por un conjunto mayor de programas, entre los cuales el grado de similitud va disminuyendo paulatinamente, es decir, los programas fuente T01GRAPARcpro.c y T06BERPARcpro.c serían los que presentarían, dentro de este *cluster*, un mayor parecido en cuanto a las estructuras sintácticas utilizadas.
A partir de la Figura 3, se observa que la situación que denotará un mejor grado de aprendizaje y una mayor originalidad en los trabajos realizados será aquella en la que no se produzcan grupos, es decir, el nivel inferior de la pirámide en el que cada grupo queda formado por un solo programa. De la misma forma la situación que habrá que intentar evitar será la que proporcione directamente el nivel superior del diagrama, es decir, un solo grupo que englobe todos los programas analizados

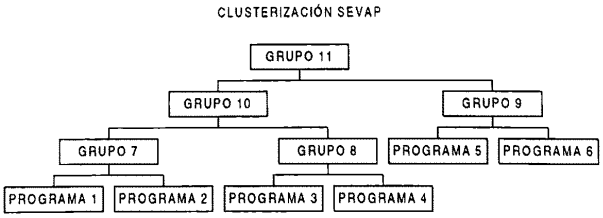

Figura 3. Diagrama representativo del proceso de *clustering* jerárquico

3.- CONCLUSIONES

Se ha desarrollado una herramienta automática que facilita la labor de evaluación de diversos trabajos prácticos realizados por el alumno, labor asumible tanto por el propio alumno como por el docente. Más concretamente, simplifica la evaluación de aquellos trabajos consistentes en uno o varios programas fuente desarrollado para plantear soluciones alternativas a distintos problemas. Esta herramienta realiza un análisis de *clustering* jerárquico sobre un conjunto de programas fuente, los cuales son preprocesados para extraer las características o parámetros representativos de cada uno de ellos. Se obtiene un fichero de resultados, del cual se puede deducir cuáles son los programas que aportan una solución con mayor grado de originalidad.
La herramienta funciona inicialmente bajo Windows 9x/2000® y analiza código fuente en C/C++; no se descarta su adaptación a otros entornos y lenguajes de programación. Del mismo modo, podrían escogerse otras características propias de los programas fuente que supongan un aumento del grado de representatividad.
La herramienta puede solicitarse a los autores, mediante el envío de un mensaje de correo electrónico o por cualquier otro medio alternativo.

4.- BIBLIOGRAFÍA

[1] "Directrices Generales Propias de los Planes de Estudio conducentes a la obtención de los títulos oficiales de Ingeniero en Informática, Ingeniero Técnico en Informática de Gestión e Ingeniero Técnico en Informática de Sistemas". Consejo de Universidades. Reales Decretos 1459, 1460 y 1461 de 26 de octubre de 1990. BOE nº 278 de 20 de noviembre de 1990, págs. 34401-34405

[2] "Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Gestión de la Escuela Politécnica Superior de la Universidad de Alicante". Resolución de 18 de septiembre de 1992 de la Universidad de Alicante. BOE nº 18 de 21 de enero de 1993, págs. 1565-1578

[3] "Plan de Estudios conducente al título de Ingeniero en Informática de la Escuela Politécnica Superior de la Universidad de Alicante". Resolución de 18 de septiembre de 1992 de la Universidad de Alicante. BOE nº 37 de 12 de febrero de 1993, págs. 4467-4478

[4] "Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Sistemas de la Escuela Politécnica Superior de la Universidad de Alicante". Resolución de 18 de septiembre de 1992 de la Universidad de Alicante. BOE nº 37 de 12 de febrero de 1993, págs. 4478-4489

[5] Medina Rivilla, A. et al. "Evaluación de los procesos y resultados del aprendizaje de los estudiantes". Universidad Nacional de Educación a Distancia, 1998

[6] Alonso Tapia, J. "Evaluación del conocimiento y su adquisición". Centro de Publicaciones del Ministerio de Educación, 1997

[7] Flórez Revuelta, F.; Fuster Guilló, A.; Grediaga Olivo, A.; Mora Pascual, J. M.; Soriano Payá, A. "Sistema Automático de Ayuda a la Evaluación Práctica Docente". Actas de las IV Jornades sobre l'Ensenyament Universitari de la Informàtica, págs. 273-277. Enginyeria i Arquitectura La Salle, Universitat Ramón Llull, 1998

[8] Ferrari, D.; Serazzi, G.; Zeigner, A. "Measurement and Tuning of Computer Systems". Prentice-Hall, 1983

[9] Hartigan, J. A. "Clustering Algorithms". Wiley, 1975