Actas de las Jenui, vol. 4. 2019.
Páginas: 357-360

Analizando el proceso de aprendizaje de nuestros alumnos: un caso práctico

Daniel Amo, Joan Navarro y Xavi Canaleta
Departament d'Enginyeria
La Salle, Universitat Ramon Llull
Barcelona
{daniel.amo,joan.navarro,xavier.canaleta}@salle.url.edu

Resumen

Actualmente la analítica del aprendizaje nos proporciona herramientas que nos permiten conocer con más detalle el proceso de aprendizaje de nuestros estudiantes y, gracias a ello, podemos realizar mejoras en las estrategias que utilizamos para conseguir una mayor eficacia y mejorar su rendimiento.
Los entornos de programación son espacios de aprendizaje idóneos para potenciar la introducción de herramientas de analítica del aprendizaje y monitorizar dichos entornos y su actividad. A partir de la captura de estos datos hacen falta instrumentos que nos permitan convertir dicha información en conocimiento útil para poder sacar conclusiones y actuar en los procesos de enseñanza - aprendizaje diseñados.
En esta ponencia se presenta un primer ejemplo práctico de uso de estos instrumentos que ha permitido al profesor de la asignatura monitorizar la evolución de los hábitos de programación de los estudiantes en este entorno y detectar sus puntos fuertes y puntos débiles.

Abstract

The advent of learning analytics has brought a set of tools and techniques to discover new insights from the students learning process and, accordingly, enhance existing teaching strategies to achieve a greater efficiency and improve their performance.
Computer programming environments are ideal learning environments to introduce learning analytics tools for monitoring students and their activity. These tools enable the teaching staff to automatically collect vast amounts of information that, when properly converted into useful knowledge and analyzed, will enable them to draw meaningful conclusions and support the teaching-learning processes designed.
This paper presents a first practical example of the use of these instruments that has allowed the subject teacher to monitor the evolution of the programming habits of the students in this environment and detect their strengths and weaknesses.

Palabras clave

Analítica del aprendizaje, Programación, Sistemas Operativos.

1. Introducción

Los procesos de innovación y seguimiento de la calidad de asignaturas dentro de los grados de Ingeniería Informática deben ser constantes y continuados [1]. La mejora constante es uno de los aspectos clave para mantener dicha calidad. Y aunque en el caso de éxito que nos ocupa se han seguido diferentes estrategias para garantizar esa calidad y realizar la mejora continua [1, 2] la aparición de nuevos elementos siempre puede dar pie a la introducción de instrumentos de análisis no contemplados hasta el momento [3], como es la analítica de datos obtenidos en entornos de programación real.

En este trabajo se considera que la idea es todavía incipiente y, aunque se ha puesto en marcha este curso académico y se han obtenido unos primeros resultados preliminares, se llevan a este encuentro con un objetivo claro de obtener retroalimentación y compartir con la comunidad de docentes universitarios de la informática las técnicas empleadas y su utilidad en el campo de los procesos de enseñanza y aprendizaje de la programación, en este caso desde la asignatura de Sistemas Operativos.

La comunicación presenta el entorno de aplicación de las técnicas de analítica de datos, para realizar a continuación de manera muy breve una descripción de las técnicas usadas para la recogida de datos en el proceso de aprendizaje, presentar unos resultados cuantitativos preliminares y mostrar la utilidad de los mismos en la reflexión y toma de decisiones gracias al análisis del progreso del alumno en un entorno de programación real.

Los últimos avances en las Tecnologías de la Información y la Comunicación (TIC) aplicados al ámbito educativo combinadas con la analítica de datos han fomentado el nacimiento y crecimiento exponencial de los conceptos analítica del aprendizaje [4] y *educational data mining* [5, 6, 7]. Este paradigma consiste en utilizar estas nuevas tecnologías [8] para obtener, entre otras cosas, información en tiempo real acerca de los estudiantes y ayudarles en el desempeño de las actividades [9].

2. El entorno analizado

La asignatura de Sistemas Operativos es una asignatura semestral con los contenidos que se describen en el Cuadro 1 [1].

SISTEMAS OPERATIVOS
1. Introducción a los Sistemas Operativos 2. El núcleo de un sistema operativo 3. Planificación 4. Mecanismos de comunicación, sincronización y exclusión mutua.

Cuadro 1: Contenidos de Sistemas Operativos

La asignatura tiene diferentes actividades de aprendizaje y evaluación muy prácticas y que son claves para los resultados finales de dicha asignatura. Destacamos dos en especial y sobre los cuales se desea obtener información del proceso:

1. Sesiones de laboratorio: son sesiones semanales obligatorias de 1,5 horas donde los estudiantes ponen en práctica las herramientas de llamadas al sistema operativo que este proporciona para los entornos de programación en C.
2. La práctica: el estudiante debe desarrollar por parejas una práctica de gran volumen (sobre las 3.000-4.000 líneas de código en lenguaje C) donde el alumno aplica todos o gran parte de los instrumentos aprendidos.

Parece interesante poder obtener información sobre los hábitos de programación. Con ellos seguramente podremos guiar al estudiante, comprender su proceso de aprendizaje y reforzar los puntos donde demuestra más carencias.

Tanto para realizar la práctica de la asignatura como para realizar las sesiones de laboratorio y también para testear los diferentes elementos de programación en C que se van introduciendo (*signals*, *forks*, *threads*, *pipes*, *sockets*, semáforos, colas de mensajes, etc.), los alumnos se conectan mediante SSH a servidores Linux de la escuela. Esto permite capturar las interacciones de los alumnos de forma centralizada y poco intrusiva. Con el objetivo de evitar hacer incómodas alteraciones en las cuentas de usuario de los alumnos, se ha modificado el proceso de ejecución del binario del compilador (*gcc*) de forma que cuando se ejecute el comando *gcc* se invoque un script en *bash* que:

1. Captura datos antes de empezar la compilación:
   - Parámetros (argumentos) que recibe el compilador (flags, fichero(s) origen, fichero destino, etc.
   - Fecha y hora actual
   - Nombre de usuario y dirección IP de origen.
2. Ejecuta el compilador.
3. Captura la salida del compilador y, además, la muestra por pantalla.

Para evitar problemas de concurrencia (varios usuarios generando *logs* de compilación simultáneamente) y agilizar el proceso de captura de datos, el *script* de captura de datos utiliza la herramienta *syslog* para almacenar los datos capturados.

3. Resultados preliminares

La asignatura de Sistemas Operativos ha tenido 42 alumnos matriculados durante el curso 2018-19. El primer ejemplo es un resultado cuantitativo que se muestra en el Cuadro 2. Corresponde al total de compilaciones realizadas en los servidores una vez filtrados los alumnos de sistemas Operativos. Cabe decir que estos servidores son utilizados prácticamente en exclusividad por estudiantes de primer curso para la asignatura de Programación I y también por los de tercer curso de Sistemas Operativos. El tercer servidor, Puigpedros, se utiliza para los estudiantes de segundo curso en la asignatura de Bases de Datos.

Servidor	**Total**	**Sistemas Operativos**
Matagalls	158.847	1.625
Montserrat	124.450	6.715
Puigpedros	4.104	6
Totales	**287.401**	**8.346**

Cuadro 2: Total de compilaciones

Esta primera información cuantitativa produce cierta sorpresa: 8.346 compilaciones desde el 15 de septiembre de 2018 hasta el 15 de enero de 2019. Hay que tener en cuenta que los estudiantes hacen las sesiones en grupos de dos personas. Por lo tanto hay una media de 400 compilaciones por pareja.

Otro dato que se ha calculado es el número de compilaciones correctas (OK) y erróneas (KO) que se han producido durante el transcurso de las sesiones de laboratorio. Se han filtrado los datos con las fechas y horarios de las sesiones para delimitar la actividad en

Sesión	OK	KO	Total
S1	541	409	950
S2	361	307	668
S3	319	269	588
S4	470	372	842
S5	46	64	110
S6	422	411	833
S7	525	689	1214
S8	601	1160	1761
Total	**3285**	**3681**	**6966**

Cuadro 3: Compilaciones con éxito (OK) y erróneas (KO) en las 8 primeras sesiones de laboratorio.

las mismas. Los datos se presentan en el Cuadro 3 agregando los datos de todos los servidores.

En esta primera extracción de datos, también se ha obtenido información cualitativa y cuantitativa sobre la tipología de errores, segmentándolos inicialmente por errores propiamente dichos y por *warnings* (avisos). El Cuadro 4 nos muestra un ejemplo de la información que podemos extraer de los *logs* generados por los servidores para poder analizar qué tipo de avisos (*warnings*) genera más el compilador debido a los programas de los estudiantes.

Finalmente, otra exploración interesante es poder observar los diferentes tipos de error que se producen durante las sesiones de laboratorio o durante la realización de la práctica o en cualquier momento de desarrollo de la asignatura. El Cuadro 5 nos muestra un caso de ejemplo del servidor Montserrat con los errores producidos en las sesiones.

Toda esta información que puede obtenerse en tiempo real, segmentada por fecha, hora y con datos cuantitativos puede ayudar al docente a detectar el comportamiento de los estudiantes y dónde es necesario incidir más en las sesiones lectivas para evitar los errores más frecuentes y, a partir de sus puntos débiles, establecer elementos de apoyo para su mejora.

Para profesores expertos en los entornos de programación está información les es de mucha utilidad puesto que al detectar los errores y avisos más comunes que producen las compilaciones de los estudiantes saben perfectamente en qué se equivocan los estudiantes (ya sea conceptualmente o en la estrategia de generación de código). Así se pueden adoptar medidas y dar la retroalimentación adecuada para mejorar en los aspectos donde el alumno presenta más carencias para futuras sesiones.

Con estos datos obtenidos podemos inferir algunos comportamientos en los hábitos de programación de los alumnos. Por ejemplo, el gran número de compilaciones denota que el estudiante utiliza el compilador como una herramienta de testeo. Es decir, compila sabiendo que obtendrá errores y, de este modo, considera que le sirve para depurar los errores sintácticos del código. Compilar no es costoso y el paradigma de "hacerlo bien a la primera" parece que ha entrado en desuso. Avisos como '*unused parameter*' (229 ocurrencias) parecen constatar esta percepción.

Tipo de aviso	Nº
warning: unused variable	282
warning: unused parameter	229
warning: implicit declaration of function	98
warning: variable '(.)' set but not used*	89
warning: (.) may be used uninitialized in this function*	87
warning: passing argument (.) of '(.*)' makes pointer from integer with*	53
warning: passing argument (.) of '(.*)' makes pointer from integer without a cast*	48
warning: character constant too long for its typ	36
warning: format '(.)' expects argument of type*	29
warning: ordered comparison of pointer with integer zero	18
warning: comparison between signed and unsigned integer expressions	10
warning: incompatible implicit declaration of built-in function	10
warning: comparison between pointer and integer	4
warning: initialization makes integer from pointer without a cast	4
warning: too many arguments for format	3
warning: format not a string literal and no format arguments	3
warning: cast from pointer to integer of different size	2

Cuadro 4: Clasificación de los avisos más frecuentes por número de ocurrencias en el servidor Matagalls.

La tipología de las actividades evaluables parece que también afecta a la metodología de desarrollo usada por los estudiantes. Por ejemplo, el número de compilaciones en las sesiones semanales (que tienen un tiempo de desarrollo limitado a la sesión) son muchísimo más elevadas que las compilaciones realizadas para resolver la práctica de la asignatura. Esta última es mucho más compleja pero no existe el factor de limitación de tiempo para su entrega. Esto hace que el estudiante desarrolle con más calma su código y compile mucho menos a menudo que en las sesiones donde se constata que se aplica una programación más compulsiva.

Tipo de error del compilador	Nº
error: '(.)' undeclared*	426
error: expected expression before	80
error: too few arguments to function	78
error: invalid operands to binary	63
error: conflicting types for	60
error: unknown type name '(.)'*	27
error: expected declaration or statement at end of	26
error: invalid type argument of unary '(.)'*	25
error: '(.)' has no member named '(.*)'*	22
error: (.): No such file or directory #include <(.*)>*	19
error: '(.)' is a pointer*	18
error: lvalue required as left operand of assignment	17
error: subscripted value is neither array nor pointer nor vector	16
error: two or more data types in declaration specifiers	11
error: invalid initializer	10
error: array size missing in	6
error: storage size of	5
error: expected statement before '(.)'*	2
error: expected identifier before '(.)'*	1

Cuadro 5: Clasificación de los errores clasificados por número de ocurrencias en el servidor Montserrat.

4. Conclusiones

Este estudio es aún muy preliminar. Tan solo es la punta del iceberg con datos explorados durante este primer semestre del curso 2018-19 y es evidente que falta mucho recorrido tanto para poder extraer conocimiento de los datos que se han recogido como para capturar más datos que puedan dar más información sobre los procesos de aprendizaje en entornos de programación.

Conocer el número de compilaciones por alumno nos ayuda también a comprender su estrategia, progreso o incluso posible bloqueo. Por ejemplo, con los datos recogidos se han podido detectar problemas con dos alumnos, uno con un 12% de compilaciones respeto al total y el otro con un 9%, valores muy altos respecto a la media (3,68%). Estos datos también permiten prevenir abandonos al detectar que hay alumnos que generan un 0% de compilaciones.

Esperamos que este artículo presentado en formato breve (póster) nos permita recoger ideas para el tratamiento de los datos que se tienen, así como propuestas de recopilación de información nueva evolucionar estos primeros análisis. Se espera poder analizar los datos ya registrados de otras asignaturas (Programación de primer curso) y realizar comparativas entre la información obtenida y la diferencia entre alumnos noveles y estudiantes de tercero de grado.

Referencias

[1] Xavi Canaleta. Sistemas Operativos, un análisis a largo plazo de los procesos de innovación docente. En *Actas de las XXIV Jornadas sobre la Enseñanza Universitaria de la Informática*. (pp. 295 - 302), Barcelona, Julio 2018.

[2] Xavi Canaleta, Joan Navarro, Xavi Solé, David Vernet y Pau López. Método no formal para la evaluación de la docencia aplicada al Grado de Ingeniería Informática. En *Actas de las XX Jornadas de la Enseñanza Universitaria de la Informática*. (pp. 445-452), Julio 2014.

[3] Joan Navarro, Daniel Amo, Xavi Canaleta, Ester Vidaña-Vila y Carmen Martínez. Utilizando analítica del aprendizaje en una clase invertida: experiencia de uso en la asignatura de Sistemas Digitales y Microprocesadores. En *Actas de las XXIV Jornadas de Enseñanza Universitaria de la Informática*. (pp. 391 - 394), Barcelona, Julio 2018.

[4] George Siemens y Phil Long. Penetratingthefog: Analytics in learning and education. EDUCAUSE review 46.5 (2011): 30.

[5] Laura Calvet Liñán, y Ángel Alejandro Juan Pérez, Educational Data Mining and Learning Analytics: differences, similarities, and time evolution. International Journal of Educational Technology in Higher Education, 12(3), 98-112.

[6] Phil Long, George Siemens, Gráinne Conole y Dragan Gašević. Proceedings of the 1st International Conference on Learning Analytics and Knowledge (LAK11), Banff, AB, Canada, Feb 27-Mar 01, 2011. New York: ACM.

[7] Cristobal Romero, Sebastián Ventura y Enrique García. Data mining in course management systems: Moodle case study and tutorial. Computers & Education, 51(1), 368-384.

[8] Emilia López-Iñesta, Daniel García-Costa, Francisco Grimaldo y Eduardo Vidal-Abarca. Read&Learn: Una herramienta de investigación para el aprendizaje asistido por ordenador. Magister: Revista miscelánea de investigación, 30, 21-28.

[9] Cristobal Romero y Sebastián Ventura. Educational data mining: A survey from 1995 to 2005. Expert systems with applications, 33(1), 135-1.