Encuentro Internacional de Educación en Ingeniería ACOFI
GESTIÓN, CALIDAD Y DESARROLLO EN LAS FACULTADES DE INGENIERÍA
Cartagena de Indias, Colombia
18 al 21 de septiembre de 2018

ENSEÑANDO PROGRAMACIÓN A NATIVOS DIGITALES

Gloria Bautista, Yuranis Henríquez, Jairo Serrano

Universidad Tecnológica de Bolívar
Cartagena Colombia

Resumen

En junio del 2014 el comité curricular de ingeniería de sistemas de la Universidad Tecnológica de Bolívar UTB, planteó la necesidad de reestructurar los cursos de algoritmos y programación, asumiendo la reflexión, la investigación y el debate alrededor del diseño curricular por competencias. El paso del currículo asignaturista al modelo por competencias, ha sido un esfuerzo que se ha venido gestando en las instituciones de educación superior en los últimos años, con el objetivo de articular con la formación para el trabajo sin perder el compromiso que la educación superior tiene con la investigación.

En este artículo se mostrará el análisis, diseño y resultados de la reestructuración del plan de curso de algoritmos teniendo en cuenta la inclusión de competencias y metodologías de aprendizaje basadas en el uso de tecnologías. La mayoría de los usuarios de este curso son estudiantes en proceso de adaptación, provenientes de la educación media que comienzan su camino a la educación superior, por lo tanto, el enfoque debe sincronizarse con el perfil de aprendizaje de los usuarios y los diferentes métodos que hoy en día se adoptan para lograr ese aprendizaje.

Palabras clave: enseñanza; ciencias computacionales; ingeniería

Abstract

In June 2014, the curricular committee of systems engineering of the Universidad Tecnológica de Bolívar posed the need to restructure the courses of algorithms and programming, assuming reflection, research and debate around the curricular design by outcomes.

The conversion from the course-based curriculum to the outcomes model has been an effort that has been developing in higher education institutions and also in secondary education.
In this article we will show the analysis, design and results of the restructuring of the syllabus of algorithms and programming taking into account the inclusion of competences and learning methodologies based on the use of technologies.

The majority of users of this course are students in the process of adaptation, from middle school who begin their path to higher education, therefore the approach must be synchronized with the user's learning profile and the different methods that today in day they are adopted to achieve that learning.

***Keywords**: teaching; programming; computer science; engineering*

1. Introducción

En el 2014 el comité curricular de ingeniería de sistemas de la UTB, realizó una revisión del curso de algoritmos y programación, un curso introductorio que toman todos los estudiantes de ingeniería en general. Este curso tiene tres créditos, para la UTB, un crédito significa tres horas de trabajo académico, distribuidas en una hora de clase presencial y dos horas de trabajo independiente. Los tres créditos del curso son la medida en tiempo para el desarrollo del syllabus, las competencias a alcanzar se deben planear teniendo en cuenta la actividad semanal el estudiante, en este caso tendrá tres horas de clase presencial y seis horas de trabajo independiente. Los semestres académicos son de dieciséis semanas. De esta forma el plan del curso de algoritmos y programación se desarrolla en 144 horas de trabajo académico.

En el 2014 el curso tenía un mayor acompañamiento presencial, se dictaban cinco horas semanales de clase presencial y cuatro horas de trabajo independiente. Aunque el plan se desarrollaba en las mismas 144 horas de trabajo académico, había algunos problemas relacionados con la eficiencia en la carga académica del profesor, el rendimiento del estudiante y los resultados esperados. Entre otros problemas, la alta deserción de estudiantes de primer nivel, reflejada en las bajas notas del curso.

Hasta ese año, algoritmos y programación se dictaba en formato de clase magistral y las prácticas o desarrollo de algoritmos, se hacían en papel y lápiz, pero con la reestructuración del nuevo plan de trabajo, para el siguiente curso se hace uso del computador, las prácticas se realizan programando en un entorno real usando python como lenguaje de programación.

En este artículo se mostrará el desarrollo del rediseño del curso y su ejecución. Además se especificarán las estrategias para la generación de contenidos llevados a estudios de casos y la integración de los resultados de aprendizaje por medio de prácticas realizadas en el computador. De esta manera se logró reducir de cinco a tres horas de clase presencial a la semana, realizando algunos ajustes al plan de curso y logrando una mejor distribución del trabajo en clase e independiente.

Este curso es tomado por estudiantes de primer año que están en proceso de adaptación a la vida universitaria. El curso se diseñó teniendo en cuenta que muchos de los colegios de donde egresan tienen algunas debilidades en cuanto a STEM (Ciencia, Tecnología, Ingeniería y Matemáticas en inglés) y además que en los currículos de educación media, los cursos de informática se dedican al trabajo en aplicaciones de ofimática. Esto ocasiona que el proceso para adaptarse a la mecánica de la universidad se torne complejo. Para dar solución a este inconveniente, se tomaron de base algunas experiencias como el modelo ADRI para la enseñanza de la programación (Malik, *et al.*, 2017) o la metodología basada en equipos (Ashraf, *et al.*, 2012) y también se incluyen técnicas de gamificación (Dixon, et al., 2011), contextualizadas y adaptadas al entorno de los estudiantes de la UTB.

2. Metodología para el diseño del curso

El desarrollo del curso se concibe a partir del trabajo realizado en clase con el acompañamiento directo del profesor y continúa con el trabajo independiente, guiado por medio de comunicación asincrónica, desde el aula de acompañamiento virtual, como apoyo a la presencialidad y para el logro de las competencias esperadas. El rediseño se comenzó con la adaptación de la metodología de casos de estudio (Marcia, et al., 1992) aplicados a la enseñanza de la programación usando situaciones problémicas, enfocados a solucionar ejercicios de fisica, calculo, matemática y otros que han sido estudiados en cursos previos, en el colegio o concurrentemente en los otros cursos del primer año de universidad.

Después de un análisis de usos, sintaxis, librerías disponibles, además de revisar métricas como el índice Tiobe[1], entre otros, se tomó como determinación usar como lenguaje de programación Python, siendo este una buena elección al día de hoy.

2.1. Resultados de aprendizaje

ACM - Computer Science Curricula- (ACM, 2013) define tres niveles de dominio de un tema para los estudiantes y recomienda en su área de conocimiento **Fundamentos de Desarrollo de Software** los temas: algoritmos y diseño, conceptos fundamentales de programación y estructuras de datos. Los tres niveles y sus objetivos de aprendizaje son:

- **Familiaridad**
  - Discute sobre la importancia de los algoritmos en el proceso de solución de problemas.
  - Discute como un problema puede ser solucionado por diferentes algoritmos, cada uno con propiedades diferentes
  - Identifica y describe posibles usos de los tipos de datos básicos
- **Uso**:
  - Crea algoritmos que resuelven problemas sencillos.
  - Usa un lenguaje de programación para implementar, probar y depurar algoritmos que solucionan problemas sencillos.

[1] https://www.tiobe.com/tiobe-index/ indice de lenguajes de programación más usados.

- Implementa, prueba y depura funciones y procedimientos recursivos sencillos
- Aplica técnicas de descomposición para implementar programas en pequeñas fases.
- Crea programas que usen tipos de datos primitivos.
- Modifica y amplia pequeños programas que usen estructuras de control condicionales o iterativas y funciones.
- Diseña, implementa, prueba y depura programas que usen los siguientes elementos: cálculos sencillos, entrada y salida, condicionales y estructuras iterativas, funciones sencillas y con parámetros.
- Escribe un programa que use entrada y salida de datos a un archivo para tener persistencia de datos entre múltiples ejecuciones.

- **Evaluación**:
  - Selecciona las estructuras condicionales e iterativas adecuadas para la tarea asignada.
  - Analiza y explica el comportamiento de programas que involucran programación básica, variables, expresiones, asignaciones, I/O, estructuras de control, funciones, paso de parámetros y recursividad.

Para la construcción de los casos de estudio semana a semana se tuvieron en cuenta los anteriores objetivos de aprendizaje y se respondieron las siguientes inquietudes:

- ¿Qué objetivo de aprendizaje se alcanzará esta semana?
- ¿Cómo se demostrará que se alcanzó el objetivo?
- ¿Cómo se realizará la práctica?
- ¿Qué retroalimentación se dará?
- ¿Qué tarea o trabajo adicional se desarrollará para ayudar en la meta semanal?
- ¿Qué tipo de evaluación es mejor según cada objetivo?
- ¿Cómo realizar y revisar el trabajo en equipo?
- ¿Cómo es la mejor manera de implementar los materiales de apoyo en SAVIO[2]?

Con base en los elementos anteriores se diseñó e implementó un trabajo semanal con los siguientes elementos:

2.2. Trabajo individual

Con el fin de desarrollar las habilidades analíticas de los estudiantes, el profesor brindará la información pertinente al contexto para comenzar la lectura al caso de estudio por medio de una clase magistral de una hora, en esa hora además brindará toda la fundamentación teórica que servirá para el desarrollo y alcance del resultado de aprendizaje semanal, el trabajo individual inicial se da en la comprensión del caso y generación de posible soluciones antes de la clase práctica de dos horas en una sesión posterior.

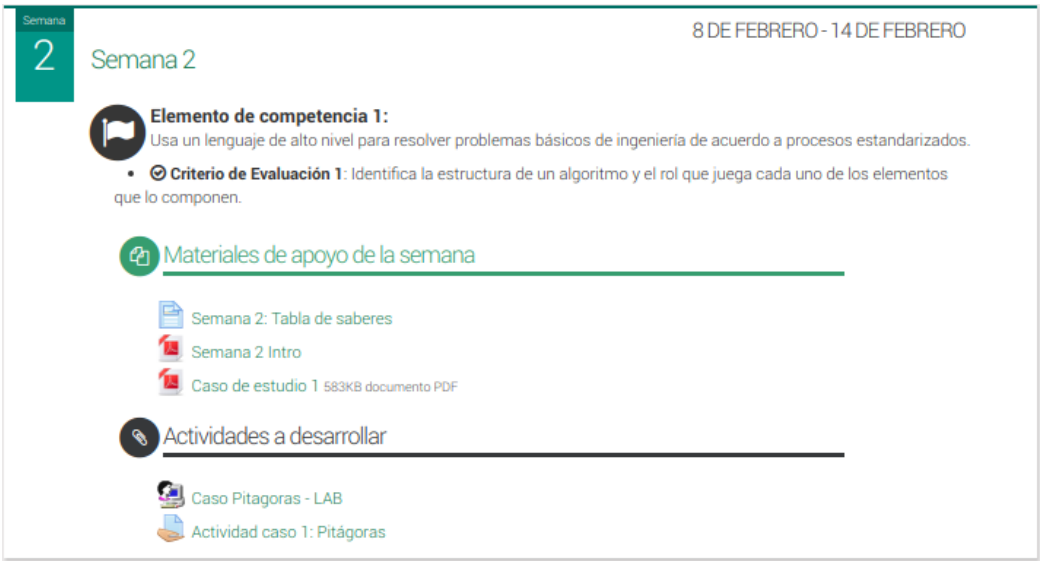
[2] Versión modificada de Moodle en la UTB

2.3. Práctica y estrategias de programación

Cada semana después de la clase magistral, se realiza la práctica en el laboratorio de dos horas y se deja planteado un trabajo adicional individual o grupal de cuatro horas fuera del aula, ayudado con Python como lenguaje de programación, y con el fin de desarrollar la capacidad de hacer y poner en práctica, se estudia la semántica y la sintaxis como una herramienta para desarrollar la solución para el caso, esta sesión de trabajo cara a cara en la computadora laboratorio, con la ayuda del profesor.

2.4. Recursos, contenidos y soporte en tecnología

Con el fin de dar soporte a las temáticas semanales y brindar materiales adicionales a los estudiantes, se implementó un curso completo con toda esta información, libros guía y contenidos adicionales que ayudan a la dedicación individual y al trabajo en equipo.
Un ejemplo del aula virtual para apoyo a la presencialidad, se puede ver a continuación:



2.5. Tabla de conocimientos semanal

Todas las semanas los estudiantes preparan para cumplir y alcanzar unos resultados de aprendizaje descritos en las competencias, a esto se suma los criterios de evaluación con que serán evaluados.

3. Implementación de los casos de estudio

El equipo de profesores desarrolló más de 26 casos de estudio totalmente integrados con la plataforma SAVIO, estos recursos fueron generados usando jupyter de python (Pérez, et al., 2007) con excelentes comentarios de la comunidad. En general la estructura de un caso de estudio contiene los siguientes elementos:

3.1. Contexto del caso

El caso debe ser cuidadosamente construido teniendo en cuenta que se debe proporcionar suficiente información para que el estudiante comprenda el problema, las estructuras de datos

que están involucradas, lo requisitos funcionales solicitados y los resultados. Todo caso debe contextualizarse, esto implica definir muy bien cada situación y crear suficientes relaciones entre la temática abordada y los elementos que la rodean, esto influye bastante en la motivación para trabajar hacia la solución válida.

3.2. Información, variables y estructuras de datos

Usar estructuras de datos es la forma de organizar la información en la memoria del computador (variables, constantes, listas, arreglos, etc.) facilitando su manipulación y además también pueden describir relaciones entre datos e información para ser procesada de manera coherente, un conjunto de operaciones sencillas que pueden realizarse son el crear una nueva variable para almacenar datos, eliminar esa variable o su contenido, modificar la información almacenada o realizar operaciones entre diversas variables existentes, por ejemplo, operaciones algebraicas entre datos numéricos u operaciones de búsqueda entre datos complejos como cadenas de caracteres.

3.3. Requerimientos funcionales

En resumen, son operaciones específicas que se deben cumplir para entregar los resultados solicitados. Cada caso necesitará la agrupación natural de los procesos, para definir cada requisito es necesario visualizar estos grupos y así poder definir cada requisito. Normalmente, un programador genera requisitos funcionales después de analizar el caso y definir las estructuras de datos, la descripción debe ser clara y concisa, definiendo cada una de las situaciones (grupo de procesos). Un requisito funcional típico contiene un nombre único y número de serie, para nuestros casos será RF1 “Requisito funcional 1” y sucesivo, RF2, etc., cada requisito funcional mostrará un resultado final.

3.4. Requerimientos no funcionales

En la solución de problemas usando el computador los requerimientos no funcionales se refieren a comportamientos necesarios para el buen funcionamiento del programa, estos pueden variar en su implementación y seguir cumpliendo con los requerimientos funcionales previamente descritos, en este caso se refieren a las estructuras de control que permiten modificar el flujo de ejecución de las instrucciones de un algoritmo. Todas las estructuras de control tienen un solo punto de entrada y un único punto de salida. Las estructuras de control se pueden clasificar en: control secuencial, iterativo y avanzado. Esta es una de las cosas que permite que se usen en este curso principios la programación estructurada, preparando el camino para la programación orientada a objetos enseñada en cursos siguientes.

3.5. Datos de prueba

Cuando el alumno tiene la solución para un problema, es necesario probar con información real y comprobar, para ello el caso de estudio aporta un conjunto de datos para las entradas y posibles resultados, si se confirma la salida en base a las entradas propuestas, el último paso es preparar los entregables.

3.6. Entregables

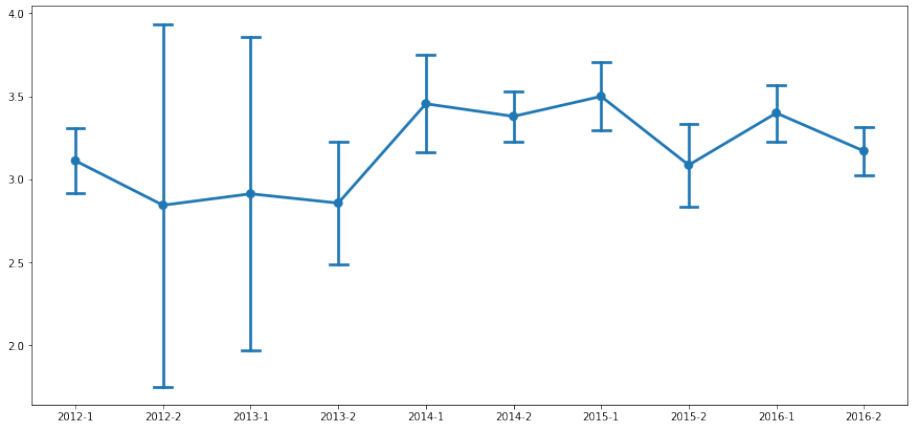
Cada caso debe generar uno o más productos que brinden retroalimentación al proceso de capacitación, estos entregables garantizarán la oportunidad de evaluar y el alumno podrá

establecer sus propios avances semana a semana, un paquete común de entregables es: código fuente o cuaderno jupyter, comentarios en el código fuente y captura de pantalla de la salida del programa o algoritmo.

4. Resultados

Podemos evidenciar 3 resultados principales: Ahora se cuenta con una guía de estudio estandarizada y muchos casos de estudio que la acompañan, se aprecia una estabilización y mejora directa de las calificaciones en todos cursos de ingeniería y se logró una optimización financiera en cuanto a los recursos invertidos en docencia directa.

Antes del rediseño en el año 2012, las calificaciones se comportan de manera irregular entre los diferentes programas académicos, debido a la diferencia entre los profesores, estilos de enseñanza y la gran cantidad de recursos no probados. Cuando el equipo desarrolla los materiales en el 2014 y unifica los procesos en una guía para todos los profesores, esto reduce la dispersión entre las calificaciones y los resultados de los estudiantes se homogenizan como se ve en la siguiente gráfica:



Además, como consecuencia de la reducción de cinco a tres horas presenciales por semana, se logró una reducción en los costos operativos y la posibilidad de atender a un mayor número de estudiantes sin disminuir los altos niveles de calidad en la universidad con los casi 400 estudiantes en promedio de grados de ingeniería cada semestre.

5. Referencias

Artículos de revistas

- Ashraf Elnagar and Mahir Ali. 2012. A modified team-based learning methodology for effective delivery of an introductory programming course. In Proceedings of the 13th annual conference on Information technology education (SIGITE '12). ACM, New York, NY, USA, 177-182. DOI=http://dx.doi.org/10.1145/2380552.2380604

- Deterding Sebastian, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From game design elements to gamefulness: defining "gamification". In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11). ACM, New York, NY, USA, 9-15. DOI=https://doi.org/10.1145/2181037.2181040
- Malik, S.I. \& Coldwell-Neilson, J. Educ Inf Technol (2017) 22: 1089. DOI=https://doi.org/10.1007/s10639-016-9474-0
- Marcia C. Linn and Michael J. Clancy. 1992. The case for case studies of programming problems. Commun. ACM 35, 3 (March 1992), 121-132. DOI=http://dx.doi.org/10.1145/131295.13130
- Perez Fernando and Brian E. Granger. 2007. IPython: A System for Interactive Scientific Computing. Computing in Science and Engg. 9, 3 (May 2007), 21-29. DOI=http://dx.doi.org/10.1109/MCSE.2007.53

Fuentes electrónicas

- ACM 2013, Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. 2013. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM, New York, NY, USA. DOI: https://dx.doi.org/10.1145/2534860

Sobre los autores

- **Gloria Isabel Bautista Lasprilla**: Ingeniera de Sistemas, Magister en Ciencias Computacionales, estudiante de doctorado en Ingeniería con énfasis en electrónica y computación. Profesora Asociada. Facultad de Ingeniería, Universidad Tecnológica de Bolívar, gbautista@utb.edu.co
- **Yuranis Henríquez Núñez**: Ingeniera de Sistemas, Magister en e-Learning, Profesora Auxiliar. Facultad de Ingeniería, Universidad Tecnológica de Bolívar, yhenriquez@utb.edu.co
- **Jairo Enrique Serrano Castañeda**: Ingeniero de Sistemas, Magister en Software Libre con énfasis en desarrollo de Software, Profesor Asistente. Facultad de Ingeniería, Universidad Tecnológica de Bolívar, jserrano@utb.edu.co