Coloquio compartiendo experiencias de enseñanza basadas en TIC

Matemáticas y programación

Mtro Juan J Carreón G, juan.carreon@gmail.com
Prof. Tit. Ing. en Comp, DIE
Responsable del proyecto **PAPIIT IN102210**, *Diseño de aplicaciones distribuidas, interactivas y gráficas*, en Android, DGAPA, UNAM.

En esta plática se presentan algunas experiencias en el uso de Tecnologías de la Información y la Comunicación, TIC, en actividades curriculares y extracurriculares, entre 2010 y 2012, con: estudiantes (Ingeniería en Computación, IC) del primer semestre, mediante tutorías; de IC en asignaturas como Algoritmos y estructuras de datos (4to semestre), Ingeniería de software (5to semestre), Lenguajes de programación (6to semestre), Temas selectos de computación gráfica (9no semestre), así como Seminario de titulación; además de Recursos y necesidades de México, con estudiantes de diversas carreras y semestres, al igual que con otros con los que se desarrollaron diversas actividades extracurriculares.

Una parte de ese conjunto de experiencias se refieren al proceso de enseñanza mediante recursos TIC en el aula; en el primer año empleando salones con proyector instalado, al que había que conectarle cada quien una laptop, sistema que significó un sistema un poco más práctico que uno previo en el cada profesor debía llevar laptop y proyector, perdiendo tiempo en la instalación y desinstalación correspondientes para cada clase. En el último año se impartió clases en salones con PC conectada a Internet y proyector, el cual si bien es menos engorroso, también es menos versátil. Por ejemplo, para algunos temas se requiere software más demandante y hardware más robusto y las PC disponibles quedan rezagadas frente a una laptop promedio.

A lo largo del último bienio, dejé de emplear grupos de noticias y listas de Yahoo, como hice a lo largo de la última década, comenzando a utilizar más simples listas

de correo y cuentas en twitter: @juancarreon para tutoría y estudiantes de ingeniería; @jcarreon, temas de ciencias sociales y humanidades, para estudiantes de ingeniería; @ayedd, estudiantes de IC; @androidjj y @androidMexico, interesados en cómputo móvil y colaboradores del proyecto **PAPIIT IN102210**, *Diseño de aplicaciones distribuidas, interactivas y gráficas, en Android*, entre otras. Con (1,616 + 8,220 + 4,925 + 3,829 + 2,494) tweets, siguiendo a (85 + 181 + 163 + 212, 186), y (138 + 260 + 273 + 259 + 488) seguidores, respectivamente. Además de otras herramientas como blogs (Ingenet) y Facebook. Mis experiencias y opinión al respecto las he plasmado en varios artículos.

La apertura de esos nuevos medios sociales permite relacionar a quienes fueron estudiantes de primer semestre hace más de cuarenta años con l@s que lo fueron en 2011, y l@s que podrían serlo en 2017, mediante expresar lo que les interesa y trabajan.

Considero que una parte de mi trabajo es aprender, enseñar e investigar, [1]entre otros temas, los relacionados con Lenguajes de programación, sus vocabularios, sintaxis, pragmáticas, usos y costumbres de empleo por sus usuarios, tanto programadores (rock-stars, ninjas, samurais, arquitectos de software, *code monkeys*, ingenieros en computación, ingenieros, y estudiantes de educación media y superior, entre otros), como usuarios finales.

Con ambos me relaciono en función de problemas como los de que en la actualidad demasiados programadores no se sienten comprometidos con comprender a fondo y con rigor los modelos en que se apoyan y generan, a pesar de que la programación nació considerándose como una rama de las matemáticas; de hecho la primera persona en escribir un programa, Ada Lovelace, se le reconoce como genio matemático.

Esa situación inicial ha dejado de ser evidente; quizá como efecto de que la industria del software ha enfatizado el conocimiento de APIs de moda, en lugar del

[1] *Diseño de aplicaciones distribuidas, interactivas y gráficas, en Android*, **Proyecto PAPIIT IN102210**, se agradece a la DGAPA, UNAM, su apoyo, sin la cual los resultados que sustentan esta investigación no se hubieran logrado.

de principios básicos; aminorándose el deseo de profundizar en modelos, además de gastar grandes sumas en impulsar metodologías de dudosa efectividad.

A ello ha contribuido, también, la falta de rigor en la formación de la mayoría de los programadores, quienes muchas veces cuentan con sólo reducidos conocimientos matemáticos, los que frecuentemente no van más allá de la aritmética. Así como un insuficiente conocimiento de la computación, lo cual los hace creer que no existe relación entre la programación y las matemáticas. [2]

El que las computadoras sean tan prácticas e indispensables en el funcionamiento del mundo moderno ha generado más de una confusión y olvido graves. La confusión tiene que ver en considerar que la computación es ciencia de la computadora; lo cual es peligroso, pues si lo fuera entonces la astronomía y la biología serían ciencia del telescopio y del microscopio, respectivamente.

No menos grave es el olvido de que la computadora fue inventada para ayudar a aclarar una cuestión filosófica concerniente a los fundamentos de la matemática. Como lo muestra la historia iniciada por el matemático alemán David Hilbert a inicios del siglo pasado, al proponer la formalización de todo el razonamiento matemático, lo cual si bien fue un fracaso, por otro lado está en el origen y desarrollo de la programación y de la computación actuales. [3]

Las matemáticas están presentes en todo código que se genera; al igual que en las relaciones entre los constructos y patrones empleados para producirlo, sin embargo, esas relaciones no son patentes debido a una insuficiente educación de los programadores que les impide formalizarlas en modelos que les permitan comprender tanto las consistencias como las inconsistencias de dichas relaciones.

De ahí que en diferentes países hayan surgido recientemente proyectos nacionales derivados de la preocupación respecto a lo que se ofrece como Computación en sus

[2] Al Swigart, How much math do I need to know to program? Not That Much, Actually, The "Invent with Python" Blog, March, 2012, http://richardminerich.com/2012/01/why-do-most-programmers-work-so-hard-at-pretending-that-theyre-not-doing-math/

[3] Gregory J. Chaitin, Ordenadores, paradojas y fundamentos de las matemáticas, Investigación y Ciencia, julio, 2003, http://www.cs.umaine.edu/~chaitin/investigacion_y_ciencia.pdf

instituciones educativas.[4] Algunos individuos y organizaciones señalan la necesidad de que todos los estudiantes universitarios aprendan a programar.[5]

En la vanguardia de estos individuos y organizaciones a lo largo de la última década están los que participan en el proyecto TeachScheme! (¡No enseñes Scheme!), logrando trascender mediante cambios en el diseño y enseñanza de los cursos introductorios de computación en la enseñanza media y superior de cientos de secundarias, preparatorias y universidades de casi todo el mundo. Al vincular la educación matemática con la computacional.[6]

Considero se requiere que los programadores comprendan aspectos como los de que: las matemáticas son excelentes para captar la esencia de un problema, con respecto al cual la programación es extraordinaria para explorar y formalizar preguntas, así como para automatizar tareas.[7]

Que las matemáticas se alimentan de la emoción de comprender cómo y por qué se relacionan las cosas; y la programación, de lo que puede lograrse.

Que un buen conocimiento matemático permite desarrollar programas sencillos y potentes; sin embargo, a veces algunos desarrollos matemáticos no son suficientemente cuidadosos, algo que nunca sucede con un buen programa.

Que los algoritmos empleados en la programación pueden provenir de cualquier rama de las matemáticas, por lo que a los programadores les conviene conocerlas en alguna medida.

Que si bien, la mayoría de los programas empleados en la vida real se apoyan en conocimientos matemáticos elementales, que muchas veces no van más allá de lo

---

[4] Gregory J. Chaitin, Ordenadores, paradojas y fundamentos de las matemáticas, Investigación y Ciencia, julio, 2003, http://www.cs.umaine.edu/~chaitin/investigacion_y_ciencia.pdf
[5] Robert Talbert, Programming vs. "Technology", Casting Out Nines, April 9, 2012, http://chronicle.com/blognetwork/castingoutnines/2012/04/09/programming-versus-technology/
[6] TeachScheme -> ReachJava, http://www.teach-scheme.org/
[7] Jeramy Kun, Thoughts after a Year of Math ∩ Programming, June 12, 2012, http://jeremykun.wordpress.com/2012/06/12/thoughts-after-a-year-of-math-programming/

de secundaria, todo programa se apoya de forma esencial e indirecta en conocimientos matemáticos sumamente elaborados.

Que programas cortos que resuelven problemas matemáticos complejos parecen pecaminosamente místicos, hasta que se invierten grandes cantidades de tiempo en comprender sus antecedentes matemáticos, luego de lo cual parecen obvios y triviales.

Que una pregunta difícil en el diseño de software es qué tanto se puede llevar una solución a sus límites, en cambio, en matemáticas una pregunta natural y sencilla es qué sucede cuando algo se lleva a sus extremos.

Que las abstracciones en matemáticas son parecidas a las de la programación; sin embargo, estas últimas pueden ser arduas; en cambio, las primeras instantáneas.

De ahí, la importancia, de tener claro tanto por legos, como por programadores tanto aficionados como profesionales, que todo programa divertido, emocionante, o productivo de forma ineludible se apoya en conocimientos matemáticos.