Live coding como técnica didáctica

Live coding as a teaching technique

Ariel Ortiz Ramírez, Tecnológico de Monterrey, México, ariel.ortiz@itesm.mx

Resumen

En este trabajo se presenta al *live coding* como una técnica didáctica útil para enseñar a programar. Se argumenta que un alumno tiene una mejor oportunidad de aprender el proceso de resolución de un problema de programación si observa cómo un experto lo hace, paso a paso y de principio a fin, en lugar de que solo se le muestre el resultado final. Así mismo, se comentan algunas prácticas específicas que pueden mejorar el impacto del *live coding* en el proceso de aprendizaje. Para terminar, se discuten las ventajas y desventajas observadas como resultado de aplicar el *live coding* en diversos cursos de programación.

Abstract

In this paper, live coding is presented as a useful technique in order to teach programming. It is argued that a student has a better chance of learning the process of solving a programming problem if they observe how an expert does it, step by step and from start to finish, rather than just showing them the final result. Also, some specific practices that can improve the impact of live coding in the learning process are discussed. Finally, there will be some remarks on the pros and cons observed as a result of applying live coding in various programming courses.

Palabras clave: live coding, programación, técnica didáctica.

Key words: *live coding, programming, teaching technique.*

1. Introducción

Los profesores que enseñábamos programación hace más de dos décadas seguramente recordamos las limitaciones que teníamos en nuestras clases cuando necesitábamos mostrar/escribir/modificar el código fuente de un programa computacional. Antes de que tuviéramos disponibles computadoras portátiles y cañones proyectores, nuestras opciones eran, por lo general, un tanto inconvenientes: pizarrón y gises/marcadores, proyector de acetatos o duplicados en papel. Por ejemplo, cuando llegábamos a escribir un programa en el pizarrón cada alumno tenía que copiarlo a mano en su cuaderno (tomarle una foto era impensable), después ir frente a una computadora en algún laboratorio y teclearlo para poderlo finalmente ejecutar. En cada etapa de este proceso se podían cometer múltiples "errores de dedo", provocando que un alumno perdiera mucho tiempo intentando comprender la razón por la que un programa no funcionaba como debía o descifrando mensajes crípticos de error.

Cuando los profesores tuvimos los medios necesarios para proyectar la pantalla de una computadora en el salón de clase, por ahí de finales de los años 90 o a inicios de este siglo, pudimos a partir de entonces experimentar con una técnica didáctica conocida como *live coding*. Este trabajo busca hacer una reflexión de cómo utilizar dicha técnica de manera más efectiva.

2. Desarrollo

2.1 Marco teórico

Se puede definir al *live coding* (codificación o programación en vivo) como un proceso que consiste en diseñar y escribir el código fuente de un programa enfrente de un grupo de alumnos durante una sesión de clase (Rubin, 2013). En tiempos recientes el *live coding* ha trascendido más allá de los cursos convencionales de computación. La Conferencia Internacional de *Live Coding* describe el alcance y nivel de inclusión que brinda esta técnica:

> El *live coding* es una práctica performativa que involucra la creación y modificación de código y algoritmos en vivo. Actualmente el *live coding* está

> en expansión como un marco de influencia para profesores, coreógrafos, programadores, compositores, psicólogos, etnógrafos, y tecnólogos, entre otros muchos, que buscan nuevas metodologías, áreas de investigación y nuevas relaciones entre tecnología, estética y programación (ICLC, 2017).

¿Por qué resulta importante el *live coding*? Haciendo una comparación con las matemáticas, Rubin (2013) explica que para enseñar conceptos de dicho dominio usualmente se resuelven problemas ejemplares de principio a fin frente a los estudiantes. Al ir paso a paso en "vivo" durante una clase, un maestro demuestra a los alumnos los mecanismos y técnicas necesarias para resolver cada problema. Pero, ¿qué sucedería si el profesor solo mostrara la solución final del problema? Con el fin de desarrollar habilidades de resolución de problemas matemáticos, ¿sería igual de efectiva la presentación de la versión estática de la solución que una demostración en "vivo"? Rubin continúa afirmando que en los cursos de introducción a la programación los instructores usualmente solo presentan versiones estáticas de ejemplos de código. Señala, específicamente, que los maestros enseñan a programar presentando a los alumnos solamente las soluciones finales completas y correctas de código. Sin embargo, ¿no será más efectivo enseñar a programar resolviendo problemas de principio a fin? Esa es la idea central detrás del *live coding*.

Por su naturaleza espontánea, *live coding* es una técnica que puede resultar emocionante pero también aterradora. La posibilidad de que el expositor cometa un error quedando mal ante su audiencia es motivo suficiente como para ahuyentar a muchos. Pero en ese sentido, el *live coding* logra captar el proceso de resolución de un problema de programación de una manera más realista que cuando solo se presentan ejemplos de códigos en su forma final.

Gaspar y Langevin (2007) señalan que los alumnos que están aprendiendo a programar batallan comúnmente con dos dificultades conceptuales, esto sin importar el enfoque pedagógico seguido (fundamentos primero, procedural primero, orientado a objetos primero, etc.) ni el lenguaje de programación utilizado. Primeramente, los alumnos tienden a programar a través de la memorización de ejemplos correctos de código en lugar de adoptar el proceso de pensamiento propio de la programación. Y segundo, tienen una tendencia a leer el código solo "de pasadita", sin comprender lo que en realidad hace. Ambas situaciones conllevan a prácticas perjudiciales, como "copiado y pegado" y "programación al azar". En otras palabras, el alumno sigue un proceso en el que busca un ejemplo similar al problema que está tratando de resolver, realiza un *copy-paste* sin entender a fondo el funcionamiento de lo que está copiando, y finalmente le hace múltiples modificaciones aleatorias al código con la esperanza de que en algún momento se obtenga el resultado esperado. Lo anterior promueve, en el mejor de los casos, la capacidad de reconocer patrones en lugar de la habilidad de resolver problemas. Por si no resulta obvio, lo anterior no es lo que deseamos promover en nuestras clases de programación. La técnica de *live coding* puede ayudar a eliminar, o al menos reducir, estas dificultades.

Gaspar y Langevin (2007) señalan que el *live coding* en clase puede ser:

- *Dirigido por el instructor.* Esta es la manera más común de efectuar el *live coding*. El profesor es el expositor y busca animar a que sus alumnos participen haciéndoles preguntas en el momento.
- *Dirigido por el alumno.* En general, los maestros no cometemos los mismos errores que los alumnos, y ésta es la principal limitación cuando el instructor es quien dirige el *live coding*. Sin embargo, si un alumno se convierte en el expositor del proceso de *live coding*, la técnica toma en ese momento un enfoque constructivista y de aprendizaje activo.

Alternar de manera frecuente entre estos dos esquemas puede hacer que la experiencia de una sesión de *live coding* sea más entretenida y enriquecedora.

El live coding se ha identificado desde hace varios años como una mejor práctica (*best practice*) para la enseñanza de la computación (Barker, Garvin-Doxas y Roberts, 2005). En el estudio elaborado por Paxton (2002) se reportó que el 80 % de sus estudiantes preferían un curso con *live coding*, 3 % preferían un enfoque más tradicional y 17 % preferían un enfoque combinado. Por otro lado, Rubin (2013) encontró en su investigación que el 90 % de los alumnos del grupo en el que se usó *live coding* tuvo una predilección hacia los ejemplos de código comparado

con el 67 % en el grupo de control en el cual el código solo fue presentado en filminas. Rubin concluye que la presentación de ejemplos de código estático no induce el mismo nivel de atención por parte de los alumnos que cuando se usa *live coding*.

2.2 Descripción de la innovación

Si definimos una técnica didáctica como "el recurso particular de que se vale el docente para llevar a efecto los propósitos planeados desde la estrategia" (Centro Virtual de Técnicas Didácticas ITESM, 2010) podemos estar de acuerdo entonces que el *live coding* es una técnica didáctica que se puede usar dentro del contexto de una *estrategia didáctica* como lo es el *aprendizaje basado en problemas* o el *aprendizaje orientado a proyectos* (Centro de Desarrollo Docente e Innovación ITESM CSF, 2012), por mencionar los que resultan más relevantes.

Para mejorar los resultados del *live coding* propongo que, además de diseñar y programar frente al grupo, se apliquen, según convenga, las siguientes prácticas concretas:

- *Utilizar un repositorio público de control de versiones de software.* Al realizar *live coding* los alumnos pueden ir siguiendo al expositor tecleando los códigos en sus equipos de cómputo portátil. Otra opción es poner atención a la exposición y quizás tomar algunas notas a mano. También es muy común que los alumnos le tomen foto con su celular al código proyectado en la pantalla. Sea cual sea la preferencia de cada alumno, siempre resulta extremadamente útil poner a la disposición de los interesados los programas hechos en clase. Para este fin es conveniente usar algún sitio como GitHub (GitHub, Inc., 2018) o Bitbucket (Atlassian, Inc., 2018). Hay dos beneficios inmediatos de esta práctica para los alumnos:
    - Los archivos fuente generados por el expositor durante el *live coding* se pueden publicar de manera fácil y en el momento en el que se desee.
    - Mientras el expositor realice *commits* frecuentes y oportunos, toda la historia de las diferentes versiones de un mismo programa elaborado en clase queda documentada para la posteridad.
- *Utilizar* Test-driven development *(TDD).* Esta práctica consiste en diseñar y escribir primero las pruebas antes de implementar el código de una unidad funcional (procedimiento, método, función). Esta es una práctica muy importante en la industria y permite enfocarse primero a entender qué debe hacer una cierta porción de código para posteriormente asegurarnos de que realmente funcione correctamente (Beck, 2002).
- *Permitir* live coding *dirigido por alumnos.* Esto se puede lograr usando un teclado inalámbrico. El alumno, sin tener que moverse de su lugar, teclea y corre programas usando la computadora del profesor (Gaspar y Langevin, 2007). Otra opción es usar un ambiente como AWS Cloud9 (Amazon Web Services, Inc., 2018) que podría describirse como el equivalente de Google Docs de la programación. Esto es, varias personas pueden simultáneamente editar/compilar/correr un mismo archivo fuente.
- *Aprovechar dispositivos móviles y salones con equipamiento especial.* Desde el 2015 los diferentes campus del Tecnológico de Monterrey cuentan con algunos salones *Media Scape* e *Innovate*, los cuales tienen múltiples pantallas proyectables a partir de computadoras portátiles y dispositivos móviles (tabletas y celulares). Estos nuevos ambientes de aprendizaje permiten que haya varias sesiones de *live coding* dirigidas por alumnos ocurriendo de forma simultánea en el mismo salón de clase. Para mantener una dinámica ágil, los profesores pueden utilizar sitios como repl.it (Neoreason, Inc., 2018) para definir ejercicios de programación que pueden ser evaluados de forma automática y de inmediato.
- *Grabar en video las sesiones de* live coding. Esto puede ser recomendable sobre todo cuando se usa de manera muy intensiva la interfaz gráfica de usuario de algún IDE. Para esto se puede utilizar software comercial como Camtasia (TechSmith Corporation, 2018) o alguna opción de software libre como Kazam (Kazam Team, 2018). Posteriormente los videos se pueden compartir a través de redes sociales o el LMS (*Learning Management System*) oficial
del curso.

2.3 Proceso de implementación de la innovación

Tal como se comentó en la introducción de este trabajo,

muchos profesores hemos estado utilizando *live coding* desde que tenemos computadoras portátiles y salones equipados con proyectores. Sin embargo el término "live coding" no es muy ampliamente difundido, ni siquiera por la gente que lo practica, y menos en los países de habla hispana. El propósito principal de este trabajo es promover al *live coding* como una técnica didáctica legítima que podamos usar de manera deliberada cuando enseñemos programación.

Personalmente, he ido incorporando las prácticas mencionadas en la sección 2.2 de forma paulatina, desde el 2001, en mi labor docente a través de los años y en diversos cursos tales como: *Fundamentos de programación*, *Estructura de datos*, *Programación avanzada*, *Lenguajes de programación*, *Desarrollo de aplicaciones web*, *Diseño de compiladores* y *Diseño y arquitectura de software*. Siempre estoy abierto a ir probando nuevas ideas y así es como he ido descubriendo y adoptando nuevas prácticas.

2.4 Evaluación de resultados

A continuación listo las ventajas que he podido observar cuando utilizo *live coding*:

1. Los alumnos se involucran más en clase cuando se les solicita que participen en las decisiones de diseño y codificación.
2. Se ilustra el proceso completo de diseño de un programa.
3. Se demuestra cómo traducir un diseño al código funcional correspondiente.
4. Se muestra el proceso de depuración completo. Cuando el expositor comete un error (ya sea de manera deliberada o accidental) se invita a todo el grupo a que participe en encontrar y corregir el problema, sin importar si fue un error de compilación o de lógica.
5. A los alumnos les resulta entretenido observar cómo un experto resuelve un problema.
6. Se muestra cómo evitar obstáculos frecuentes. Gracias a que el expositor tiene usualmente mucha experiencia programando puede compartir con su audiencia los errores más comunes antes de que ocurran.
7. Obliga a que los alumnos piensen seriamente en una solución antes de poderla ver. Esto permite que los pros y contras de diversos diseños propuestos por los alumnos sean evaluados.

Para conocer la percepción de los alumnos presento a continuación un puñado de comentarios anónimos tomados directamente de los resultados de mis encuestas de opinión de alumnos (ECOAs) desde el semestre agosto-diciembre de 2015 hasta el semestre enero-mayo de 2018. En todos estos cursos más del 50 % de las sesiones presenciales de clase las llevé a cabo utilizando la técnica de *live coding*.

- *La manera que tiene de enseñar es la mejor porque se adapta a cualquier forma de aprendizaje que tenga cualquier alumno, desde lo visual hasta lo didáctico, todo lo que enseña lo guarda y lo publica, lo cual hace que estés 100 % enfocado a lo que está mostrando.*
- *Uno de los mejores profesores que he tenido. Me hizo sufrir mucho durante su curso pero su método de enseñanza es verdaderamente bueno que siento que en una clase de él he aprendido más que en toda la carrera.*
- *Sus métodos para enseñar a programar son muy eficientes, los mejores a mi parecer.*
- *El profesor es bastante claro, da muchos ejemplos de cómo hacer el programa, e incluso te muestra cómo es que funcionan los programas detalladamente en la pantalla.*
- *Es de los únicos profesores que se toma el tiempo de estructurar sus clases y se preocupa por que aprendamos. En particular esta clase podría definir mi carrera profesional en el futuro.*
- *Su clase es muy amena. Aprendes y te diviertes, aspecto que en muchos profesores han olvidado: cuando te diviertes aprendes mucho mejor, a parte que te agrada más la clase como tal.*

Llegan a haber también comentarios negativos, aunque son menos frecuentes:

- *La clase teórica debería ser mucho más dinámica que escribir solo código.*
- *Lo único que mejoraría sería la manera de dar la clase. A veces es un poco pesada y eso hace que te aburras.*

Y con ello puedo listar las desventajas que personalmente pude observar:

1. El expositor puede llegar a cometer errores. Es importante concientizar a la audiencia de que la probabilidad de que esto ocurra es muy alta, pero que es parte del proceso de aprendizaje.
2. Hay un cierto nivel de redundancia en las clases cuando una cierta porción de código se tiene que repetir múltiples veces (por ejemplo, asociar un evento a un componente visual). Sin embargo esto genera una oportunidad para verificar si los alumnos ya entendieron cierto concepto. Cuando el concepto ha sido dominado es justificable copiar y pegar el código para agilizar las cosas.
3. A menudo se tienden a revisar conceptos fundamentales de programación que los alumnos ya dominan de cursos anteriores y esto hace que el ejercicio resulte un tanto tedioso y/o aburrido.

3. Conclusiones

El *live coding* es una técnica didáctica que, llevada a cabo de manera adecuada, puede facilitar el proceso de la enseñanza de la programación. Muchos profesores hemos usado esta técnica por años, pero existen ciertas prácticas que pueden ayudarnos a ofrecer un mejor servicio a nuestros alumnos. Debemos tener presente también que existen variados apoyos de índole tecnológica que podemos utilizar para hacer el proceso de *live coding* todavía más eficiente y atractivo para nuestros estudiantes.

Es común que como maestros nos sintamos cómodos dirigiendo una sesión de *live coding*, sin embargo puede resultar muy enriquecedor el permitir que los alumnos tomen un rol más activo en este proceso. Al final debemos recordar que aunque nuestra labor sea enseñar, lo que realmente importa es el aprendizaje que se llevan ellos.

Referencias

Amazon Web Services, Inc. (2018). AWS Cloud9 *A cloud IDE for writing, running, and debugging code* Recuperado de https://aws.amazon.com/cloud9/.

Atlassian, Inc. (2018). *Bitbucket*. Recuperado de https://bitbucket.org/.

Barker, L., Garvin-Doxas, K. y Roberts, E. (2005). What can computer science learn from a fine arts approach to teaching? *Proceeding of the 36th ACM Technical Symposium on Computer Science Education SIGCSE'05*, 421-425. doi: 10.1145/1047124.1047482

Beck, K. (2002). *Test Driven Development: By Example*, Boston, Estados Unidos, Addison-Wesley Professional.

Centro de Desarrollo Docente e Innovación ITESM CSF (2012). *Estrategias Didácticas.* Recuperado de http://micampus.csf.itesm.mx/rzmcm/index.php/tutorials/2012-09-12-14-41-19.

Centro Virtual de Técnicas Didácticas ITESM (2010). *Características de una Técnica Didáctica*. Recuperado de http://sitios.itesm.mx/va/dide2/tecnicas_didacticas/caract_td.htm.

Gaspar, A. y Langevin, S. (2007). Restoring "Coding with Intention" in Introductory Programming Courses. *Proceedings of the 8th ACM Conference on Information Technology Education SIGITE'07*, 91-98. doi: 10.1145/1324302.1324323

GitHub, Inc. (2018). *GitHub*. Recuperado de https://github.com/.

ICLC (2017). *Conferencia Internacional de Live Coding 2017*. Recuperado de http://iclc.livecodenetwork.org/2017/index.html.

Kazam Team (2018) *Kazam Screencaster*. Recuperado de https://launchpad.net/kazam.

Neoreason, Inc. (2018). *repl.it – Online REPL, Compiler & IDE*. Recuperado de https://repl.it/.

Paxton, J. (2002). Live Programming as a Lecture Technique. *Journal of Computing Sciences in Colleges*, *18*(2), 51-56.

Rubin, M. (2013). The Effectiveness of Live-Coding to Teach Introductory Programming. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education SIGCSE'13*, 651-656. doi: 10.1145/2445196.2445388

TechSmith Corporation (2018). *Camtasia – Video Editor & Video Editing Software*. Recuperado de https://www.techsmith.com/video-editor.html