Introducción de la programación concurrente en el primer curso de programación

Adolfo Di Mare
Escuela de Ciencias de la Computación e Informática
Universidad de Costa Rica
adolfo.dimare@ecci.ucr.ac.cr

RESUMEN

Se presenta una forma simple y directa de incorporar los fundamentos de programación concurrente en el primer curso de programación, sin que esta modificación curricular signifique una carga académica adicional excesiva. Además se discute el impacto que este cambio curricular tiene en el docente universitario.

Palabras clave: Programación concurrente, diseño curricular, programación por ejemplos, introducción de técnicas de programación, software.

ABSTRACT

We present a simple and straightforward way to include the most important concurrent programming techniques in the first programming course, without having this curricular change impose an excessive additional academic burden. The impact that this curriculum change has on the university professor is also discussed.

Keywords: Concurrent programming, curricular design, programing by example, introduction to programming techniques, software.

1. INTRODUCCIÓN

Es una realidad que los seres humanos vivimos tomando decisiones en cada momento. Hacer algo siempre significa dejar de hacer muchas otras cosas más. Por eso, cuando necesitamos hacer más en la misma cantidad de tiempo tenemos 2 caminos: dejar de lado lo que antes considerábamos necesario o inventar una forma diferentes de acción que permita lograr lo mismo pero con menos recursos. A los segundo lo llamamos tecnología, y las ideas que se presentan aquí nacen de aprovechar la tecnología para mejorar la productividad del proceso de enseñanza aprendizaje. Conviene comenzar por una propuesta curricular básica como la que se presenta en (DiMare-1997) o la de la ACM (*Association for Computing Machinery*) para el primer curso de programación, "*CS1*" (CC2001).

Desafortunadamente, no pasó siquiera la mitad de la década que termina en el 2010 y ya no fue posible continuar duplicando en cada nuevo ciclo de construcción la velocidad de los procesadores (HPG-2002). Para aumentar el rendimiento y la potencia de las máquinas actuales ya no es posible simplemente adquirir otros equipos: ahora es necesario programarlos mejor. Por eso, para aprovechar las nuevas arquitecturas de varios núcleos es necesario contar con programadores más educados, quienes conozcan y dominen las técnicas de programación concurrente y distribuida (talvez antes de dos décadas los compiladores y sistemas operativos le incluyan la concurrencia a los algoritmos automáticamente, pero antes de llegar a ese paraíso soñado hay que vivir la realidad actual). Por eso es necesario que los profesionales y académicos de la computación conozcan las técnicas de programación

concurrente, como también es un hecho que la mayor parte de los docentes, quienes tienen la misión de liderar el proceso que lleve a la asimilación de estas tecnologías, son personas que encuentran muchas razones para posponer la asimilación de estos nuevos conocimientos. Ese es el temor a lo desconocido que muchas veces impide alcanzar los objetivos.

El problema inmediato que hay que resolver es determinar cómo convencer a los maestros para que aprendan y enseñen programación concurrente. Además de mostrar un enfoque apropiado para lograr esta meta, aquí también se indica cómo puede el docente lograr cambiar su enfoque paradigmático para incorporar las nuevas tecnologías sin que eso engorde innecesariamente su curso. Es necesario que la actitud del maestro sea positiva para que trate de buscar la forma de usar concurrencia desde el principio; aquí se exponen ideas que muestran que esa tarea no es tan complicada como parece, pero después de todo solo es el mismo profesor el que puede tomar la decisión de adoptar la concurrencia porque no es usual en los ambientes académicos obligar a digerir el nuevo conocimiento.

2. APRENDIZAJE EN EL PRIMER CURSO DE PROGRAMACIÓN

La gran mayoría de los estudiantes han interactuado mucho con computadores, pues todo el tiempo usan aparatos electrónicos como música MP3, teléfonos celulares, cajeros automáticos, etc. Enfrentarlos a problemas de programación es más fácil y por eso sí es posible cubrir más materia en menos tiempo.

CS1: PROGRAMACIÓN I
Objetivo: Proveer al estudiante la formación básica en programación para su adecuado desempeño en los cursos subsiguientes de la carrera, fomentándole sus habilidades generales para la resolución de problemas.
Contenidos: Algoritmos y estructuras de datos, bifurcación, iteración, recursividad, entrada y salida, clases y objetos, herencia, polimorfismo, jerarquías funcionales y procedimentales, excepciones, clases contenedoras, ordenamiento, concurrencia y sincronización, documentación, prueba de programas, herramientas, depuración.

Figura 1

En la Figura 1 se muestra el programa del primer curso de programación, en el que están la mayor parte de los conceptos fundamentales que incorporan los lenguajes contemporáneos más importantes. El tema de la concurrencia aparece como un tópico más; a ningún estudiante le causará molestia alguna verlo ahí incluido.

Si el docente escoge bien los ejemplos utilizados a lo largo del curso, la mayor parte de sus alumnos los asimilarán; notarán que algunos conceptos son más difíciles que otros: eso es lo normal cuando se estudia cualquier cosa. Por eso, lograrán asimilar todos esos conceptos básicos durante el curso. En muchos casos conviene incorporar las siguientes ideas al impartir un curso en el que la programación concurrente es un tópico relevante:

- Use un ambiente de desarrollo Java adecuado.
- Los alumnos deben hacer al menos 5 proyectos programados por ciclo lectivo.
- Con el fin de que los estudiantes tengan la oportunidad de resolver sus dudas inmediatamente imparta al menos un 20% de lecciones en el computador.
- Utilice las buenas ideas de la programación extrema (Beck-2000): ponga una pareja de alumnos por máquina e intercambie el rol del que maneja el teclado.
- Para motivar a los estudiantes use al menos un programa en que se necesiten usar gráficos o juegos (KP-2001).

- Programe usando una biblioteca preconstruida y muestre cómo el uso de concurrencia permite aumentar el desempeño.
- Exponga tanto en forma teórica como práctica la teoría relevante.

La técnica "*MapReduce*" expuesta en (DG-2004) es defendida por muchos como una alternativa muy simple para implementar concurrencia. Su uso en el primer curso de programación es saludable si ya se cuenta con algunos ejemplos que los estudiantes pueden utilizar.

Varios ejemplos específicos de programación concurrente están disponibles en la red, por ejemplo en (Hartley-1998).

3. CÓMO VENCER EL TEMOR DEL PROFESOR

El temor a lo desconocido es un sentimiento natural que nos protege de acciones que puedan producirnos daño. Por eso, al emprender una nueva aventura es necesario controlar el temor. Para los jóvenes la programación concurrente no representa un gran reto, pues están acostumbrados a lidiar con muchos tipos de máquinas programables como su teléfono celular, su MP3 personal, el cajero electrónico del banco, etc. Pero para el profesor, quien ha consolidado un arsenal educativo para los cursos que imparte, aprender un tema como éste puede ser abrumador.

Sin embargo, no es la primera vez que los maestros debemos enfrentar un enfoque paradigmático diferente. La OOP, Programación Orientada a los Objetos, también causó revuelo en la comunidad académica, que encontró muchas razones para no adoptarlas. Si se examina el temario de un curso de programación para los años sesenta o setenta, es difícil encontrar referencias a los conceptos tan comunes como herencia o clase. En los pasillos de las universidades se escuchan muchos argumentos contra la adopción temprana de técnicas de programación concurrente: es demasiado duro, requiere de equipo especializado, faltan herramientas, el concepto es abrumador para los alumnos que están acostumbrados a pensar secuencialmente, etc. Todas estas apreciaciones son incorrectas, en especial la última, pues los seres humanos vivimos en un mundo altamente concurrente (¡podemos hablar por teléfono celular mientras conducimos el auto!), mientras que obligar a una persona a pensar secuencialmente es una de las grandes barreras que debe aprender cualquier programador. Afortunadamente, la experiencia incorporando OOP en los cursos de programación es muy útil para mostrarnos el camino para hacer lo mismo con los conceptos de concurrencia. Veremos en el mediano plazo que ambas experiencias son similares.

Hace falta que el docente, conscientemente, cambie su actitud. En lugar de encontrar todas las razones por la que no es posible hacerlo, más bien conviene pensar en como sí se puede lograr. Una buena estrategia de acción puede incorporar varias de estas ideas:

- Use un libro de texto adecuado.
- En lugar de construir toda la solución, agregue nuevas partes conforme las vaya obteniendo. No hace falta llegar al final pronto, vaya paulatinamente.
- Cuente con el apoyo de un estudiante avanzado como asistente: muchas veces el joven puede ayudar mucho a que el viejo aprenda.
- Programe algunos ejemplos en equipo con su asistente.
- No cambie de curso didáctico; manténgase dentro del plan original.
- No permita que el temor de aprender lo domine y comparta sus avances con sus colegas.
- Use ejemplos simples ya disponibles; use algunos juegos sencillos.

4. HERRAMIENTAS DISPONIBLES PARA JAVA

La primera decisión que debe tomar el docente es la plataforma que usará para la enseñanza. Java es generalmente la elección, pues es un lenguaje simple y muy completo; además, incorpora suficientes construcciones sintácticas para crear programas que usen hilos concurrentes en un ambiente bastante controlado. Otros lenguajes también

son buenos candidatos como primer lenguaje de programación, pero la popularidad de Java también es producto de la buena cantidad de ambientes de desarrollo disponibles (IDE: *Integrated Development Environment*); varias de estas plataformas de desarrollo son soluciones de código abierto, lo que las hace más atractiva a los universitarios. Una rápida búsqueda en la red arroja la siguiente lista de IDE's para Java:

Yahoo ➔ **`http://search.yahoo.com/search?n=100&p=IDE+Java`**

Google ➔ **`http://www.google.com/search?num=100&as_q=IDE+Java`**

`http://www.bluej.org/`

The aim of BlueJ is to provide an easy-to-use teaching environment for the Java language that facilitates the teaching of Java to first year students.

`http://www.netbeans.org/`

The only IDE you need! Runs on Windows, Linux, Mac OS X and Solaris. NetBeans IDE is open-source and free.

`http://www.drjava.org/`

DrJava is a lightweight development environment for writing Java programs. It is designed primarily for students, providing an intuitive interface and the ability to interactively evaluate Java code.

`http://sourceforge.net/projects/tide/`

tIDE is a full featured opensource Java integrated development environment (IDE), distributed under the GNU General Public Licence (GPL).

`http://www.eclipse.org/downloads/moreinfo/java.php`

The Eclipse IDE for Java Developers contains what you need to build Java applications.

5. EJEMPLOS PERTINENTES

La cantidad de tiempo disponible para cada curso en la carrera es limitada; para aumentar la cantidad de material cubierto es necesario eliminar algo o mejorar el proceso de enseñanza - aprendizaje para hacerlo más eficiente. La utilización de las herramientas modernas de enseñanza, en las que destaca la posibilidad de mostrar mediante un proyector la interacción del programador con el ambiente de construcción de programas, facilitan cubrir más material en cualquier curso de programación. Sin embargo, no es posible enseñar todos los conceptos de programación concurrente en el primer curso de programación, lo que obliga a determinar cuáles temas son los más apropiados o relevantes.

Una manera de escoger cuáles temas de programación concurrente impartir es examinar un buen libro de texto como (Downey-2008). Semáforos, el paradigma Productor - consumidor y el modelo *Map/Reduce* (DG-2004) se destacan por las siguientes razones:

Semáforos

Desde el principio estos inteligentes datos han servido para lograr la sincronización de procesos (Hoare-1985); no es posible ignorarlos.

Productor - consumidor

Además de que este patrón es muy conocido, existen muchos ejemplos muy simples de su uso. También es importante mostrarle a los alumnos las condiciones de error que se pueden producir en programas concurrente mediante estos ejemplos.

Map/Reduce

Pese a que el ideal es simplificar los conceptos al máximo, a veces no es posible hacerlo, pues existen muchas cosas que no son simples. Pese a la complejidad inherente a la realidad, algunos paradigmas ayudan mucho a disminuir el esfuerzo intelectual necesario para entender y aprender. Muchos ejemplos de concurrencia se pueden modelar usando *Map/Reduce* que es lo que hace conveniente para mostrarlo en el aula, pues es muy natural que un nodo o proceso maestro se encargue de partir tarea total y luego la asigne a los subnodos o subprocesos la porción que debe solucionar concurrentemente.

Al final de este documento se encuentran la referencia para obtener estos ejemplos, a sabiendas de que cada profesor debe determinar cuál material es mejor para su curso. Esta propuesta temática solo es un primer paso en la dirección correcta; no es razonable cubrir todo sobre sincronización y concurrencia cuando los estudiantes se enfrenten por primera vez a la programación, como tampoco cabe sugerir que se puede eliminar completamente de la carrera el curso de Programación Concurrente.

El docente debe contar con ejemplos de programación concurrente que le sirvan para introducir superficialmente los conceptos y las destrezas de esta nueva tecnología en el primer curso de programación. Estos ejemplos no pueden ser completos y exhaustivos, pues será siempre necesario contar con un curso formal en el que se afinen todos los conceptos teóricos y prácticos de programación concurrente, pero esto no impide visitar estos conocimientos pronto para que luego no les resulten extraños a los estudiantes. De nuevo, una nueva búsqueda en la red permite encontrar material disponible:

Yahoo ➔ **`http://search.yahoo.com/search?n=100&p=Java+concurrent+example`**
Google ➔ **`http://www.google.com/search?num=100&as_q=Java+concurrent+example`**

`http://www.exampledepot.com/egs/java.util.concurrent/FixedWorkQueue.html`
A work queue is used to coordinate work between a producer and a set of worker threads. When some work needs to be performed, the producer adds an object containing the work information to the work queue. One of the worker threads then removes the object from the work queue and acts upon the information.

`http://www.doc.ic.ac.uk/~jnm/book/book_applets/concurrency/carpark/CarPark.java`
The carpark holds a maximum of four cars. Further arrivals are blocked when the carpark is full.

http://www.coderanch.com/t/437544/Threads-Synchronization/java/Thinking-Java-s-Concurrency-chapter
What the program does is simulate a line of customers queueing in front of a number of teller machines. Each customer is served by a teller for a certain amount of time. Additional customers get in line while the programming is running, and the number of active tellers adjusts itself to the total number of customers (in the TellerManager class).

`http://www.cs.bris.ac.uk/Teaching/Resources/MR09/cw/`
The exercises go alongside the lectures to help you practice Java programming, and later to understand concurrency. You are strongly recommended to do each exercise first, and then read the comments afterwards, which are intended to help you assess your program.

`http://www.cs.drexel.edu/~shartley/ConcProgJava/parallel.html`
The animated quick sort example.

Es muy productivo aplicar la técnica didáctica común de usar ejemplos para mostrar conceptos difíciles. Cada educador tiene su estilo propio, pero es fácil concluir al examinar esta lista que sí es posible encontrar ejemplos para todos los gustos. También es necesario contar con un buen libro de texto que describa los conceptos teóricos fundamentales, como por ejemplo (Hartley-1998). En especial, es muy gratificante utilizar el pequeño libro de los semáforos para obtener ideas de cuáles programas concurrentes es interesante construir (Downey-2008). Además, es importante mencionar el texto clásico "*Principles of Concurrent and Distributed Programming*" del reconocido autor Ben-Ari, (Ben-Ari-2005).

6. APRENDIZAJE EN EL SEGUNDO CURSO DE PROGRAMACIÓN

Talvez no es posible comenzar a enseñar programación concurrente en el primer curso, en cuyo caso vale la pena hacerlo en el segundo. El gran inconveniente que presenta esta estrategia es que los estudiantes ya no verán la programación como algo natural, sino que necesitarán salir del paradigma secuencial que ya aprendieron. Eso les resulta extraño pues, por un lado tiene que hacer un gran esfuerzo para aprender a pensar secuencialmente pero luego tienen que cambiar y pensar en paralelo: ¡eso confunde a cualquiera! En aquellos programas académicos que se continúa usando el mismo lenguaje el desafío es menor, pues basta hacer después lo que no se hizo antes.

Si en el segundo curso se usa un lenguaje diferente, por ejemplo al cambiar de Java a C++ (Str-1998), para que el esfuerzo rinda frutos es necesario contar con una biblioteca preconstruida y sólida que permita introducir en la práctica todos los conceptos. Un candidato para esto es la biblioteca *Boost C++ libraries* (Boost-2009).

7. CONCLUSIÓN

Conforme pase el tiempo surgirán cada vez más herramientas que faciliten la programación concurrente; eso ocurrió antes con la programación gráfica, la programación por objetos y con otras innovaciones tecnológicas que han surgido conforme la tecnología de computación avanza. Debido a que ya la capacidad de construcción de máquinas está llegando a sus límites (HPG-2002), la forma natural de aumentar la potencia de cómputo es la programación paralela. Esto es un hecho indiscutible, aunque posiblemente el desarrollo tecnológico recargará en los compiladores la responsabilidad de generar el código y las salvaguardas requeridas para utilizar mejor las arquitecturas distribuidas y de multi-núcleo.

Aunque se cuente con herramientas muy poderosas siempre será necesario conocer los fundamentos teóricos de la programación concurrente. La estrategia que aquí se ha expuesto sirve para facilitar la introducción de los conceptos fundamentales en el primer curso de programación, pero conforme pase el tiempo será natural que los programas universitarios cuenten con un curso avanzado de programación concurrente, el que puede situarse pronto, antes de llegar a la mitad de la carrera. Si se educa en paralelismo a los estudiantes, poco a poco el conocimiento subirá hasta los profesores. Un aspecto que queda pendiente es la distribución de conceptos de programación paralela en los demás cursos de programación, ejercicio que además de necesario es complicado.

8. AGRADECIMIENTOS

Durante el año 2008 la compañía Intel ofreció un Seminario de Programación Concurrente en el Instituto Tecnológico de Monterrey, México, al que el autor junto con otros 3 colegas tuvo la oportunidad de asistir. Este trabajo es uno de los subproductos de ese esfuerzo. Además, varios académicos han colaborado con este trabajo, especialmente Ileana Alpízar y Maureen Murillo. Alejandro Di Mare y José Sánchez Salazar aportaron no solo observaciones y sugerencias sino que colaboraron en la construcción de varios programas de ejemplo.

9. CÓDIGO FUENTE

➔ **http://www.di-mare.com/adolfo/p/cs1cp/cs1cp.zip**
➔ **http://www.di-mare.com/adolfo/p/cs1cp/en/index.html**

BIBLIOGRAFÍA

(Beck-2000) Beck, Kent: *Extreme Programming Explained*, Addison-Wesley, 2000.

(Ben-Ari-2005) Ben-Ari, Mordechai: *Principles of Concurrent and Distributed Programming*, 2cnd edition, Addison Wesley, 2005. http://stwww.weizmann.ac.il/g-cs/benari/

(Boost-2009) Boost: Boost C++ Libraries, 2009. http://www.boost.org/

(CC2001) Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM): *Computing Curricula 2001 project (CC2001) Final Draft*, Diciembre 15, 2001. http://www.computer.org/education/cc2001/final/index.htm

(DG-2004) Dean, Jeffrey & Ghemawat, Sanjay: *MapReduce: Simplified Data Processing on Large Clusters*, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, Diciembre, 2004. http://labs.google.com/papers/mapreduce-osdi04.pdf

(DiMare-1997) Di Mare, Adolfo: *Propuesta para mejorar el curso Principios de Informática*, Reporte Técnico ECCI-97-01, Escuela de Ciencias de la Computación e Informática; Universidad de Costa Rica; 1997. http://www.di-mare.com/adolfo/p/princinf.htm

(Downey-2008) Downey, Allen B. *The Little Book of Semaphores* 2cnd ed., Green Tea Press, 2008. http://www.greenteapress.com/semaphores/

(Hartley-1998) J. Hartley, Stephen: *Concurrent Programming: The Java Programming Language*, Oxford University Press, Marzo 1998. http://www.cs.drexel.edu/~shartley/ConcProgJava/

(Hoare-1985) Hoare, C. A. R.: *Communicating Sequential Processes*, 1st ed Prentice Hall International, 1985-2004. http://www.usingcsp.com/cspbook.pdf/

(HPG-2002) Hennessy, John L. & Patterson, David A. & Goldberg, David: *Computer Architecture: A Quantitative Approach*, 3rd Edition (The Morgan Kaufmann Series in Computer Architecture and Design), ISBN 978-1558605961, Morgan Kaufmann, 2002.

(KP-2001) Kaiser, Claude & Pradat-Peyre, Jean-François: *Chameneos, a Concurrency Game for Java, Ada and Others*, 2001. http://cedric.cnam.fr/PUBLIS/RC474.pdf

(Str-1998) Stroustrup, Bjarne: *The C++ Programming Language*, 3rd edition, ISBN 0-201-88954-4; Addison-Wesley, 1998. http://www.research.att.com/~bs/3rd.html

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.