Software Conscientes para la Educación

Paul Leger
Centro de Investigación Avanzada en Educación (CIAE)
Departamento de Ciencias de la Computación (DCC)
Universidad de Chile
pleger@dcc.uchile.cl

RESUMEN

El uso de la computación en la educación es ampliamente conocido y transversal a la vida de un estudiante [1]. Varios tipos de software han sido propuestos para asistir el aprendizaje del estudiante [2, 6]. Por ejemplo, asistir a un estudiante de primaria a través un animado tutorial cuando él comete errores al resolver adiciones. Sin embargo, estos tipos de software poseen fuertes limitaciones de modularidad para evolucionarlos, por ende, no pueden ser fácilmente modificados para incorporar nuevas asistencias para los estudiantes. Otro tipo de software, conocido como *software consciente* [3], permite asistir a los estudiantes sin las anteriores limitaciones. Este tipo de software permite a los desarrolladores de manera *modular* implementar el funcionamiento del software y las asistencias necesitadas por un estudiante en un momento dado. En este póster, nosotros proponemos el uso de software conscientes para asistir a los estudiantes en sus procesos de aprendizaje. Así, estos software conscientes podrán evolucionar/mejorar sus asistencias (incluso, en tiempo real) de manera modular. Para validar nuestra hipótesis, nosotros planeamos desarrollar varios software conscientes para estudiantes. Nosotros usaremos componentes de diversas áreas de la computación como programación orientada a aspectos e inteligencia artificial para desarrollar estos software.

PROBLEMA

Es posible encontrar actualmente dos tipos de software asociados a la educación: *a)* los software que solamente pueden cumplir la tarea para lo cual son diseñados (ej. mostrar un conjunto de adiciones y registrar las respuestas del estudiante) o *b)* los software, que además de lo anterior, asisten a los estudiantes cuando ellos presenten ciertas dificultades y/o habilidades para trabajar. El segundo tipo de software es conocido como *context-aware software* [2]. Aunque context-aware software presenta evidentes beneficios frente al primer tipo de software, el uso de context-aware software no ha sido ampliamente difundido debido principalmente a su complejo desarrollo. A continuación explicamos el término context-aware software y sus dificultades para implementarlo.

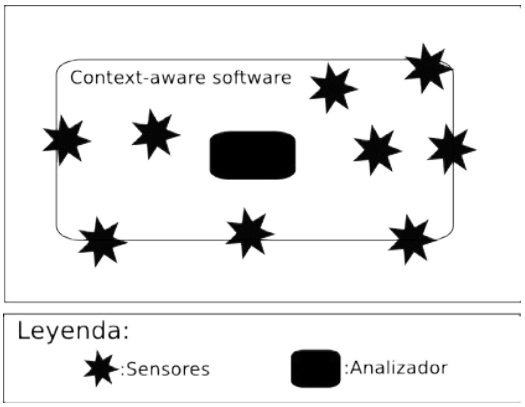

Figura 1. Las intrusivas modificaciones de un context-aware software. Los sensores son insertados de manera transversal e intrusiva a través del software.

Context-Aware Software. Como muestra la figura 1, un context-aware software es compuesto de *sensores* y un *analizador*. Los sensores son usados para capturar las interacciones de un estudiante con el software; y el analizador es usado para lanzar una asistencia al estudiante cuando cierta secuencia de interacciones es detectada. Es decir, un contex-aware software adapta su conducta para asistir a un estudiante cuando este software detecta una secuencia de interacciones que representa una habilidad o dificultad del alumno para resolver (algunas) operaciones matemáticas. Por ejemplo, si un alumno resuelve tres veces mal una misma adición, la asistencia del software puede mostrar un tutorial para enseñarle al estudiante a sumar.

El desarrollo de un context-aware software requiere modificaciones transversales e intrusivas para insertar cada sensor. Además, este desarrollo requiere que el analizador observe las interacciones de cada sensor para saber si el estudiante necesita una particular asistencia o no. Lo anterior implica que un context-aware software es complejo de implementar, mantener y evolucionar. Por ejemplo, considere que se desea cambiar el actual analizador, uno que reconoce un limitado conjunto de secuencias de interacciones, por otro analizador que puede aprender en tiempo real nuevas secuencias de interacciones y/o eliminar secuencias obsoletas. Para cambiar este analizador es necesario modificar intrusivamente el context-aware software dado que la implementación del analizador está mezclada con la implementación de los sensores y del funcionamiento del software en sí.

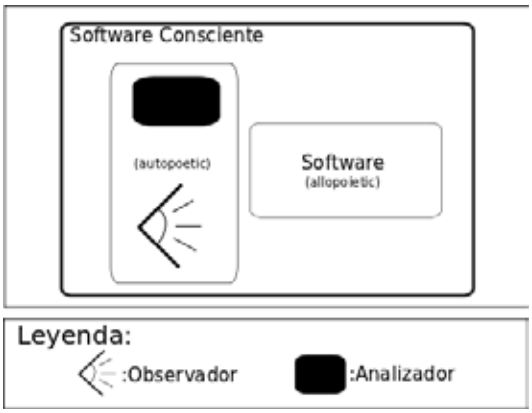

Figura 2. El modular desarrollo de un software consciente. El observador y el analizador son implementados de manera independiente al desarrollo del funcionamiento del software.

PROPUESTA

En este póster, nosotros proponemos el uso de software conscientes para asistir al estudiante en su proceso de aprendizaje. A continuación explicamos el término software consciente y sus beneficios:

Software Consciente. Un software consciente es responsable de manera modular e independiente por su funcionamiento y su futuro. El funcionamiento (alias *allopoietic)* corresponde a que el software debe realizar la tarea para el cual fue diseñado. El futuro (alias *autopoetic*) corresponde a que el software debe preocuparse de su mantenimiento y de ser útil en el futuro. Como muestra la figura 2 y a diferencia de un context-aware software, su implementación es modular. Todos los sensores son intercambiados por un *observador* de la ejecución del software. Ambos, el observador y el analizador, son implementados sin afectar el funcionamiento del software. Por ejemplo, el software consciente sí permite cambiar el actual analizador por otro que aprende en tiempo real sin afectar intrusivamente el funcionamiento del software.

VALIDACIÓN

Nosotros planeamos validar nuestra hipótesis a través de la implementación de unos software conscientes para estudiantes. Las asistencias dada por estos software hacia a un estudiante puedan evolucionar en tiempo real, es decir, a medida que el alumno use estos software. Para lograr esto, nosotros usaremos componentes de diversas áreas de la computación:

- Nosotros usaremos Programación Orientada a Aspectos (POA) [4] para observar de manera no intrusiva las interacciones del software con un estudiante (es decir, el observador). En particular, usaremos OTM [5], un mecanismo basado en POA, para observar las secuencias de interacciones del estudiante. Nosotros usaremos Inteligencia Artificial para implementar el analizador. En particular, usaremos Aprendizaje Incremental [7] para aprender nuevas secuencias de interacciones en tiempo real que representan asistencias a un alumno. Como la implementación del analizador es modular, nosotros podemos agregar asistencias para los estudiantes sin afectar el funcionamiento actual del software.

La elección de escenarios para estos software conscientes es todavía un trabajo en progreso. A pesar de esto, nosotros estamos tentados en usar software conscientes en estudiantes de primarias (K-1 a K-4) que desean aprender a sumar, restar, multiplicar y dividir. Estos software conscientes proveerán las asistencias apropiadas cuando detecten dificultades y/o habilidades por parte de los alumnos para resolver estas operaciones matemáticas. Además, estos software agregarán secuencias de interacciones a medida que son usados por los estudiantes.

REFERENCIAS

[1] Abowd, G. (1999). Classroom 2000: An experiment with the instrumentation of a living educational environment. IBM Systems Journal, 38(4), 508–530.

[2] Augusto, J. (2007). Intelligent Computing Everywhere. Springer Verlag, Chapter Ambient Intelligence: The Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence, 213 – 234.

[3] Goldman, R., & Gabriel, P. (2006). Conscientious software. Proceedings of the 21st annual ACM SIGPLAN conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), 433-450.

[4] Kiczales, G., Irwin, J., Lamping, J., Loingtier, J., Lopes, C., Maeda, C., and Mendhekar, A. (1996). Aspect oriented programming. In Special Issues in Object-Oriented Programming. Max Muehlhaeuser (general editor) et al. 74.

[5] Leger, P. & Tanter, E. (2010). An open trace-based mechanism. In Aldrich, J. and Massa, R., editors, Proceedings of the 14th Brazilian Symposium on Programming Languages, Salvador - Bahia, Brazil.

[6] Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. IEEE Personal Communications 8, 4 (Aug), 10 –17.

[7] Wang E. & Kuh A. (1992). A smart algorithm for incremental learning. In Proceedings of Int. Joint Conference Neural Network, vol. 3, 1992, pp. 121–126.