Propuesta de Trabajo de Inserción Profesional

Título:

Desarrollo de un Entorno Integrado de Aprendizaje de Programación utilizando Editores Proyetivos siguiendo la didáctica de Gobstones

Alumno:
Ariel Alvarez

Director:
Ing. Nicolás Passerini

Codirector:
Ing. Javier Fernandes

Carrera:
Tecnicatura Universitaria en Programación Informática

Propuesta de Trabajo de Inserción Profesional

13 de octubre de 2015

Resumen

El lenguaje Gobstones posee una secuencia didáctica bien definida que ha demostrado ser eficaz tanto en cursos iniciales universitarios como en escuelas secundarias. En el marco de una comunidad creciente de usuarios, proponemos el desarrollo de un *Entorno Integrado de Aprendizaje de Programación* a partir de una implementación de Gobstones sobre un *Editor Proyectivo*, haciendo uso de sus cualidades intrínsecas para facilitarle al alumno la comunicación de soluciones en términos de conceptos en lugar de trabajar sobre texto plano, reduciendo así elementos superfluos que pudieran entorpecer la secuencia didáctica.

1. Introducción

En esta sección se detallan brevemente las características del lenguaje Gobstones, se determina qué se entiende por *Entorno Integrado de Aprendizaje de Programación* y se da una introducción al funcionamiento de los editores proyectivos.

1.1. El lenguaje Gobstones

Gobstones es un lenguaje de programación que fue diseñado para la enseñanza teniendo como elementos fundacionales de su filosofía: el desarrollo de la abstracción, el aprendizaje de la división de tareas y la obtención de pautas de estilo que favorezcan la posterior interpretación de los programas generados, todo manteniendo un alto grado de simplicidad conceptual y eliminando elementos superfluos que claramente complican el aprendizaje inicial. Actualmente es soportado por diversas herramientas, entre las cuales se destaca el editor [5]

1.2. Entorno Integrado de Aprendizaje

Los entornos integrados de aprendizaje[1] cubren un gran número de herramientas. En el presente trabajo al hablar de *Entorno Integrado de Aprendizaje de Programación*, de ahora en más *ILE* (*Integrated Learning Environment*), nos referiremos a una aplicación diseñada para la enseñaza de la programación, orientada a cumplir cierta estrategia didáctica y facilitar la creación de planes de estudio por parte del docente.

Es importante diferenciar al *ILE* de un *IDE* (*Integrated Development Environment*) ya que este último se enfoca en el desarrollo de programas, y por lo tanto posee funcionalidades que en ciertos contextos de aprendizaje pueden resultar perniciosas. Por ejemplo, es esperable que un *IDE* provea mecanismos de depuración de código inspeccionando el estado del programa en una ejecución secuencial. Sin embargo, en un *ILE* esto puede ser considerado pernicioso dentro de ciertas estrategias didácticas, ya que permite al alumno buscar soluciones mediante fuerza bruta en lugar de utilizar las herramientas abstractas que se intentan transmitir.

1.3. Editores Proyectivos

El término Editor Proyectivo fue acuñado por Martin Fowler en el año 2005[4], al intentar plantear un ambiente de desarrollo donde el programador pueda expresar sus ideas en términos de conceptos en lugar de texto. Lo que vemos como texto pasaría entonces a constituir una representación editable del concepto al que hace referencia (y al cual Fowler llama *representación abstracta*).

De esta manera, los conceptos del lenguaje son el dominio de los editores proyectivos, y decimos que un programa es una *representación abstracta* construida utilizando dichos conceptos. Para modificar esta representación el programador interactúa con una interfaz de usuario, llamada *representación editable*, sobre la cual la *representación abstracta* se *proyecta* en forma de texto[3].

A su vez, la *representación abstracta* puede persistirse de diferentes maneras a diferentes soportes, con lo cual se introduce la idea de *representación persistida* para hablar del formato en que guardará el programa, ya sea en una base de datos, un archivo binario, un texto con formato XML, etc.

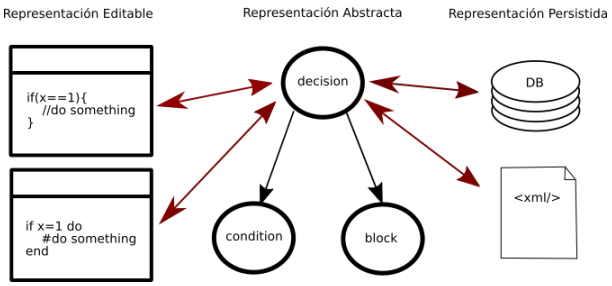

Figura 1: Representaciones de un programa en un editor proyectivo

Esto tiene varias consecuencias:

- Deja de necesitarse un parser para el lenguaje, volviéndolo más sencillo de extender.
- El programador trabaja más cerca de los conceptos que quiere expresar.
- Reduce drásticamente los posibles errores de sintaxis.
- El editor trabaja directamente con las instancias de los conceptos, con lo cual:
  - es más sencillo analizar el programa
  - se simplifica la construcción de herramientas (ej: refactors, migrado de versiones de lenguaje, intentions, etc)
  - se mejora la performance del editor al eliminarse la etapa de parseo.

2. Motivación

En el libro *"Las bases conceptuales de la Programación. Una nueva forma de aprender a programar"*[2] se presenta una secuencia didáctica bien detallada junto con el entorno *PyGobstones*[5]. Notamos que, en el marco de una adopción cada vez mayor este lenguaje, puede resultar útil complementar las herramientas existentes con otras que refuercen activamente su secuencia didáctica, acercando al alumno a las abstracciones que se desean transmitir.

A su vez, se reconoce que los editores proyectivos poseen ciertas características deseables en una herramienta de enseñanza:

- permite al alumno concentrarse en el concepto que desea expresar al reducir notablemente los posibles errores de sintaxis.
- el programa siempre se presenta al alumno con un formato estándar (bien indentado, secciones organizadas, etc), ayudándolo a incorporar más rápidamente estas buenas prácticas al reconocerlas como algo natural. En otras palabras, el alumno no posee de antemano una noción de estilos *correctos*, sino que se construye a medida que es expuesto a estos.

3. Propuesta

Se propone desarrollar un *ILE* centrado en el lenguaje Gobstones a partir de un *Editor Proyectivo*:

- que incluya una implementación de Gobstones
- que incluya la idea de proyecto y ejercicio como mecanismos de organización de código.
- que permita limitar las herramientas de lenguaje que puedan ser usadas para resolver cierto ejercicio.

- que reconozca la especificación del propósito y precondición de funciones y procedimientos (que no sean un comentario más en el código).
- que provea autocompletado, atajos de teclado y navegación de código entre otras funcionalidades comunmente encontradas en editores (siempre y cuando no vayan en contra de la estrategia didáctica)
- que provea un mecanismo de integración con otras herramientas que puedan desarrollarse en el futuro, mediante la creación de plugins.

4. Plan de trabajo

Se organiza el desarrollo en sprints con duración de dos semanas.

- Sprint 1 Modelo puro de Gobstones. Editor que soporte comandos básicos (*Poner*, *Sacar*, *Mover*) y procedimientos sin parámetros.
- Sprint 2 Que el ditor soporte procedimientos con parámetros, variables, ciclos, expresiones booleanas y aritméticas.
- Sprint 3 Se comienza el intérprete del lenguaje. Generación de un tablero a partir de comandos básicos.
- Sprint 4 Soporte a nivel lenguaje para los tableros iniciales. Se concluye el intérprete del lenguaje.
- Sprint 5 Subproyecto de deploy, generación del entorno. Se agrega soporte para estructura de proyecto y ejercicio.
- Sprint 6 Ejercicios configurables con nivel de lenguaje.

Referencias

[1] Peter Brusilovsky. «Intelligent learning environments for programming: The case for integration and adaptation». En: *J. Greer (ed.) Proceedings of AI-ED'95, 7th World Conference on Artificial Intelligence in Education*. International Center for Scientific y Technical Information, Moscow, Russia, 1995, págs. 1-8.

[2] Pablo E. Martínez López. *Las bases conceptuales de la Programación: Una nueva forma de aprender a programar*. La Plata, Buenos Aires, Argentina, 2013. ISBN: 978-987-33-4081-9. URL: `http://www.gobstones.org/bibliografia/Libros/BasesConceptualesProg.pdf`.

[3] Markus Voelter y col. «Towards User-Friendly Projectional Editors». En: *7th International Conference on Software Language Engineering (SLE)*. 2014.

[4] M. Fowler. *Language Workbenches: The Killer-App for Domain Specific Languages?* URL: `http://martinfowler.com/articles/languageWorkbench.html`.

[5] Gobstones. *Qué es Gobstones*. URL: `http://www.gobstones.org/?page_id=50`.