Propuesta de Trabajo de Inserción Profesional

Título:
Implementación de un IDE para el lenguaje Gobstones utilizando tecnología proyectiva en MPS Workbench

Alumno:
Ariel Alvarez

Director:
Ing. Nicolás Passerini

Codirector:
Ing. Javier Fernández

Carrera:

Tecnicatura Universitaria en Programación Informática

Propuesta de Trabajo de Inserción Profesional

10 de octubre de 2015

1. Introducción

1.1. El lenguaje Gobstones

Gobstones[2] [hablar sobre gobstones]

1.2. 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 pudiera 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.

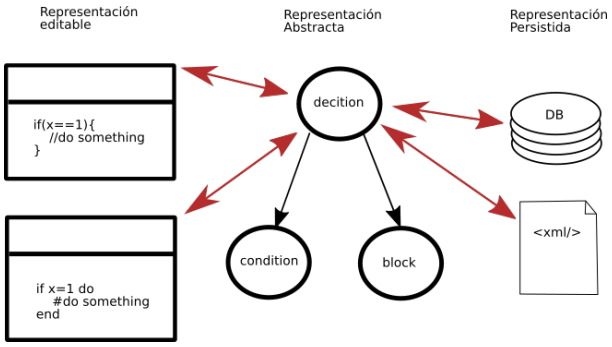

Figura 1: Representaciones de un programa en un editor proyectivo

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.
- Deja de haber errores de sintaxis como tal.
- 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.

1.3. MPS Workbench

[1]

2. Problemática

3. Propuesta

4. Plan de trabajo

Referencias

[1] Markus Voelter y col. «Language modularity with the MPS language workbench». En: *Software Engineering (ICSE), 2012 34th International Conference* (1905). DOI: 10.1109/ICSE.2012.6227070.

[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.