Wollok – y una frase que describa la idea

Nicolás Passerini   Javier Fernandes

October 13, 2014

Abstract

In this context... We consider this problem P... P is a problem because... We propose this solution... Our solution solves P in such and such way.

1 Introduction

Enseñanza tradicional: - Muchas veces se parte del lenguaje - Bajos niveles de aprobación - Pasa mucho tiempo hasta que un chico puede hacer un programa "real", demasiados conceptos. - ... después pienso más

Nosotros propusimos: - Elegir un recorrido que permite ir incorporando los conceptos uno a uno - Tener una herramienta que da soporte a eso - Focalizar en objeto-mensaje-polimorfismo-referencias, los demás conceptos aparecen después.

Also these hindrances reduce the opportunity of students to apply the concepts of the paradigm effectively in their further professional practice, resulting in several IT-projects not taking advantage of the possibilities offered by the potential of good object-oriented practices, even in cases where the tools used may allow the application of object-oriented programming gracefully. [4]

Secondly, better environments have become necessary. Earlier introductory courses focused on the development of algorithms in procedural or functional languages. To do this, an editor and a compiler was all that was needed for the practical part of the work. Modern courses now use object-oriented languages and subject material taught includes testing, debugging and code reuse. This creates the need to deal with multiple source files and multiple program development tools from the very start. To give a beginning student a chance to cope with this increased complexity, better environment support is needed. [3]

We propose to provide the student a reduced and graphical programming environment in which the object and the message are the central concepts instead of defining classes and then instantiate them. [2]

(b) lo que aprendimos en estos 8 años haciendo eso, que nos lleva a querer darle una vuelta más.

Si bien todo esto que se nos ocurrió en su momento fue maravilloso y genial y los pibes aprenden más, programan mejor y la ponen más seguido, fuimos aprendiendo más cosas y estamos trabajando ne unos cambios:

Los fáciles: - integrar clases y objetos en un mismo programa - integrar interfaces de usuario automáticas - utilizar herramientas avanzadas para guiar a los alumnos en el proceso de aprendizaje, la vedette acá sería el sistema de tipos. - acercar la práctica de lo que hacemos a la práctica industrial (acá hay que ver qué decimos y qué no, por ejemplo me molesta que no sea un archivo... pero desde lo metodológico se puede pensar en tests o incluso en un repositorio de código... sobre la relación con la industria se podría escribir un libro, hay que ver cuánto queremos meternos).

No sé si hablar de los temas de colecciones, son un poco particulares de Ozono.

(c) las ideas nuevas... y wollok como herramienta para dar soporte a esas ideas.

Lo concreto que hicimos es tener un lenguaje con - clases y objetos integrados - un IDE profesional con syntax highlighting, refactors, autocompletion y la vedette (?) inferencia de tipos (en progreso). - bueno, muchas mejoras a nivel lenguaje, como literales para coleciones, imports - una forma fácil de construir tests.

2   Problem Description

This means students must dive right into classes and objects, their encapsulation (public and private data, etc.) and methods (the constructors, accessors, modifiers, helpers, etc.). All this is in addition to mastering the usual concepts of types, variables, values, and references, as well as with the often-frustrating details of syntax. Now, add event-driven concepts to support interactivity with GUIs! As argued by [11], learning to program objects-first requires students grasp "many different concepts, ideas, and skills. . . almost concurrently. Each of these skills presents a different mental challenge." [1]

Ozono hereda muchos de los problemas de los entornos Smalltalk

Smalltalk, however, lacks other important facilities: no visualisation tools for class relations are available. The main problem with this lies in the Smalltalk language itself: since it is not statically typed, it is not possible to extract usage relations from its source code. No indication exists before runtime as to the call relationships between classes. Inheritance relationships as shown in the browser do not present the relationships of one application but rather the whole Smalltalk environment and so the browser is not used as an application modelling tool. Smalltalk blurs the distinction between the environment and the application under development. Reports about the use of Smalltalk systems for teaching also point to another problem: its size. While the language itself (in terms of the number of constructs) is small, the class library and tools are large and often confusing. Several authors reported difficulties with the students ability to cope with the environment [7, 8], especially that experimentation and self directed learning was not working well because students were overwhelmed by the system. They also found that the functionality of the browser should be limited, since its power and flexibility caused more problems than it solved. [3].

3 Proposed Solution

Wollok is a complete new tool based on the same fundamental ideas that were present in Ozono and LOOP. In particular:

- **Incremental concepts introduction**: references, objects, messages, polymorphism, classes, inheritance.

- **General purpose**: meaning not tied to any specific domain (e.g.: robots)

Besides this we also tried to address some other concerns detected while using Ozono for several years. Here's a list of main cathegories or lines of work in which Wollok extends the previous work

- **Profundizar y pulir el highlighting the conceptos primarios y la estratificacion de conceptos**. (ej: literales de objetos, literales de colecciones. Objetos no como un elemento de la IDE -Ozono: nueva referencia global-, sino como un elemnto del lenguaje. Evita referencias globales.)

- **Introducción de nuevos elementos concretos que explicitan conceptos ya existentes** (ej: 1- var/val, 2- la idea de hacer un effect system power que detecte efecto de lado, y asi poner checkeos para resolver el problema de si un método es una 'orden' o una 'pregunta', 3- program/libreria/test, 4-override).

- **Unificar las fases del aprendizaje** (ej: objetos+clases: un solo lenguaje, misma herramientas, poder reutilizar y hacer convivir)

- **Proveer un entorno inteligente que**: por un lado, estructure en forma más estricta/explícita la experiencia; y que, por el otro lado, permita una gran asistencia al estudiante/desarrollador (esto tiene muchos elementos: 1- desde content assist, 2-syntax coloring, 3- resaltado de errores (sintaxis y tipado) 4-navegación de código, 5-busqueda de referencias, 6-diagramas automáticos de clases, 7-hasta llegar un sistema de tipos que permita la detección temprana de errores, 8-reducir errores frustrantes: se cancela la edicion por tener 1 solo editor de metodo por ves (poder visualizar más que un sólo método simul), evitar errores de imagenes)

- **Acercar la experiencia de aprendizaje a las prácticas industriales**: (acá el palo de que la imagen sólo existe en smalltalk, y en la industria nadie la usa. Atrás de eso, la idea de archivos, y poder compartir con SVC. Por último la idea de actualizarse a un lenguaje con influencia de lenguajes modernos como xtend, scala, ruby, etc.)

4 Discussion

5 Related Works

LOOP is presented as a visual environment to teach OOP using a reduced set of language constructions and a prototype approach to create objects. It presents the main

concepts of object, message and reference in a specialized tool with a visual representation of the object environment. Several visual tools to teach programming already exists, like ObjectKarel[2], Scratch[14] and Etoys[4]. ObjectKarel presents a visual tool based on the abstraction of robots to teach OOP, using a map where the robots-the objects- move when messages are sent to them. LOOP does not center on a specific abstraction like a robot: it allows the student to create any other abstraction. Scratch and Etoys, are aimed to teach the basics of programming to children, using visual objects and scripts to play tween objects. This kind of diagrams could be inferred from the evaluation of any piece of code, even the execution of tests. Another subject of research is a "debugger" for the tool [1]. We think that a live and powerful debugger a ' la Smalltalk is a rich tool for the understanding of the whole environment behaviour. After a message is sent, a debugger view can be used like a video player, with play, forward and backward buttons to navigate the message stack and see how the state changes after each message send in the object diagram. Finally, there are some improvements to be made to the user interface, such as shortcuts, code completion, improved menus or internationalization. Currently the tool is only available in spanish, we want to make it configurable to add more languages as neces- sary.

6 Conclusion

Y a futuro le agregaríamos: - Integración con la UI - Integración fácil con SCM

Another characteristic of programming in the real world is the need to work in teams. The success of object-oriented languages is partly due to their advantages in group projects. Ideally, we also want to teach our students about the techniques needed for teamwork. To do this, it is essential that the environment has some form of support for group work. [3]

- Más refactors y mejora del sistema de inferencia. - Una versión web / versión liviana con el objetivo de poder ejecutarse en las netbooks que tienen los chicos de secundaria.

Probar la nueva herramienta en entornos educativos.

Eso pensando el lenguaje/herramienta, si pienso a nivel docencia se me ocurre que lo que tenemos que hacer es integrarnos con otras entidades, como Sadosky u otras universidades. Haciendo foco en que el punto no es la herramienta sino que tenemos que repensar cómo enseñamos a programar.

References

[1] S. Cooper, W. Dann, and R. Pausch. Teaching objects-first in introductory computer science. *SIGCSE Bull.*, 35(1):191–195, Jan. 2003.

[2] C. Griggio, G. Leiva, G. Polito, G. Decuzzi, and N. Passerini. A programming environment supporting a prototype-based introduction to OOP. In *Proceedings of*

the International Workshop on Smalltalk Technologies, IWST '11, pages 5:1–5:5, New York, NY, USA, 2011. ACM.

[3] M. Kölling. The problem of teaching object-oriented programming, part i: Languages. *JOOP*, 11(8):8–15, 1999.

[4] C. Lombardi, N. Passerini, and L. Cesario. Instances and classes in the introduction of object oriented programming. DC - FCEN - UBA, Buenos Aires, Argentina, Dec. 2007.

A   Images

Imágenes y otros detalles de wollok que no entran en las 6/7 páginas del artículo