A cool title

October 9, 2014

Title that Describes the Contribution that Solves a Problem O. Thor C. O. Ottohr October 9, 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

Quiero tener reglas de reescritura, para poder transformar código, por ejemplo para hacer refactors, para tener buscadores de más alto nivel, browsear código más fácil simplificar todas esas tareas dejar una API que permita hacerlo programáticamente

Nico ►*Seguro acá tenemos que hablar de the refactoring tool. Contar para qué sirve, qué cosas buenas tiene y por qué no termina de cerrar (entiendo que es porque es compleja de usar).*◄

Currently Pharo has a rewrite tool... basado en patterns... bla bla. For matching, the tool provides the possibility to indicate a pattern, and the ability to specify throw symbols different meta variables for matching. The real power of the tool is the number of variants that admit, making it really flexible.

In order to win flexibility, the patterns seems quite complex to write from scratch. The real power for matching derives in the combination of symbols, but as the patterns are represented by strings it seems quite confusing. Many times it's easier to think in examples before writing the pattern

Now we have a very powerful tool but quite complex to use... The pattern's definitions ends up been confusing and lot of people dismiss the tool for this.

hay una herramienta desarrollada, tiene algunos problemas, hay un blog sobre el tema. No estoy seguro creo que vos también decís algo de los ejemplos, Flamel se enfoca en los ejemplos, estoy bien? También se puede decir que definiste una UI para simplificar el ... refactoring?

Nico ►*Traje acá un montón de ideas de tu blog. Esto debería ser mucho más breve, veamos qué queda acá y el resto se mueve a otro lugar más adelante.*◄ una forma sencilla de definir búsquedas y transformaciones programáticamente búsqueda y transformación basados en ejemplos (utilizando una herramienta visual para manipular los ejemplos

It would be great to have a very expressive api for what we want to do, for example make possible to say something similar to:

"some scope" pattern includesStatement: anStatement; hasTemporaryVariable: aVariable. We should write some nice examples of usage and also take a look into the rewriting engine, we could do lot of improvements.

select a method and drop into the example text editor. manipulate the ASTs nodes in the text editor to make simple define transformations and matching. highlight the matched parts from the example. select scopes to apply the rule. preview the changes. show the transformation in the example expression. modify an expression throw the example or writing the pattern. output the smalltalk valid code that produces the effect we see in the example.

Paper structure...

2 Problem Description

Context, exposed with the **most precise terms possible** (don't open unwanted doors for the reader)

Probably set the vocabulary before to cut any misinterpretation

Constraints that influenced the solution (because the solution is not universal) *e.g.* our requirements for a solution, possibly not all satisfied. They should be sound and believable. Analysis of the criteria. Imagine that you are another guy having this problem do the constraint matches yours so that you could apply the solution

no hay una forma fácil de hacer transformaciones y búsquedas genéricas orientadas a un ejémplo

The idea is very simple: We want to change some code in a programmatic way In order to do this, first we want to match the code we want and then change it.

Many times we want to match an expression having different information: about the selector, about the source code (temporary vars, statements).

Some examples: all the methods with 2 arguments, all methods called in a particular way, a method containing a specific statement, The first conclusion is that we want a flexible way for matching.

In order to do transformations sometimes we want to refer to the existing code, sometimes this expressions are very complex. My goal for this time is to develop tools to make simple writing searching patterns and easier writing transformations. Here I will write about my GSoc project: Better rewriting rule tools implemented in Pharo Smalltalk, about the experience and progress.

3 Proposed Solution

4 Discussion

5 Related Works

Algunas notas sobre the Rewrite Tool [2]

- The rewrite tool is a parse-tree matcher
- Parse trees are created by parsing a superset of Smalltalk into the pattern tree.
- It is not a textual search and replace, the pattern `foo` will not match `self foo` but will match `foo := foo + 1`
- Pattern can be used to look for matches, but also to change the code.
- Pattern variables are identified by a backquote, for example: `‘receiver`
- There are specific variables for matching a single identifier, literals, a subtree, a statement, a list of statements.

Eelco Visser [4] ...

Cosas que se podrían leer: Unterholzner [3] propone otra estrategia para mejorar los refactorings, agregando static type inference.

Uquillas Gómez *et al.,* [**?**]

Bergel *et al.,* [1]

6 Conclusion

In this paper, we looked at problem P with this context and these constraints. We proposed solution S. It has such good points and such not so good ones. Now we could do this or that.

Acknowledgements

This work was supported by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council, FEDER through the 'Contrat de Projets Etat Region (CPER) 2007-2013', the Cutter ANR project, ANR-10-BLAN-0219 and the MEALS Marie Curie Actions program FP7-PEOPLE-2011- IRSES MEALS (no. 295261).

References

[1] A. Bergel, S. Ducasse, C. Putney, and R. Wuyts. Creating sophisticated development tools with OmniBrowser. *Journal of Computer Languages, Systems and Structures*, 34(2-3):109–129, 2008.

[2] D. Roberts, J. Brant, and R. E. Johnson. A refactoring tool for Smalltalk. *Theory and Practice of Object Systems (TAPOS)*, 3(4):253–263, 1997.

[3] M. Unterholzner. Refactoring support for Small-talk using static type inference. In *Proceedings of the International Workshop on Smalltalk Technologies*, IWST '12, pages 1:1–1:18, New York, NY, USA, 2012. ACM.

[4] E. Visser. A survey of strategies in rule-based program transformation systems. Technical Report UU-CS-2005-022, Department of Information and Computing Sciences, Utrecht University, 2005.