Product Line Variability Refactoring Tool

Fernando Calheiros
Meantime Mobile Creations
fernando.calheiros@cesar.org.br

Paulo Borba
Informatics Center - UFPE
phmb@cin.ufpe.br

Sérgio Soares
Computing Systems Department - UPE
sergio@dsc.upe.br

Vilmar Nepomuceno
Meantime Mobile Creations
vilmar.nepomuceno@cesar.org.br

Vander Alves
Lancaster University - UK
v.alves@comp.lancs.ac.uk

Abstract

With the growing academic and industrial interest in Software Product Lines (SPL), one area demanding special attention is tool support development, which is a pre-requisite for widespread SPL practices adoption. In this paper, we present FLiPEx, a code refactoring tool that can be used for extraction of product variations in the context of developing mobile game SPLs.

Keywords Refactoring, Software Product Lines

1. Introduction

The extractive and the reactive Software Product Line (SPL) [1] adoption strategies [4] involve, respectively, bootstrapping existing products into a SPL and extending an existing SPL to encompass another product. In both cases, product line refactorings [2],3] are useful to guide the SPL derivation process by extracting product variations and appropriately structuring them. They also help to assure the safety of the whole process by preserving SPL configurability [2] — the resulting SPL has at least the same instances than the initial set of independent products being bootstrapped or the initial SPL being extended.

In single system refactoring, ensuring safety and effectiveness of a refactoring process already requires automation, in practice, In the SPL context, where complexity increases due to the need to manage a high number of variants, such support becomes even more indispensable. In this context, we describe FLiPEx, a refactoring tool that implements code transformations [3] for extracting product variations from Java classes to AspectJ aspects. Aspects are used since we need a better modularization technique. The tool is built on top of Eclipse and interacts with the FLiPG tool, which integrates with Feature Model (FM) [5] tools for updating a SPL FM accordingly to code transformations, which, for example, might turn mandatory into optional features. FLiPEx has been designed and tested in the context of mobile game SPLs.

The main contribution of this paper is to describe our experience on designing and developing FLiPEx, its supported refactorings, and the associated user-centric view of the SPL refactoring process (Section 2). We also present and discuss FLiPEx's architecture (Section 3).

2. FLiPEx

FLiPEx is based on the Eclipse plugin platform and uses the Eclipse infrastructure to perform source code refactorings that extract product variations. The tool extracts code related to an application feature and modularizes it using AspectJ aspects. Besides refactoring source code, FLiPEx, interacting with FLiPG, also updates the SPL feature model and the configuration knowledge, for example by including new extracted features into the model and adding the aspect to the configuration knowledge so that when the feature is selected, the corresponding aspect will appear in the product.

We will illustrate the entire refactoring process with one of the implemented refactorings: *Move Extends Declaration to Aspect*., Consider the following example. `MainCanvas` is a class responsible for managing the graphical part of the application, the graphics depending on the API provided by each device. Depending on the API available at the devices, the `extends` clause will change.

```
import com.nokia.mid.ui.FullCanvas;
public class MainCanvas extends FullCanvas
{…}
```

or,

```
import javax.microedition.lcdui.Canvas;
public class MainCanvas extends Canvas {…}
```

The code snippet below shows the result of applying this refactoring for the first variation. The refactoring consists of checking the preconditions of the selected code above, which is represented with bold text, removing the original code, and generating the AspectJ code:

```
//core
public class MainCanvas {...}

//Nokia configuration
import com.nokia.mid.ui.FullCanvas;
public aspect NokiaCanvasAspect {
    declare parents: MainCanvas extends
         FullCanvas;
}
```

This aspect will insert the variation at the point from where it was extracted, preserving the behavior of the original product but offering the possibility of plugging in to the common code alternative variations of that feature.

The refactoring process is presented to the user in the form of a wizard that the user interacts with to provide all the information required to perform the refactoring. The user is presented with a list of suggested refactorings; after selecting the refactoring, s/he selects or creates the features to which the code to be extracted belongs, and then chooses the destination aspects and associates them with the selected features. In the previous example, all destination features will be alternative, and each aspect will define a `declare parents` clause for the `MainCanvas` class. The possibility of selecting several destination aspects to which the generated AspectJ construct is copied helps the user to develop different implementations of the same variation. When the user performs the extraction of a single feature spread throughout the code, FLiPEx provides an option to remember the features and aspects selection, thus simplifying the process of

code extraction, which is minimized to selecting the desired refactoring.

The following list summarizes the refactorings provided by the FLiPEx tool in its current version.

- Extract Before Block
- Move Field to Aspect
- Move Import Declaration to Aspect (this uses an ABC[7] extension we've developed)
- Move Interface Declaration to Aspect
- Move Method to Aspect
- Move Extends Declaration to Aspect

The implementation of six more refactorings is planned, including a family of Extract After Block (call, execution, initialization and set) refactorings, and more refined ones such as a refactoring for extracting context and moving static block to before-initialization.

3. Architecture

FLiPEx is part of the FLIP tool suite, built upon the Eclipse plugin platform. Figure 1 shows the relation between FLiPex, FLiPG, the Eclipse framework, and a Feature Model tool, currently Pure::Variants [5].

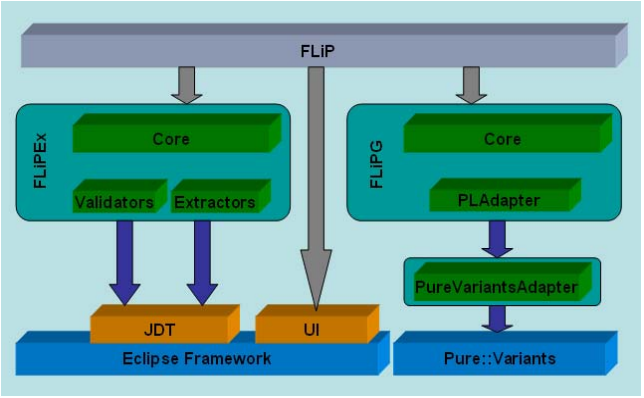

Fig. 1 Architecture

All parts of the FLiPex schema are independent plugins, *Core* is the main plugin and gives support to the *Extractors* and *Validators*. It is responsible for acquiring the selected code from Eclipse's Java editor, creating an object that contains all the information needed to perform the validation and extraction, such as the beginning and final position of the selected code, the file from which it will be extracted, and the file into which the aspect code will be written.

Each refactoring has an *Extractor*, which is responsible for removing the code from the Java class and creating the AspectJ code that inserts the variation. Each *Extractor* has a corresponding *Validator*, which is responsible for analyzing the selected code to check if it meets all the preconditions of its refactoring [3]. When a user wants to extract some code, FLiPex runs all registered *Validators* through that code, returns a list of the extractors whose preconditions have been met, and a list of the extractors that cannot be used and the reasons their *Validators* failed.

FLiPex already provides an abstract base extractor class that takes care of most of the peripheral tasks that all extractors need to perform, such as writing the Java and the aspect files, updating the imports, and removing the selected code from the Java source file. With this base extractor, the code of the concrete extractors is very small and only takes care of creating the aspect code that will be inserted in the aspect file. Additional refactorings can be added to the platform through the extractors and validators extension points that FLiPEx exports to plugin developers. Also additional feature model tools can be used with FLiP, since it exports an extension point for feature model tools adapters.

FLiPEx uses JDT's infrastructure, AST and visitors, to perform analysis on the preconditions that must be met by the Java classes and to remove the selected code from the Java class. Due to incompleteness of AJDT's [6] plugin infrastructure for AspectJ AST manipulation, code generation in FLiPEx is currently being done by string manipulation.

4. Conclusions

We have presented FLiPEx, a tool for SPL refactorings, introduced the refactorings it implements and a few that will be implemented in the future, and described its architecture. Eclipse's plugin infrastructure was of easy understanding, not only because of useful documentation, but also due to the available framework, which is easy to use. On the other hand, working with AJDT was difficult because its infrastructure for AST manipulation is not finished yet, so our code generation is hardcoded with string manipulation at this point.

As future work, we intend to develop more refactorings, integrate with other feature model tools, and improve our process of product line refactoring.

Acknowledgments

We gratefully acknowledge the other FLiP team members: Isabel. Wanderley, Geraldo. Fernandes, Andréa. Frazão and former members Jorge Pereira and Davi Pires. We would also like to thank Meantime Mobile Creations and C.E.S.A.R. for supporting the FLiP project. This research was partially sponsored by CNPq and MCT/FINEP/CT-INFO.

References

[1] P. Clements and L. Northrop. Software Product Lines: Practices and Patterns. Addison-Wesley, 2002.

[2] V. Alves, R. Gheyi, T. Massoni, U. Kulesza, P. Borba, and C. Lucena. Refactoring Product Lines. In *Proceedings of GPCE'06*, October 2006. ACM Press.

[3] V. Alves, P. Matos Jr, L. Cole, P. Borba, and G. Ramalho. Extracting and Evolving Mobile Games Product Lines. In *Proceedings of the 9th SPLC'05*. September 2005. Springer-Verlag.

[4] C. Krueger. Easing the transition to software mass customization. In *Proceedings of the 4th International Workshop on Software Product-Family Engineering*. Spain, October 2001.

[5] Pure::Variants. http://www.pureystems.com/Variant_ Management.49.0.html, 2007

[6] AJDT: AspectJ Development Tools. http://www.eclipse.org/ajdt/, 2007.

[7] abc. The AspectBench Compiler. http://www.aspecbench.org, 2007.