Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 107 (2004) 51–55
www.elsevier.com/locate/entcs

Supporting JAsCo AOP By Means of Eclipse

Davy Suvée[1], Wim Vanderperren[2], Joris Elsocht and Viviane Jonckers

System and Software Engineering Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium

{*dsuvee, wvdperre, jelsocht, vejoncke*}*@vub.ac.be, http://ssel.vub.ac.be*

Abstract

In this tool demonstration paper, we present a plugin for the Eclipse tool platform which provides support for JAsCo. JAsCo is a dynamic AOP language focused at component-based software engineering. The JAsCo Eclipse plugin provides a visual integrated development environment for JAsCo that guides developers to program JAsCo artifacts using visual wizards. The plugin also supports running JAsCo enabled applications and offers an extensive range of debugging and introspection facilities.

Keywords: JAsCo, Aspect-Oriented Programming, Eclipse.

1 Introduction

Aspect-Oriented Programming (AOP) aims at achieving a better separation of concerns than possible using current software engineering paradigms. In order to capture "crosscutting" concerns, AOP introduces a novel modularization construct, called an aspect [3]. Nowadays, several aspect-oriented approaches

[1] The author is supported by a doctoral scholarship from the Flemish Institute for the Improvement of the Scientifical-Technological Research in the Industry (IWT or in Flemish: "Vlaams instituut voor de bevordering van het wetenschappelijk-technologisch onderzoek in de industrie")

[2] The author is supported by a doctoral scholarship from the Fund for Scientific Research (FWO or in Flemish: "Fonds voor Wetenschappelijk Onderzoek")

doi:10.1016/j.entcs.2004.02.046

are available, such as AspectJ [2], HyperJ [5], Adaptive Programming [4] and Composition Filters [1]. The JAsCo AOP approach [6] has recently been introduced to overcome some of the limitations of these approaches and aims at combining the ideas behind aspect-oriented and component-based software development. JAsCo is currently supported by a set of command-line tools and a run-time infrastructure. Although these command-line tools allow a developer to unleash the full power of JAsCo AOP, the development process of a JAsCo enabled application is split up in two distinct phases as no integration is provided with the development environment in which the base application is written. To this end, we present the JAsCo Eclipse plugin, which offers an integrated development environment for JAsCo AOP, as it guides developers with writing, running and debugging JAsCo enabled applications.

In the next section, the main features of the JAsCo AOP language and technology are highlighted. Section 3 presents an overview of the JAsCo Eclipse plugin. Finally, we state our conclusions and present some topics that require further investigation.

2 JAsCo AOP Approach

JAsCo is a dynamic AOP approach that aims at combining the ideas of aspect-oriented and component-based software engineering. The JAsCo language itself is an extension of Java and stays as close as possible to the regular Java syntax and concepts. JAsCo introduces two new entities: aspect beans and connectors. An aspect bean is an extended version of a regular Java bean and is specified independent of concrete component types and APIs. An aspect bean contains one or more logically related hooks that describe the crosscutting behavior itself. The main difference with approaches such as AspectJ, is that JAsCo aspect beans are described in terms of an abstract context. A connector is used for deploying one or more aspect beans within a concrete context and allows to explicitly instantiate and initialize hooks. In addition, connectors are able to specify explicit precedence and combination strategies in order to manage the cooperation among several applicable aspects.

At the technological level, a new component model is proposed where traps that enable aspect interaction are already built-in. Each trap refers to the JAsCo run-time infrastructure that manages the registered connectors and aspect beans. For each encountered trap, the run-time infrastructure evaluates which hooks are applicable and executes their advice. Because of this dynamic architecture, it is possible to add, change and remove aspects at run-time. In order to optimize the trapped component model, it is also possible to add and remove traps just-in-time using the novel Java HotSwap functionality.

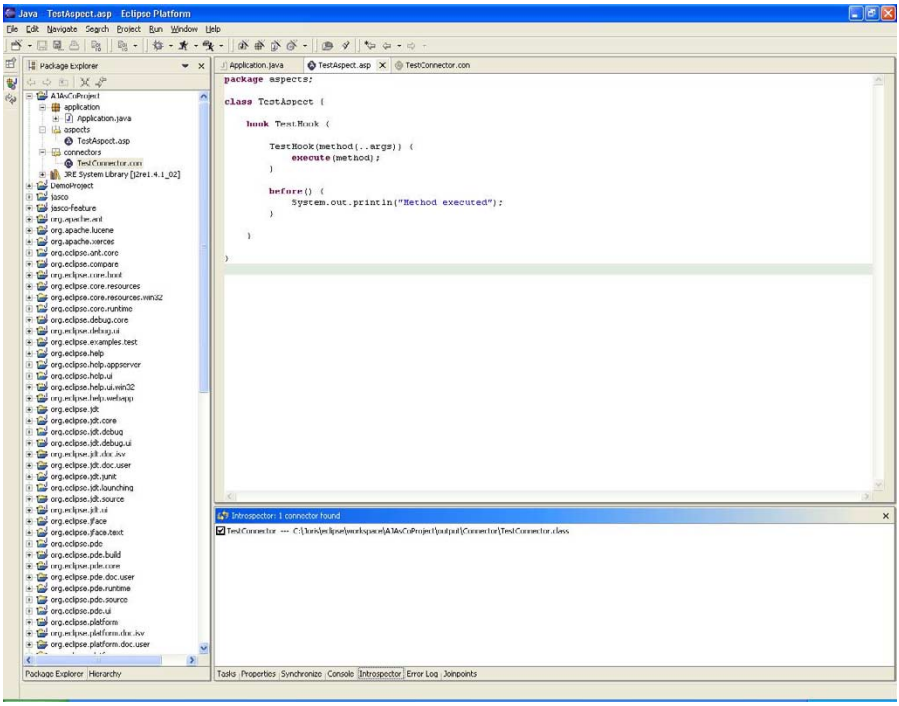

Fig. 1. The JAsCo Eclipse plugin

For more information about the JAsCo approach, we refer to [6].

3 JAsCo Eclipse Plugin Overview

In order to provide a fully featured IDE for JAsCo AOP, the JAsCo Eclipse plugin is proposed. Using this plugin, dedicated JAsCo projects can be developed and the developer is guided for implementing, running and debugging a JAsCo enabled application. The JAsCo plugin is an extension of the Java plugin, so all features provided by this plugin are also available. Figure 1 shows a screenshot of the JAsCo Eclipse plugin.

3.1 Programming support

The JAsCo Eclipse plugin provides two extensive wizards which assist a developer with describing JAsCo aspects and connectors. The aspect bean wizard is mainly used to generate a template in which the crosscutting behavior of an aspect can be described. A user specifies the name of the aspect and one or more hooks. Additionally, the developer is able to define the advice that has to be generated by default. Using this wizard, a JAsCo aspect bean is au-

tomatically generated. Only the advice, which are essentially described using plain Java, need to be filled in for obtaining an useful aspect. The aspect bean wizard also contains an extensible library of aspect beans that implement a range of typical aspects. The aspect beans are categorized using keywords and some aspect beans can also be customized for the concrete context at hand.

The connector wizard is employed for describing connectors and allows to automatically generate a complete connector. Using this wizard, a developer is able to instantiate one or more hooks. For each hook separately, the concrete context can be defined by making use of the native Eclipse type and method selection dialogs. In addition, a developer is able to specify which advice to execute as well as the precedence strategies between the instantiated hooks. As such, a developer is able to specify the order in which advice need to be executed for each connector separately. The connector wizard also contains an extensible library of combination strategies that can be integrated in the connector at hand.

The JAsCo aspect and connector wizards simplify the use of the JAsCo aspect language significantly. As a result, developers who are new to AOP and JAsCo in particular can easily adopt and integrate this technology within their projects.

3.2 Application execution support

The JAsCo run-time infrastructure provides many options which allow a developer to tune the underlying JAsCo AOP run-time framework to his/her needs. The JAsCo HotSwap functionality for example requires the Java Virtual Machine to be started differently. Using the Eclipse plugin, which implements a dedicated JAsCo configuration runner, this process is hidden for the developer. Furthermore, JAsCo offers other run-time configuration options, like for example the Jutta just-in-time compiler. As several configurations can be maintained at the same time, a developer is able to pick the JAsCo run-time environment he/she needs. Note however, that a JAsCo project is in fact an extended Java project. As a result, it can still be run as such.

3.3 Run-time introspection and debugging support

JAsCo is a dynamic AOP approach as it allows to add, change and remove aspects from the system at run-time. This kind of functionality is often required for business-oriented applications, as requirements tend to change often in such an environment. To this end, an introspector has been integrated within the JAsCo Eclipse plugin which allows one to manipulate the aspectual behavior of the application at run-time. Using this introspector, developers are able to

determine at run-time which aspects are loaded into the system. Furthermore, aspects can be changed, removed and added at run-time making use of this uniform interface.

Also, when developing an AOP application, keeping track of which aspects are applicable at which joinpoints (i.e. the concrete context of an aspect) is a difficult task. Therefore, as debugging support, a joinpoint lookup feature is included in the JAsCo Eclipse plugin which allows one to statically determine the possible concrete joinpoints of a hook instantiation.

4 Conclusions and Future work

In order to support the JAsCo AOP language and technology, a JAsCo plugin for Eclipse is proposed. The JAsCo plugin guides developers with writing JAsCo aspect beans and connectors by providing two extensive wizards. JAsCo enabled applications can easily be run using dedicated JAsCo application configurations. In addition, an introspector and joinpoint lookup tool is provided which allows one to debug a JAsCo enabled application.

At the moment, the JAsCo plugin provides support for the basic JAsCo language and technology features. JAsCo however provides many other features which are not supported by the plugin at this time. Several extensions to the JAsCo Eclipse plugin are being considered: improved syntactical checking of aspect and connector code, automatic support for moving existing Java project towards JAsCo projects and aspect-oriented refactoring support which allows one to extract crosscutting behavior out of an existing application into a set of JAsCo aspect beans and connectors.

References

[1] Bergmans, L., M. Akşit and B. Tekinerdoğan, *Aspect composition using composition filters*, in: Kluwer, editor, *Proceedings of the Symposium on Software Architectures and Component Technology: The State of the Art in Software Development*, 2001.

[2] Kiczales, G., E. Hilsdale, J. Hugunin, M. Kersten, J. Palm and W. G. Griswold, *An overview of aspectj*, Lecture Notes in Computer Science **2072** (2001), pp. 327–355.

[3] Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier and J. Irwin, *Aspect-oriented programming*, in: *Proceedings European Conference on Object-Oriented Programming*, Atlanta, USA, 1997, pp. 220–242.

[4] Lieberherr, K., D. Orleans and J. Ovlinger, *Aspect-oriented programming with adaptive methods.*, Communications of the ACM **44** (2001).

[5] Ossher, H. and P. Tarr, *Using multidimensional separation of concerns to (re)shape evolving software*, Communications of the ACM **44** (2001).

[6] Suvée, D., W. Vanderperren and V. Jonckers, *Jasco: an aspect-oriented approach tailored for component based software development*, in: *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development*, Boston, USA, 2003, pp. 21–29.