

# AJHotDraw: A showcase for refactoring to aspects

— Current Status and Invitation to Join —

## Arie van Deursen

Software Evolution Research Lab  
CWI & Delft Univ. of Technology  
The Netherlands  
Arie.van.Deursen@cwi.nl

## Marius Marin

Software Evolution Research Lab  
Delft University of Technology  
The Netherlands  
A.M.Marin@ewi.tudelft.nl

## Leon Moonen

Software Evolution Research Lab  
Delft Univ. of Technology & CWI  
The Netherlands  
Leon.Moonen@computer.org

## Abstract

*Adoption of aspect-oriented techniques in existing systems is hindered by the fact that developers have difficulties adapting to a “multi-view”, concern-based reasoning over the software systems, and by a lack of reference solutions that show how to address crosscutting functionality in real-life systems. In this paper, we argue for the need of a common showcase for the adoption of aspect-oriented techniques in existing systems and propose an open-source model application to serve this role. We present the AJHotDraw project whose goal is to take an existing well-designed open-source system (JHotDraw) and migrate it to a functionally equivalent aspect-oriented version. In addition to creating a showcase, the project aims to assess feasibility of aspect solutions in a real application, to document the crosscutting concerns in the original system and address the testing challenges that rise from ensuring behavior conservation between the original and the refactored version of the system.*

## 1. Introduction

The adoption of aspect-oriented techniques to modularize crosscutting concerns is equally beneficial for the development of new applications as well as for existing ones. The latter goal can be achieved by code refactoring: using code transformations to improve the structure of the code without altering the external behavior of the application.

However, refactoring the scattered and tangled implementations of crosscutting functionality into aspects has received limited attention compared to the development of new aspect languages (such as AspectJ,<sup>1</sup> CaesarJ<sup>2</sup>)

and frameworks that support aspect-oriented development (AspectWerkz,<sup>3</sup> JBoss AOP,<sup>4</sup> Spring AOP,<sup>5</sup> etc.). Yet, the migration of existing applications can provide a number of important benefits for the adoption of aspect-oriented techniques. The availability of comparative implementations allows for assessment of concern modularization improvements. Moreover, refactoring requires one to identify, document and organize the crosscutting concerns in a given system. This enables aspect-based modular reasoning about the system’s concerns which can help the developers to get accommodated with this new approach. Finally, recurring crosscuttings, instances of the same generic concern, can be addressed by general solutions and thereby contribute to the development of refactoring catalogs, idioms and good practices for aspect-oriented programming.

We argue that there currently is no (open-source) project that explicitly makes a goal from addressing these issues and serving as a model for adoption of aspect-oriented techniques in existing systems. In this paper, we propose to address this need for a common showcase by taking an existing well-designed open-source system and migrating it to a functionally equivalent aspect-oriented version. The contributions of this project include:

- Identification and documentation of the crosscutting concerns in the original system. The results of this step will help developers to be aware of the existence and understand the underlying implementation of the crosscutting concerns in the given system. Further, the categorization of these concerns can be used to create or extend aspect libraries, as those provided by, for example, the JBoss framework.
- Building a common benchmark for testing aspect

mining techniques and tools. A serious shortcoming in developing techniques and tools for aspect identification is the lack of a system for which the aspects are documented. The reported results of such experiments lack a basis of comparison and require investigation of the system for validation.

- Associating aspect solutions to the identified crosscutting concerns, and extract general solutions, idioms, patterns and good practices to further build confidence in adopting aspect-oriented techniques.
- Assessing the reliability of proposed aspect solutions to known crosscutting concerns in the context of a real application.
- Building a model application based on an aspect-oriented solution.
- Providing support for refactoring from an object-oriented implementation to an aspect-based solution: extract refactorings and good practices. Use the case-study to provide useful feedback to aspect languages developers for a better language support for refactoring.
- Addressing the challenges risen by testing aspect-oriented systems and by ensuring behavior conservation between the original and the refactored version of the project.

## 2. AJHotDraw

The system we propose to migrate to an aspect-oriented solution is JHotDraw,<sup>6</sup> an open-source drawing application, that was developed as model application to show good use of design patterns in Java. It was first implemented in Smalltalk (HotDraw) and then in Java (JHotDraw), both implementations aiming to provide a framework for technical and structured two-dimensional graphics editors. The fact that JHotDraw is considered a well-designed application makes it an ideal candidate as showcase for aspect-oriented migration.

AJHotDraw<sup>7</sup> is our proposed showcase for the aspect-oriented refactoring of JHotDraw. We started it to experiment with the feasibility of adopting aspect-oriented solutions in existing software and to demonstrate the strategies proposed by our research. It is written using AspectJ, an aspect language that extends Java with crosscutting functionality.

The first steps in refactoring JHotDraw have been taken in [13], where fan-in analysis was employed for identification of crosscutting concerns. The results from that

analysis initiated the refactoring of some of the identified concerns, such as *persistence* and *undo* [12]. Presently, we are migrating our refactored solutions to version 6.0 of JHotDraw. A release plan for that effort is presented in Table 1.

Since refactoring implies preserving the observable behavior of the application and the original JHotDraw system cannot without tests, a testing sub-project (TestJHotDraw) is being developed to ensure behavioral equivalence between the original and the refactored solutions. In addition, the fan-in analysis results are currently being extended through collaboration with two other research groups, as detailed in section 3.

### 2.1. AJHotDraw Organization

AJHotDraw is organized into two parts: (1) the main project is the AspectJ implementation of the system, where the identified crosscutting concerns are refactored to aspects; (2) the test subproject comprises all the test cases aimed at ensuring equivalence between the the original Java solution and the refactored AspectJ one. The aspects are put in separate packages, one per concern. Changes to the original files are restricted just to removing concerns that have been migrated to aspects.

The tests suite can be compiled with and executed on the archived binary files (jar) of any of the two solutions. Building and executing the test suite is automated using ANT.<sup>8</sup>

### 2.2. Invitation to Join

All developers interested in contributing to AJHotDraw are invited to join its development. This may include researchers from the areas of aspect-oriented software testing, refactoring, aspect mining and so on, or anyone interested in aspect-oriented development.

The present release plan is based on effort estimates for the three initiators. If others are willing to contribute, additional cross cutting concerns can be taken into account as well.

## 3. Related ongoing efforts

### 3.1. Mining for aspects in JHotDraw

In a joint effort with two other research groups that independently developed aspect mining techniques, we are in the process of making a qualitative comparison of the three techniques (identifier analysis [17], dynamic analysis [16], and our own fan-in analysis [13]). In this study, we use JHotDraw as a common case for our techniques.

Release	Date	Description
TestJHotDraw 0.1	January 8, 2005	Test suite for persistence concern of selected figures
AJHotDraw 0.1	January 15, 2005	Full JHotDraw 6.0 sources included; systematic build, test and release process implemented in ant.
TestJHotDraw and AJHotDraw 0.2	February 1, 2005	Aspect-Oriented implementation of persistence for selected figures
TestJHotDraw and AJHotDraw 0.3	February 15, 2005	Test suite for selected commands
TestJHotDraw and AJHotDraw 0.4	March 1, 2005	Aspect-oriented implementation for contract enforcement for selected commands
TestJHotDraw and AJHotDraw 0.5	March 15, 2005	Test suite for undoing selected commands
TestJHotDraw and AJHotDraw 0.6	April 1, 2005	Aspect-oriented implementation for undo for selected commands
TestJHotDraw and AJHotDraw 1.0	May 1, 2005	Consolidation release; organize publicity and involve additional people

**Table 1. Release plan for AJHotDraw.**

The results of this work will contribute to a more detailed list of documented concerns in the analyzed case-study.

### 3.2. JHotDraw as case-study

Other researchers have also used JHotDraw code to illustrate refactoring approaches ([7]). Although their primary goal was not to report or document the crosscutting concerns in the system discussed, such experiences can contribute to the set of reported concerns.

### 3.3. AspectJ in open-source projects

At the time of writing, there are very few open-source projects developed in AspectJ. To our knowledge there is only one project that was stated with an intended goal of showing good practices in AOP: aTrack<sup>9</sup> is a bug tracking application developed in AspectJ, intended to provide support for technical, middleware and business concerns. However, the state of the project is unclear; it started in November 2003 and no code has been released yet.

### 3.4. Refactoring to aspects

In the area of refactoring to aspects, significant attention was given to aspect solutions for a number of typical crosscutting concerns [10, 8, 11, 1]. However, these solutions are not integrated into a large application, but only in small cases serving as examples.

Another research direction investigates how aspect refactorings can be organized in catalogs, based on code transformations from Java to AspectJ specific modularization units [15, 14]. The approach describes steps in a feature extraction process, emphasizing the mechanics associated to code transformations, and follows the format

used by Fowler [5] to describe object-oriented refactorings. Iwamoto and Zhao [9] also propose a number of refactorings, but without providing details about any of the specific cases. Their attention tends to focus on potential conflicts between the aspect refactorings and the traditional, object-oriented ones. This issue is also addressed by Hanenberg *et al* [6].

Coady *et al* investigate the benefits of aspect-oriented solutions for evolving operating system code and for better managing its variability [3, 2, 4]. Although their work aims at assessing the benefits of aspect-oriented software development, it has not led to a publicly available aspect-oriented and non-aspect-oriented version of the same OS, which would enable comparative experimental software evolution research by others.

## 4. Concluding remarks

In this paper, we argued for the need of a common showcase for the adoption of aspect-oriented techniques in existing systems and propose an open-source model application to serve this role. We presented the AJHotDraw project whose goal is to address these issues by taking an existing well-designed open-source system (JHotDraw) and migrating it to a functionally equivalent aspect-oriented version.

There are a number of benefits to this approach:

1. comparing the legacy implementation with the new version helps developers understand the challenges of migration to aspects and shows them how this problem can be tackled;

2. the migration requires identification of crosscutting concerns in JHotDraw which provides us with a benchmark for various aspect mining techniques and tools;
3. dealing with recurring instances of the same concerns contributes to the development of refactoring catalogs, idioms and good practices for aspect-oriented programming.

Consequently, we believe that the project will contribute to a gradual and safe adoption of aspect-oriented techniques in existing applications, and allow for a better assessment of aspect orientation.

Future work concerning this project includes the identification of additional aspects in JHotDraw and further migration of JHotDraw to AJHotDraw.

## References

- [1] The AspectJ Team. *The AspectJ Programming Guide*. Palo Alto Research Center, 2003. Version 1.2.
- [2] S. Bray, M. Yuen, Y. Coady, and M. E. Fiuczynski. Managing variability in systems: Oh what a tangled OS we weave. In D. Beuche, K. Czarnecki, M. Mezini, C. Schwanninger, and M. Voelter, editors, *Managing Variabilities Consistently in Design and Code (MVCDC OOPSLA04)*, 2004.
- [3] Y. Coady and G. Kiczales. Back to the future: A retroactive study of aspect evolution in operating system code. In Mehmet Akşit, editor, *Proc. of 2nd Int. Conf. on Aspect-Oriented Software Development (AOSD)*, pages 50–59. ACM Press, March 2003.
- [4] Y. Coady, G. Kiczales, M. Feeley, and G. Smolyn. Using AspectC to improve the modularity of path-specific customization in operating system code. In *Proc. of 8th European Softw. Eng. Conf. (ESEC/FSE)*, pages 88–98. ACM Press, 2001.
- [5] Martin Fowler et al. *Refactoring: Improving the Design of Existing Code*. Addison Wesley Professional, 1999.
- [6] S. Hanenberg, C. Oberschulte, and R. Unland. Refactoring of aspect-oriented software. In *Proc. of Net.ObjectDays Conference*, pages 19–35. Springer-Verlag, 2003.
- [7] J. Hannemann, Murphy G.C., and Kiczales. G. Role-based refactoring of crosscutting concerns. In *(to appear) Proceedings of International Conference on Aspect-Oriented Software Development*, 2005.
- [8] J. Hannemann and G. Kiczales. Design pattern implementation in Java and AspectJ. In *Proceedings of the 17th Annual ACM conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 161–173. ACM Press, 2002.
- [9] M. Iwamoto and J. Zhao. Refactoring aspect-oriented programs. In *4th AOSD Modeling With UML Workshop, UML’2003, San Francisco, California, USA*, 2003.
- [10] R. Laddad. Aspect-oriented refactoring. [www.theserverside.com](http://www.theserverside.com), December 2003.
- [11] R. Laddad. *AspectJ in Action - Practical Aspect Oriented Programming*. Manning Publications Co., 2003.
- [12] M. Marin. Refactoring JHotDraw’s Undo concern to AspectJ. In *Proc. of First Workshop on Aspect Reverse Engineering (WARE)*. Delft University of Technology, 2004.
- [13] M. Marin, A. van Deursen, and L. Moonen. Identifying aspects using fan-in analysis. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004)*. IEEE Computer Society Press, 2004.
- [14] M.P. Monteiro. Catalogue of refactorings for AspectJ. Technical Report UM-DI-GECS-200401, Universidade do Minho, 2004.
- [15] M.P. Monteiro and J.M. Fernandes. Object-to-aspect refactorings for feature extraction. In *Industry paper, 3rd International Conference on Aspect-Oriented Software Development*. University of Lancaster, UK, 2004.
- [16] P. Tonella and M. Ceccato. Aspect mining through the formal concept analysis of execution traces. In *Proc. of the 11th IEEE Working Conference on Reverse Engineering (WCRE 2004)*, Delft, The Netherlands, November 2004. IEEE Computer Society.
- [17] T. Tourwé and K. Mens. Mining aspectual views using formal concept analysis. In *Proc. of the Fourth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2004)*, Chicago, Illinois, USA, September 2004. IEEE Computer Society.

## Web References

- <sup>1</sup> [aspectj.org](http://aspectj.org)
- <sup>2</sup> [caesarj.org](http://caesarj.org)
- <sup>3</sup> [aspectwerkz.codehaus.org](http://aspectwerkz.codehaus.org)
- <sup>4</sup> [jboss.org/developers/projects/jboss/aop](http://jboss.org/developers/projects/jboss/aop)
- <sup>5</sup> [springframework.org](http://springframework.org)
- <sup>6</sup> [jhotdraw.org](http://jhotdraw.org)
- <sup>7</sup> [sourceforge.net/projects/ajhotdraw/](http://sourceforge.net/projects/ajhotdraw/)
- <sup>8</sup> [ant.apache.org](http://ant.apache.org)
- <sup>9</sup> <https://atrack.dev.java.net/>