# Assisting Aspect-Oriented Framework Instantiation: Towards Modeling, Transformation and Tool Support

Marcilio Mendonca
University of Waterloo
Canada
marcilio@csg.uwaterloo.ca

Paulo Alencar
University of Waterloo Canada
palencar@csg.uwaterloo.ca

Toacy Oliveira
PUC-RS
Brazil
toacy@inf.pucrs.br

Donald Cowan
University of Waterloo
Canada
dcowan@csg.uwaterloo.ca

## ABSTRACT

Aspect-Oriented (AO) frameworks improve a framework-centered development process by providing appropriate means for handling crosscutting concerns. However, the instantiation process of AO frameworks remains complex and error-prone. We propose a modeling and transformation approach with tool support to assist the instantiation of AO frameworks.

**Categories and Subject Descriptors**: D.2.2 [**Software Engineering**]: Design Tools and Techniques - *Computer-aided software engineering (CASE)*

**General Terms**: Design, Languages, Verification.

**Keywords**: Aspect-oriented frameworks, aspect modeling languages, model transformation, application frameworks.

## 1. INTRODUCTION

Application frameworks have been widely used as valuable tools to produce families of applications at a lower cost and higher quality. In fact, frameworks have been successful in representing the common and variable features of software products, thus becoming a key component in software product lines [1]. However, the traditional object-oriented approach still fails to support a clean separation of concerns commonly found in application frameworks (crosscutting concerns). As a result, aspect-oriented (AO) frameworks [2] are moving to the mainstream by taking advantage of new mechanisms (e.g., aspects) to properly handle crosscutting concerns. The combination of aspects and application frameworks is really promising and is already demanding approaches to facilitate the instantiation process of AO Frameworks. This paper summarizes our research work towards the specification of an approach to assist the instantiation process of aspect-oriented frameworks. The approach relies on UML [10] extensions to represent the AO framework design model and on transformations to produce application instances. A Case tool to support automation is also described. In the following sections we discuss aspect-oriented frameworks, our approach to framework instantiation and a case study we developed to evaluate our ideas.

## 2. ASPECT-ORIENTED FRAMEWORKS

An AO framework contains a set of classes and aspects that play together to decompose architectural elements into a more manageable set of modules. Aspects support a better separation of concerns that are normally spread throughout the design and source code in object-oriented architectures. A good example of crosscutting concerns are design patterns [3] which, in fact, are very likely to be found in a framework architecture. For instance, in the *Observer* design pattern the objects playing the roles *Observer* and *Subject* may also play other roles in the framework architecture. The use of aspects allows framework developers to proper separate object roles into distinct modules improving encapsulation and avoiding tangled code. A previous research work [5] has already proposed an aspect-based implementation of the 23 GoF design patterns [3] using the AspectJ programming language [6]. While in object-oriented frameworks hotspots are represented by abstract classes and methods, AO frameworks benefit from other abstractions to represent variability including aspects and pointcuts. Thus, during the instantiation process of AO frameworks, application developers have to provide concrete implementations for all the abstract aspects and pointcuts specified. For example, with the *Observer* design pattern, application developers have to specify the execution points (represented by pointcuts in AspectJ) in which *Subjects* will notify *Observers* about changes in their internal state.

## 3. OUR APPROACH

Our approach to aspect-oriented framework instantiation consists of a modeling language based on UML extensions, a process language for model transformation, and a Case tool (see Figure 1).

**Modeling Language-** In our approach we propose a modeling language named AF-UML that represents an effort to take advantage of current UML extensions proposals for aspects (e.g., aSideUML [9]) and frameworks (e.g., UML-FI [8]). In UML extensions for aspects framework developers can model pointucts, advices, and aspects appropriately, but there is normally no support for describing hotspots (e.g., abstract pointcuts). In contrast, current UML extensions for frameworks do not tackle aspects and its related concepts but are adequate in modeling variability. Our modeling language aims at providing adequate means to describe AO framework models, i.e., aspects and its related-concepts as well as variability. Moreover, we also plan to address model serialization issues, in particular, we aim at making AF-UML models compliant with the OMG XMI (XML Metadata Interchange) [7] format.

**Transformation-** The core of our approach relies on model transformations. In particular, we propose a process language named AF-RDL (Aspect-oriented Framework-Reuse Definition Language) that takes as input a framework model (e.g., aspect diagram) and based on inputs from application developers

*OOPSLA'05*, October 16–20, 2005, San Diego, California, USA.
ACM 1-59593-193-7/05/0010.

generates application instances models. The process language is based on the cookbook approach to framework documentation [4] and uses concepts such as design patterns and instantiation commands to increase the level of abstraction during the instantiation process. As well, AF-RDL uses the notion of recipes to encapsulate inter-related instantiation tasks. Framework developers use AF-RDL commands to specify the instantiation steps required to produce valid application instances. Subsequent, application developers run AF-RDL scripts that are provided in order to generate application instances. During the instantiation process, not only abstract classes and methods are extended but also abstract aspects, aspect methods and pointcuts.

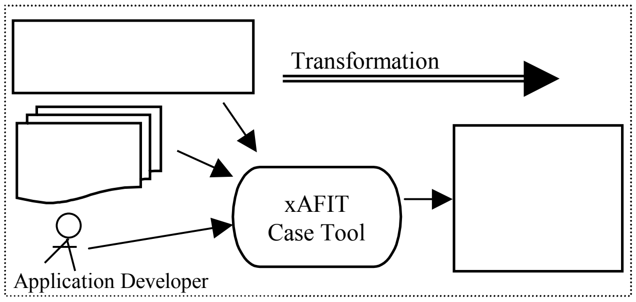

**Figure 1: An Overview of Our Approach**

**Tool Support-** A Case tool named xAFIT (XML Aspect-oriented Framework Instantiation Tool) to support our approach is under development. The tool transforms a semi-complete AF-UML aspect diagram (framework design) into a complete aspect diagram (application design) based on application developers' inputs. xAFIT provides a runtime environment for AF-RDL scripts. At the low-level, xAFIT converts AF-RDL high-level command calls into XQuery [11] user-defined function calls in order to achieve transformations over the serialized XMI models (XML format). For instance, the ASPECT_EXTENSION AF-RDL command is mapped to the xq_aspect_extension function in XQuery that takes the super-aspect name as a parameter and returns a valid XML tag that represents the new sub-aspect model element created.

## 4. CASE STUDY

We developed an initial case study to evaluate the feasibility of our approach. The idea was to produce an application instance through transformations over an AO framework design. We developed a Drawing Editor framework in AspectJ [6] which exposed 6 hotspots (CSGDrawingEditor Framework). The *Observer* design pattern was used allowing Figure objects (*Subjects*) to notify registered *Observers* about size changes (e.g., zoom in/out). The *Observer*, in our case represented by the *Display* class would handle the notifying events by repainting the drawing in the appropriate canvas window. We used an aspect version of the *Observer* design pattern (described in [5]) that exposes 2 hotspots: i) the *Display (Observer)* reaction to *Figure*'s (*Subjects*) resizing, realized by an aspect method extension, and ii) the Figure object's that should notify the *Display* about state changes (e.g., calls to the resize() method of Figure objects), realized by a pointcut extension. We represented the instantiation tasks in an AF-RDL script, mapped some AF-RDL commands to XQuery user-defined functions and performed the corresponding XQuery transformations. At the end we obtained a serialized XMI-like model representing our application instance design. Aspects, poincuts, aspect methods and advices were represented by specific model elements in AF-UML. The case study showed that the idea of mapping high-level AF-RDL commands to XQuery functions is feasible. Indeed, XQuery turned out to be a powerful language for transformation. Therefore, the combination of AF-RDL and XQuery was seen as very positive and promising.

The next steps in our research include i) enhancing the AF-UML expressiveness by defining new models for AO frameworks, ii) specifying a corresponding XMI-compliant description to all model elements in AF-UML, iii) mapping all RDL instantiation commands to related XQuery functions, and iv) developing a tool to manipulate AF-UML models.

## 5. CONCLUSION

We propose an approach to facilitate the instantiation process of aspect-oriented frameworks. The approach includes a modeling and a process language, and a Case tool. We presented a preliminary case study we performed to validate our ideas and pointed out the future directions of our research.

## 6. ACKNOWLEDGMENTS

This work has been partially supported by the Natural Sciences and Engineering Research Council (NSERC) and CAPES (Brazilian Ministry of Education Agency).

## 7. REFERENCES

[1] SEI-Software Engineering Institute – Software Product Lines - http://www.sei.cmu.edu/productlines/index.html

[2] Constantinides C., Bader A., Elrad T., Fayad M., and Netinant P., Designing an Aspect-Oriented Framework in an Object-Oriented Environment, ACM Computing Surveys, 32(1), 2000.

[3] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[4] Pree W., Pomberger G., Schapert A., Sommerlad P., Active Guidance of Framework Development, Software-Concepts and Tools (1995) 16: 94-103, Springer-Verlag.

[5] Hannemann J., Kiczales G. Design pattern implementation in Java and AspectJ. In Proceedings OOPSLA '02, ACM SIGPLAN Notices, 2002.

[6] AspectJ Project Official Website - http://eclipse.org/aspectj/

[7] XMI (XML Metadata Interchange). OMG XMI Website http://www.omg.org/technology/xml/

[8] Oliveira, T. C. Filho, I. M., Lucena, C. J. P. , Alencar, P. S. C., Cowan, D. D., Software Process Representation and Analysis for Framework Representation, IEEE Transactions on Software Engineering, March 2004, Volume 30, Issue 3, p.145-159.

[9] Chavez, Christina von Flach Garcia. A Model-Driven Approach to Aspect-Oriented Design. PhD Thesis, Rio de Janeiro: PUC-Rio, 2004.

[10] Unified Modeling Language. OMG UML Website http://www.uml.org/

[11] XQuery. W3C XQuery Website http://www.w3.org/TR/xquery/