Using Agent-Based Technology for Aspect-Oriented Development

Federico Trilnik

ISISTAN Research Institute, UNICEN University
Campus Universitario, (B7001BOO) Tandil, Bs. As., Argentina
Also CONICET

E-mail: ftrilnik@exa.unicen.edu.ar

Abstract

Aspect-oriented technologies are increasingly promoting new ways of developing software in order to better control change and improve software adaptability and evolution. However, aspect-oriented development still appears strongly attached to the underlying programming approach used to support it, rather than to design mechanisms guiding development. In this context, we propose an approach for enhancing aspect-oriented software development considering aspects as first-class design entities. This work presents an agent-based tool called Smartweaver, which allows developers to describe aspect designs using UML extensions and a special documentation method, and then provides smart assistance through a planning agent to translate these specifications into particular AOP implementation technologies.

Index Terms: Aspect orientation, frameworks, framework documentation, IA techniques, CASE tools

Introduction

Aspect-oriented technologies [5] are increasingly promoting new ways of developing software in order to better control change and improve software adaptability and evolution. Current aspect technologies such as aspect languages [4] and aspect-oriented frameworks [2, 3] make now available novel means to achieve the principle of separation of concerns. However, instead of being based on design mechanisms to guide the development, aspect-oriented development still appears strongly attached to the underlying programming approach used to support it. Supporting aspects at the design phase can greatly improve aspect-oriented development, because aspects can be identified and incorporated earlier in the development process. Therefore, tools or CASE environments assisting developers to bridge the gap between aspect-based design specifications and current AOP technologies are becoming increasingly necessary.

In this context, we propose an approach for enhancing aspect-oriented software development considering aspects as first-class design entities. The proposal puts together lines of research coming from different fields, namely: aspect-oriented frameworks, aspect models extending UML models, knowledge-driven framework documentation and agent-based planning.

The key idea of the approach is the concept of *smart-weaving*. This concept promotes essentially an early incorporation of aspects in the development cycle, so that designers would be able to specify their designs by means of aspect models, and also provide different strategies to map generic aspect structures to specific implementations. With this purpose, we have built an experimental environment called *Smartweaver* aiming to support this process. The kind of assistance provided by the tool relies on the *Smartbooks* method [6], a method extending traditional techniques for framework documentation. First, the user designs the application using aspect-oriented design models, then an agent using a specially designed UCPOP-like algorithm [9] and special rules generates a plan to implement the aspect-based application.

The Smartbooks documentation method

The *Smartbooks* method conceives the instantiation of frameworks as an activity based on a well-defined number of basic instantiation tasks, for example, class specialization or method overwriting, among others. The method prescribes that the framework designer should describe the functionality provided by the framework, how this functionality is implemented by different framework components, and provide rules to somehow constraint the ways the framework can be specialized. This knowledge is represented through what is called instantiation rules.

Smartbooks is based on agent technology [1], more precisely, it includes a special planning agent that is able to derive the sequence of activities that should be executed to implement a given functionality from a target framework. In order to do so, framework designers need to supply some information to the agent in advance. This knowledge is described through a predefined format of rules that we call the *Smartbooks* documentation method.

In particular, this method can be applied to the case of aspect-oriented frameworks. The *Smartweaver* tool takes advantage of *Smartbooks* ideas and provides an UML-based environment where developers can define classes, aspects and crosscutting relationships among them. All this information is collected by a special tool, which is also provided with adequate knowledge about how to map generic aspectual specifications into a given AOP implementation support. Then, the *Smartweaver* engine is able to combine this domain knowledge to derive an instantiation plan to implement the desired aspectual behavior (see Figure 1).

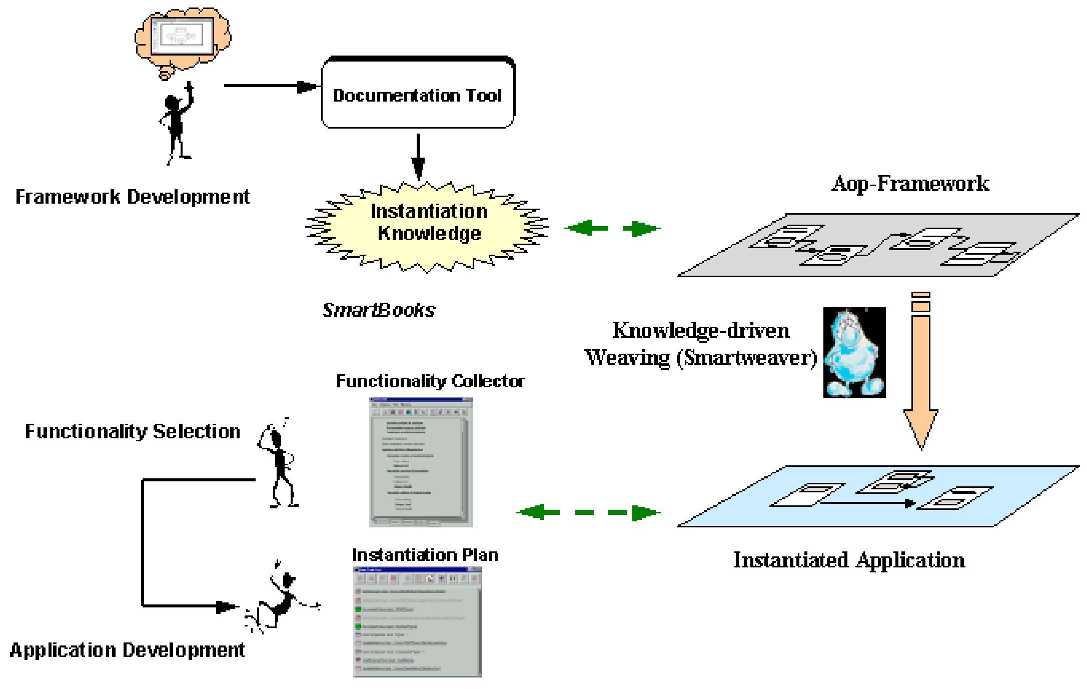

Figure 1. Assisting framework instantiation using the Smartbooks approach

Applying smart guidance

The developer/designer that uses the *Smartweaver* tool should go through a sequence of phases in order to implement their applications. The user should develop or acquire Smartbooks documentation of support technology, then he has to design the application, next he has to add smart-weaving information and, finally, the user has to follow the instantiation plan to implement his application. In the different phases, the application developer adds diverse documentation and information to the tool. All this knowledge is collected by the *Smartweaver* tool and is used by the planning agent to propose an instantiation plan to the user.

In the following, a brief description of the necessary stages is presented:

- **Building SmartBook of Support Technology:** At the beginning, the approach requires the definition of the core *Smartbooks* knowledge documenting a given aspect implementation technology to build later applications on top of it. We have developed a study case to prove these ideas. In our case study, we have used the *Aspect-Moderator(AMF)* framework [2]; therefore we had to express AMF-related knowledge in terms of functional rules.
- **Designing the Target Application:** Here, the target application is designed through the specification of core components as well as the crosscutting properties affecting component's functionality. The specification is made using both UML models and *aspects diagrams*. These aspects diagrams are extended UML models that include special stereotypes and relationships to support different AOP features [7].
- **Adding Smart-Weaving Documentation:** During this phase, developers should give more details about their previous aspects diagrams, that is, they should specify the ways crosscutting relationships and aspect interactions should be implemented, according to the aspectual support included in the first stage.
- **Putting it all together:** Finally, the application requirements list has to be generated. In this task, the Smartweaver tool uses information from the previous stages and new information provided by a special wizard in order to obtain the required functionality. That is, once aspect models have been defined using the documentation tool, it is possible to translate them to rules expressing a portion of the functionality required by the final application. Additionally, the wizard collects more user requirements on the basis of the initial available functionality, based on the instantiation knowledge previously provided by the designer. When the list of selected requirements is determined, the Smartweaver engine is able to build an instantiation plan. This plan consists of a sequence of tasks that should be carried out in order to build the application.

Conclusion and Future work

The presented approach joins different research lines, aspect-oriented frameworks, aspect models extending UML models, knowledge-driven techniques for framework documentation and agent-based planning, to assist the user in the development of aspect-based applications from aspect design models.

The *smart-weaving* notion proposes an earlier incorporation of aspects in the development cycle. The visual formalism developed to specify aspect models, as an extension of conventional UML diagrams, intends to be independent of particular AOP implementation technologies. This favors communication among aspect developers, promoting a common documentation model for aspect-based applications. In later design stages, developers are free to decide which rules will map these models into a given AOP technology, for instances *AspectJ*, *Hyper/J*[8] or the *Aspect-Moderator* framework. From this perspective, it would be possible to have a general-purpose tool, and add such aspect technologies as plug-in packages. These packages would define a corpus of Smartbooks-based knowledge to map aspect models into particular AOP technologies. This could be also complemented with wizards to collect desired functionality.

At this moment, a basic UML-compliant CASE environment integrated with the *Smartweaver* engine has been developed. This prototype currently supports a part of the features provided by commercial tools such as Rational Rose or Together, however, we are planning to improve these capabilities in the future. Regarding the *Smartweaver* engine, it still presents some limitations. Not all the instantiation actions can be derived if there is not enough knowledge available. In addition, the design of similar applications cannot be detected, loosing some opportunities to make simpler the instantiation process. For these reasons, we have started to explore the possibilities of increasing the agent's reasoning capabilities with more advanced reasoning techniques, as case-based reasoning and bayesian networks.

References

1. Bradshaw, J. *Software Agents*. AAAI Press, Menlo Park, USA. 1997
2. Constantinides C., Bader A., Elrad T., Fayad M. *Designing an Aspect-Oriented Framework*. Computing Surveys 32(1es):41. 2000.
3. Fayad M., Schmidt D., Johnson R. *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. Wiley Eds. 1999
4. Homepage of AspectJ. Xerox Palo Alto Research Center (Xerox Parc), Palo Alto, California. http://aspectj.org/
5. Kickzales G., Lamping J., Mendhekar J., Maeda C., Videira Lopes C, Loingtier J., Irwin J. *Aspect-Oriented Programming*. Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finlad. Springer-Verlag LNCS 1241. June 1997.
6. Ortigosa A., Campo M. *Towards Agent-Oriented Assistance for Framework Instantiation*. Proceedings of OOPSLA 2000, October 2000.
7. Suzuki J., Yamamoto, Y. *Extending UML with Aspects: Aspect Support in the Design Phase*. 3rd Workshop on Aspect-Oriented Programming at ECOOP'99. 1999
8. Tarr P., Ossher H. *Hyper/J User and Installation Manual.* Homepage of Hyper/J. IBM Research. http://www.research.ibm.com/hyperspace/HyperJ
9. Weld D. *An Introduction to Least Commitment Planning*. AI Magazine, Summer/Fall 1994.