How to develop secure applications with Aspect-Oriented Programming

Mónica Pinto, José M. Horcas
Dept. de Lenguajes y Ciencias de la Computación
University of Málaga, Málaga, Spain
{pinto,horcas}@lcc.uma.es
CAOSD Group, http://caosd.lcc.uma.es

***Abstract*—In the last few years several new programming techniques for achieving a better separation of concerns have been defined. One of the most accepted has been Aspect-Oriented Programming (AOP). Those attending this tutorial will learn how to improve the modularization, maintainability and evolution of secure applications by using AOP to achieve a better separation of the security concerns (e.g. authentication, authorization, encryption). A running example will be used throughout the tutorial to illustrate how AOP works. The tutorial will principally illustrate the use of the AspectJ programming language, although other AOP languages will be used to cover features that are not included in AspectJ. As a proof of concept we will discuss the use of AOP in the context of the INTER-TRUST project, a STREP project that brings together experts from the security and aspect-oriented research communities to demonstrate that security policies can be dynamically deployed and adapted at runtime using AOP.**

I. Brief summary and outline

Security has been largely identified by software engineers as a crosscutting concern [1], [2]. A *crosscutting concern* is either a functional or a non-functional concern that is tangled and/or scattered with other concerns of the application. On the one hand, a concern is *scattered* when it is not well-encapsulated in a software module. On the other hand, a concern is *tangled* with other concerns when the same software module contains more than one concern. For instance, a software module that implements part of a bank application will usually include code related to the core behaviour of the application, such as accounts, clients, automatic teller machines, etc., but also security code that appears tangled with the bank application code. Moreover, security will be needed by several software modules of the bank application, being scattered among them.

Thus, crosscutting concerns are concerns than cannot be appropriately modularized using traditional programming techniques, such as Object-Oriented Programming (OOP) or Component-Based Software Engineering (CBSE). This problem has been identified as the Tyranny of the Dominant Decomposition problem, and indicates that the software modules that are used by these traditional approaches (i.e. the objects and components), as well as their composition mechanisms (i.e. object/component communication through references between them) are not always appropriate for achieving a good separation of concerns. The consequence is that the system modularity, maintainability and evolution are negatively affected.

In the last few years several new programming techniques for achieving a better separation of concerns have been defined. One of the most accepted has been Aspect-Oriented Programming (AOP) [3]. AOP is an evolution of OOP and CBSE and it is basically based on: (1) the definition of a new software module named "aspect", which helps to improve the encapsulation of crosscutting concerns, and (2) a new composition mechanism known as "aspect weaving", which avoids direct references between objects/components. As nowadays the aspect-oriented separation of concerns is applied not only at the implementation level, where it was firstly defined, but to all the phases of the software development –i.e. from requirements specification to the testing and maintaining phases, the general term normally used is Aspect-Oriented Software Development (AOSD) [4]. The term AOP is still used at the implementation level.

An important number of AOSD proposals exist at each level of the development. At the implementation level, the most well-known and mature one is AspectJ [5]. In fact, it was the AspectJ language which initially introduced the concept of "aspect" and the fact is that the rest of proposals basically use the same terminology defined by AspectJ. Other interesting approaches in the Java world are JBoss AOP [1] and Spring AOP [6]. Moreover, there are also aspect-oriented versions of other programming environments, such as AspectC [2] for C, AspectC++ [7] and FeatureC++ [3] for C++ and Spring .NET [4] por .NET.

As previously stated, in the aspect-oriented community, security is a typical example of crosscutting concerns [2]. Thus, it is usually taken as an example to show the benefits of AOP. When we talk about the separation of the security concerns, we mean to separate security functionalities, such as authentication, authorization, privacy or encryption, from the core functionality of the applications requiring them. Thus, the primary aim of this tutorial is to introduce the audience to the use of aspect-oriented techniques as a means

[1] http://www.jboss.org/jbossaop
[2] http://www.aspectC.net
[3] http://wwwiti.cs.uni-magdeburg.de/iti_db/fcc/
[4] http://www.springframework.net

to improve the modularity, maintainability and evolution of secure applications. In order to achieve this goal, the outline of this tutorial is as follows:

1) Problem statement
   a) A running example
2) The AOP solution
   a) Terminology: joinpoint, pointcut, advice, aspect, weaving
   b) Types of weaving: static weaving versus dynamic weaving
3) Developing secure applications with AOP
   a) Definition and implementation of security aspects
   b) Weaving security aspects with applications
4) Showing evidences of the benefits of AOP
5) A practical example: The INTER-TRUST project

II. Specific goals and objectives

In this section we describe the specific goals and objectives of this tutorial. We follow the outline described in the previous section.

1) **Problem statement.** In this section the motivations to use AOP to improve the development of secure applications by implementing the security concerns as aspects are presented. We will briefly introduce the running example that will be used throughout the tutorial. The main objectives will be to:
   a) Introduce the concepts of concern, tangled concern, scattered concern and crosscutting concern.
   b) Describe an example that makes intensive use of security.
   c) Use the aforementioned example to illustrate the problematic of implementing crosscutting concerns using traditional programming techniques.
2) **The AOP solution.** This part of the tutorial will be an introduction to the AOP paradigm. The goal is to briefly discuss existing AOP approaches, defining at the same time the main concepts of AOP. The main objectives will be to:
   a) Introduce the AOP terminology: joinpoint, pointcut, advice, aspect, weaving.
   b) Define the different types of weaving: static weaving versus dynamic weaving.
   c) Briefly present existing proposals at implementation level, as well as at other development levels.
3) **Developing secure applications with AOP** This section will be of special interest to the attendees since it is the section where they will learn how to use AOP to develop secure applications. That is, they will learn how to "separate" the security concerns from the core application (i.e., the functionality of the application without security) and how to implement them as aspects. They will also learn how the previously separated aspects are then "weaved" again with the core application. The main objectives will be to:
   a) Learn the basics of implementing security using aspect-oriented languages (AspectJ, JBoss AOP, Spring AOP, ...).
   b) Learn the basics of weaving security aspects with applications using compile-time weavers (AspectJ), load-time weavers (AspectJ, JBoss AOP) and runtime weavers (JBoss AOP, Spring AOP).
   c) Learn how to deploy a security policy using AOP.
4) **Showing evidences of the benefits of AOP.** Learning and using a new programming technique is not a straightforward task. Moreover it requires having some evidences (either qualitative or quantiative ones) that demonstrate that the time and effort invested in learning the new approach is worthwhile. The goal of this section is to show existing evidences of the sucessful use of AOP in both research and industrial projects, showing:
   a) How AOP improves the modularisation of secure applications.
   b) How AOP improves the evolution of secure applications.
5) **A practical example of applying AOP to separate security: The INTER-TRUST project.** The INTER-TRUST (Interoperable Trust Assurance Infrasstructure) project is a STREP project that uses AOP to achieve the following goals: (1) separate security aspects from the core applications; (2) negotiate the security policies between different distributed parties at runtime; (3) take advantage of the runtime weaving of security aspects to dynamically deploy and adapt the negotiated security policy at runtime, and (4) take advantage of AOP to monitor and test the behaviour of applications (regarding its impact on security) both during the application's deployment and at runtime. Partners from different top European universities and industries are participating in this project, which is in its first year of development. The goal of this section is to demonstrate that the use of AOP in the development of security applications is already in demand, equally in research as in the industry, and that it has very promising prospects in the near future.

An additional goal of this tutorial is to make it as practical as possible. Thus, we will book some time for the attendees to complete practical and guided exercises. In order to do this the participants must attend the tutorial with the appropriate hardware and software. Since at this point of the proposal, it is not clear whether this requirement can be satisfied or not, we will organize the tutorial in a way so as the impossibility of doing these practical exercises does not negatively affect the quality of the tutorial, or its attractiveness to the attendees.

III. Intended audience

The intended audience of this tutorial are software developers from the securiy research community, who are experts on modeling and/or implementing security issues, and who want to learn how to use AOP/AOSD in the development of secure applications.

IV. Expected background of the audience

The only requirement to be able to attend and to adequately follow the tutorial is to have knowledge of OOP. A basic knowledge of AOP can be useful but it is not required, since the tutorial will introduce all the required AOP concepts.

V. Biographical Sketch of the presenters

a) Mónica Pinto: is an associate professor in the Languages and Computer Science Department at the University of Málaga (Spain). She received her MSc. degree in Computer Science in 1998 from the University of Málaga, and her Ph.D in 2004 from the same University. Her main research areas are component-based software engineering, aspect-oriented software development, architecture description languages, model-driven development, and context-aware mobile middleware. In the last few years she has co-organised the Early Aspects workshop at ICSE, and has been member of the programme committee of several workshops and conferences on AOSD and software composition. She was publicity co-chair at AOSD 2011 and AOSD 2012. She is involved in the AOSD European Network of Excellence and currently participates in several national and international projects on AOSD. She also participates in the INTER-TRUST STREP project that applies AOSD techniques to dynamically deploy and adapt security policies at runtime.

b) José Miguel Horcas: is a Ph.D student in the Languages and Computer Science Department at the University of Málaga (Spain). He received his Computer Engineering degree in 2012 from the University of Málaga, and will receive his MSc later this year. In the last few years, he has participated in the European Higher Education Area (EHEA) in the same department of the University of Málaga in 2010, working in the area of artificial intelligent. He has also been an engineer intern in a startup company in Málaga, focusing on smart document management and enterprise content management in 2011. He is currently working on aspect-oriented software development, quality attributes, software product lines and variability; and is also participating in the INTER-TRUST STREP project.

Acknowledgment

Work supported by the European Project INTER-TRUST 317731 and the Spanish Projects TIN2012-34840 and FamiWare P09-TIC-5231.

References

[1] B. De Win, W. Joosen, and F. Piessens, "AOSD as an enabler for good enough security," *URL: http://www. cs. kuleuven. ac. be/cwis/research/distrinet/resources/publications/41066. pdf*, 2003.

[2] M. Huang, C. Wang, and L. Zhang, "Toward a reusable and generic security aspect library," *AOSD: AOSDSEC*, vol. 4, 2004.

[3] G. Kiczales and E. Hilsdale, "Aspect-oriented programming," *SIGSOFT Softw. Eng. Notes*, vol. 26, no. 5, pp. 313–, Sep. 2001. [Online]. Available: http://doi.acm.org/10.1145/503271.503260

[4] R. Filman, T. Elrad, S. Clarke, and M. Aksit, *Aspect-oriented software development*, 1st ed. Addison-Wesley Professional, 2004.

[5] R. Laddad, *AspectJ in action: practical aspect-oriented programming*. Manning Greenwich, 2003, vol. 6.

[6] ——, *AspectJ in action: enterprise AOP with Spring applications*. Manning Publications Co., 2009.

[7] H. Kim, "AspectC#: An AOSD implementation for C#," Ph.D. dissertation, Trinity College Dublin, 2002.