Aspect-Oriented Programming with C++ and AspectC++

— AOSD 2006 Tutorial Proposal —

Olaf Spinczyk and Daniel Lohmann

October 14, 2005

Abstract

Aspect-oriented programming with C++ does not necessarily require a language extension like AspectJ for Java. There is a set of idioms, which allows the modular implementation of crosscutting concerns in C++, purely based on the features provided by the language. The first part of this tutorial will present these idioms and also discuss the drawbacks of this approach in comparison to a language extension. The rest of the tutorial will concentrate on AspectC++, a language extension to C++ which extends the AspectJ approach into the C++ world. With this extension, aspects can be implemented and applied to component code without having to change the existing code base. The tutorial will introduce AspectC++ language elements and present various examples. A compiler for AspectC++, which transforms AspectC++ code into standard C++, as well as an Eclipse Add-In, are available under the GPL from `www.aspectc.org`. Furthermore, pure-systems GmbH offers commercial support for the compiler and an Add-In to integrate it into Visual Studio. After the language introduction, a dedicated part will focus on development environments for AspectC++. It covers the usage of the `ac++` compiler and also the available integrated development environments, namely the ACDT Eclipse Add-In and the Visual Studio Add-In. These demonstrations are followed by the presentation of a more complex "real-world" application, which shows that AspectC++ is an ideal language to develop (embedded) software product lines. The tutorial ends with a summary and a discussion of future work.

1 Instructors

1.1 Dr. Olaf Spinczyk (primary contact)

Affiliation:
 	
Job title:
Postal address:
Email address:
URL:
Phone number:
FAX number:
	Friedrich-Alexander-University Erlangen-Nuremberg,
	Department of Computer Science 4
	Assistant Professor
	Martensstr. 1, 91058 Erlangen, Germany
	`os@aspectc.org`
	`http://www4.cs.fau.de/~os`
	+49 (9131) 8527906
	+49 (9131) 8528732

Brief biography:	Olaf has a background of more than seven years research on AOP and operating systems. In 2002 he got the "best dissertation of 2002" award by the computer science faculty of the University of Magdeburg, Germany, for his work in this field. In 2001 he started the development of AspectC++. Today he is the main designer and developer of the ac++ weaver. In 2002 he started to cooperate with the pure-systems GmbH in Magdeburg, Germany, to speed up the ac++ development and to evolve it from a research prototype to a commercial product.
Teaching experience:	Olaf taught about 30 classes and seminars at the University of Magdeburg and Erlangen-Nuremberg and gave lectures on aspect-oriented system programming and operating system engineering. In 2002 he was invited by the German Computer Science Society (GI) to give a tutorial on aspect-orientation and operating systems. He already gave this tutorial at AOSD '04 and AOSD'05. At AOSD '05 he was, furthermore, invited to give a talk on AOP with C++ in the industry track.

1.2 Daniel Lohmann

Affiliation:	Friedrich-Alexander-University Erlangen-Nuremberg, Department of Computer Science 4
Job title:	Research Assistant
Postal address:	Martensstr. 1, 91058 Erlangen, Germany
Email address:	`dl@aspectc.org`
URL:	`http://www4.cs.fau.de/~lohmann`
Phone number:	+49 (9131) 8527904
FAX number:	+49 (9131) 8528732
Brief biography:	Daniel worked as software developer, consultant and trainer for several years. He finished his Diploma in Computer Science in 2002. His PhD research is on the development of aspect-oriented operating system product-lines. Since joining the Operating Sytems group at Friedrich-Alexander-University, he actively participates in the AspectC++ language design and the ac++ development. His main focus is the combination of aspects with generic code.
Teaching experience:	Daniel has more than four years of expertise in the training of IT professionals. He taught developers from Deutsche Telekom, Siemens, Deutsche Bank and many other companies. Today, he is teaching classes and seminars on aspect- and object-oriented operating system design at Friedrich-Alexander-University. He already gave this tutorial at AOSD '04 and AOSD' 05.

2 Tutorial

2.1 Classification

“Aspect-Oriented Programming with C++ and AspectC++” (half-day)	
Expected Audience:	Developers and Researchers from industry and academia.
Level of the tutorial:	Introduction to AOP with C++
Prerequisites for participants:	Participants should be familiar with C/C++. Basic understanding of AOSD is recommended.
Previous venues:	This tutorial was held at AOSD ’04 and AOSD ’05. Although seven participants were registered at AOSD ’04, only three finally attended our presentation. Probably, this was mainly caused by the program schedule as we were told by several other AOSD participants. However, our audience was very impressed by the slides and the whole presentation. At AOSD ’05 we had 7 participants (4 paid registrations). We hope that this positive trend will continue at AOSD ’06. Some parts of this tutorial were also presented at the AOSD 2003 demonstration “AspectC++: Bringing Aspects into Deeply Embedded Systems” and at the OOPSLA 2003 demonstration “Variant Management for Embedded Software Product Lines with Pure::Consul and AspectC++”. Moreover, the material was used in university courses and for demonstrations at various IT companies.
Required equipment:	Video projector (1024x768 or higher).

2.2 Synopsis

The aim of this tutorial is to familiarize the attendees with aspect-oriented software development in conjunction with C/C++. While much embraced by the academic community, Java has still limited importance in the “real world”. Many large commercial development projects are still implemented in C/C++ and will be for many more years to come. Being stuck with billions of lines of legacy C/C++ code, developers are seeking for alternatives to refactor old code using the AOSD paradigm as well as to implement new C/C++ code in a more flexible and modular manner. Previous tutorials in this area have focused on emulating AOP with builtin C++ mechanisms and paradigms. The proposed tutorial, in contrast, gives a brief introduction to template-based AOP with C++, but then quickly focuses on the freely available aspect-oriented C++ language extention AspectC++. The tutorial is designed to be very much practice oriented. Theoretical parts will be limited to the extent necessary for the audience to understand the underlying concepts of AOP and AspectC++. Most of the time will be spent guiding the audience step by step through the AOSD cycle using the AspectC++ tool chain. We will pay special attention to the integration of AOP with popular C/C++

developer platforms and IDEs. The ultimate goal of this tutorial is to enable the audience to "get started" with AOSD in general and AspectC++ in particular. For this, we will distribute CDs to all attendees containing the AspectC++ tool chain as well as all example code covered during this tutorial. This will enable attendees to experiment "hands on" with the concepts and tools we introduce during the tutorial. After an overview of the tutorial structure, the remainder of this synopsis will briefly describe each of the six topics to be covered during the proposed tutorial.

Overview and Schedule

Sec.	**Title**	**Style**	**Contents & Educational Goal**	**Duration**
I	Introduction	lecture	overview and motivation ⇒ understand the structure and goals of this tutorial	10 min.
II	AOP with Pure C++	lecture	C++ idioms for AOP, pros and cons ⇒ there are cases in which it makes sense to apply these patterns, but also limitations	30 min.
III	AOP with AspectC++	lecture	AspectC++ concepts and programming, implementation insights ⇒ the AspectC++ language extension approach has several advantages	60 min.
IV	Tool Support	demo	ac++ compiler, IDE support, development cycle ⇒ applying AOP concepts can be easy	50 min.
V	Real-World Example	demo	case-study: aspect-oriented embedded product-line development, weather station product line ⇒ there are practical advantages of using aspects in product lines, ac++ generated code is nearly overhead-free	20 min.
VI	Summary and Future Work	lecture, discussion	tutorial summary, discussion of the development roadmap ⇒ AspectC++ and the ac++ development project need user feedback	10 min.

Section I: Introduction (10 minutes)

After a few words on the presenters, the structure, and the goals of the proposed tutorial, this section will concentrate on the motivation for applying AOP concepts in the C++ world. We will quickly reiterate the benefits of AOSD and show that the problems addressed by it are highly relevant, especially in the development and maintainance of large C++ projects.

Section II: AOP with pure C++ (30 minutes)

The multi-paradigm nature of C++ allows to perform modularization of crosscutting concerns to a certain degree purely with builtin C++ mechanisms. The most commonly used technique for AOP in pure C++ is template meta-programming. As this topic has been covered extensively in previous tutorials, our main focus will be to explore the limitations of this approach, enabling the audience to understand in what use scenarios this technique is adequate and in what situations dedicated AOSD tools should be preferred.

Section III: AOP with AspectC++ (60 minutes)

This section will address the shortcomings of pure C++ AOP approaches which have been identified in the previous section. First, we will look at the well known example of AspectJ, which successfully integrated powerful AOP-mechanisms into Java. We will demonstrate how such a language extention streamlines and simplifies the aspect-oriented software development cycle by being much more powerful and less invasive than “in-language” solutions like template meta-programming. Re-using examples from Section I, we will introduce the audience to the basic concepts and syntax of AspectC++ (aspects, pointcuts, advice, introductions, context variables, join point API, aspect instantiation). After this informal introduction, we will give a more formal and categorized overview of AspectC++ language mechanisms and their design rational. The source-to-source transformation approach of AspectC++ will enable us to give the audience an insight how aspect code and regular C++ code are merged by the aspect weaver. After the introductional part of this section, we will move on to advanced concepts and C++ specific aspects of AOSD. Among other things, we will address how aspects can be aligned with the traditional C++ translation unit model and how in C++ often occurring circular references can be resolved using forward and backward join points. A discussion of selected advanced AspectC++ features including dynamic pointcuts, aspect ordering, aspect control, and aspect templates and the combination of aspects with templates and template meta-programming will round up this section.

Section IV: Tool Support (50 minutes)

Having introduced the audience to the concepts and features of the AspectC++ languages, this section will focus on the actual tool support of AOP in C++. For this, selected examples covered throughout the previous section will be passed through the actual AspectC++ tool chain. We will start with the freely available ac++ compiler, which implements a large subset of the AspectC++ language. The audience will be shown how to invoke the compiler and how to integrate ac++ with popular C++ built tools like make. We will also address the ac++ translation modes, manipulation of libraries as well as link-once code. This section will also cover the integration of AspectC++ into the Eclipse IDE and the the Microsoft Visual Studio IDE. A typical development cycle with editing, compiling, debugging, and testing will be demonstrated.

Section V: Real-World Example (20 minutes)

Integration of novel software development techniques into existing and new projects is a challenging task. We will give an example how AspectC++ is used to enable product line engineering for embedded systems. As part of this section we will report on our experience with using a high-level modeling and implementation technique such as AOSD in the embedded system domain. This is traditionally reserved for assembly level programming, and maybe some C code. We hope to inspire and encourage the audience to consider AOSD for their own projects.

Section VI: Summary and Future Work (10 minutes)

In this section we will summarize the tutorial contents and present our plans for future work. We also would like to get feedback from attendees on the language design and development road-map.

2.3 Final Remarks

Repeatedly pronounced dead or at least dying, C/C++ is still a major player in commercial software development. Developers of complex C++ projects were so far excluded from the benefits of AOSD due to the lack of adequate tool support. "Just switch your implementation language" is often not an option in commercial software development. This tutorial aims to offer practical and immediately applicable advice to C/C++ developers how to exploit the power of AOSD in their software projects. We believe that this tutorial will send a signal to the industry that the AOP community is serious about the integration of AOSD in all relevant programming languages, not just the ones that are currently considered "state of the art" from an academic perspective.

The tutorial material is of great interest for practitioners. It has already been downloaded about 1500 times since may 2005. We hope that the growing maturity of the AspectC++ tool chain and the high-quality presentation will also attract an acceptable number of AOSD participants this year.