Improving Design and Source Code Modularity
Using AspectJ™ *

Cristina Videira Lopes
Computer Science Laboratory
Xerox Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304, USA
lopes@parc.xerox.com

Gregor Kiczales
Department of Computer Science
University of British Columbia
201-2366 Main Mall
Vancouver, BC V6T 1Z4, Canada
gregor@cs.ubc.ca

ABSTRACT
Using only traditional techniques the implementation of concerns like exception handling, multi-object protocols, synchronization constraints, and security policies tends to be spread out in the code. The lack of modularity for these concerns makes them more difficult to develop and maintain. This tutorial shows how to use Aspect-oriented programming (AOP) [2, 3] to implement concerns like these in a concise modular way. We discuss the effects aspects have on software design and on code modularity. The concrete examples in the tutorial use AspectJ [1], a freely available aspect-oriented extension to the Java™ programming language.

OUTLINE
This half-day tutorial is divided into three main parts. What follows is a brief overview of those three parts.

Introduction to Aspect-Oriented Programming. We describe the motivation for AOP with one concrete example, we introduce the terminology and we present the big picture of the AOP approach. A complex system contains features that don’t align well with the primary units of modularity. This is a natural phenomenon, but, using only traditional implementation techniques, it causes some problems in the source code: the program units (e.g. Java classes) end up containing pieces of implementation for many different purposes, and, in many cases, those pieces of implementation are tangled, line by line, in ad-hoc ways. The goal of AOP is to provide better support for separation of concerns that crosscut the primary units of modularity.

Basic Mechanisms of AspectJ. We present the basic mechanisms of the AspectJ language, using a single simple example, and in a way that smoothly extends what the audience knows about Java. We start by presenting the notion of “points in the execution” (or “events”) that are captured by a specific AspectJ construct. Examples of those points are the execution of a specific method, the calls to a specific method, the instantiation of classes and the handling of an exception of a certain type. These sets of points in the execution can be composed to form more complex patterns; for example, all the calls that follow the execution of a specific method, or the calls to objects of type SecurityManager. Having this concept explicitly in the language, allows us to define pieces of implementation that crosscut many classes and methods. We present how that can be done with two mechanisms: advice and introduction. We show how aspect code can be reused by inheritance.

Using Aspects. This part of the tutorial presents how the kinds of crosscutting mechanisms AspectJ can capture can be used to design and implement pieces of real systems. The examples include: tracing and debugging, instance counters, layers of functionality for a simple telecom application, consistency checkers, RMI aspects, GUI skins and the subject/observer design pattern.

Besides these three main parts, we will show a brief demonstration of an integrated development environment for AspectJ. We will also present an overview of related systems. The tutorial concludes with an open-ended conversation with the audience.

REFERENCES
1.  http://aspectj.org

* This work was supported in part by the Defense Advanced Research Projects Agency under contract number F30602-97-C-0246.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to publish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

ICSE 2000, Limerick, Ireland
© ACM 23000 1-58113-206-9/00/06 ...$5.00