OpenAOP: Analysing Cross-Cutting Concerns in Open Source Software

Tony Kong
University of British Columbia
Vancouver, BC
y2k0b@ugrad.cs.ubc.ca

Raymond Situ
University of British Columbia
Vancouver, BC
r7p0b@ugrad.cs.ubc.ca

Kevin Wong
University of British Columbia
Vancouver, BC
b2j0b@ugrad.cs.ubc.ca

Benjamin Hwang
University of British Columbia
Vancouver, BC
r6o0b@ugrad.cs.ubc.ca

James Yoo
University of British Columbia
Vancouver, BC
l4k0b@ugrad.cs.ubc.ca

ABSTRACT

Aspect-oriented Programming is a technique for improving the modularity of a system via the *Aspect*, which encapsulates the cross-cutting concerns and enforces a separation of concerns between business and application logic. In *Open AOP*, we analyze an open-source Java software system, measure the extent of cross-cutting as well as the overall cohesion of the system prior to refactoring using AspectJ. We then analyze the evolved system, with respect to cross-cutting and cohesion.

KEYWORDS

Aspect-oriented Programming, Cross-cutting concern, Joinpoint, Pointcut, Advice, Tangling, Cohesion, AspectJ

ACM Reference Format:
Tony Kong, Raymond Situ, Kevin Wong, Benjamin Hwang, and James Yoo. 2018. OpenAOP: Analysing Cross-Cutting Concerns in Open Source Software. In *Proceedings of Intro. PL (CPSC 311).* ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 MOTIVATION

Modern software systems are extremely complex and often require large amounts of logic which may be unrelated to its core concerns. Classical examples of this include logging and exception handling. When these cross-cutting concerns are scattered throughout modules, we call it *tangled.* A consequence of tangling is that the cohesion of tangled modules. As a result, the system becomes more difficult to reason about and evolve.

2 METHODOLOGY

(1) Select a medium-sized [1] open-source project and extract the following metrics.
  - Degree of crosscutting [2]
  - Cohesion
(2) With these metrics in place, we will refactor the project using AspectJ
(3) We revisit the metrics we extracted in (1)

[1] $\leq$ 6000 SLOC
[2] 1 Murphy et al: Identifying, Assigning, and Quantifying Crosscutting Concerns

CPSC 311, 2018W1, UBC Vancouver
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 HIGH LEVEL OBJECTIVES

Our primary goals for this project are as follows

(1) Refactor an open-source software system using AspectJ
(2) Compare and contrast the degree of crosscutting and cohesion present in the prior system to that of the evolved system

Successful completion of the objectives above would map to the 100% project completion milestone.

4 PROJECT MILESTONE OUTLINE

Although we have committed to the 100% completion goal, below we detail a high-level overview of what our project would look like at different levels of completion

4.1 80% Completion

For this milestone, we expect each member to be comfortable with the following

(1) Identify a cross-cutting concern in the context of AOP
(2) Write an AspectJ aspect to address the concern above

This would be achieved by consulting the literature in our weekly meetings and each member "owning" a single facet of AspectJ and teaching others. Using the competencies above, we would perform a static analysis of a very small Java software system and generate a report on the cross-cutting concerns existent in it, and *how* AspectJ could be utilized to mitigate them.

4.2 90% Completion

This milestone assumes that all group members are comfortable with the competencies outlined in **4.1**, but takes it further by

(1) Writing a small-scale software system (e.g. small game, some tool) in Java
(2) Evolving said system by introducing aspects via AspectJ
(3) Comparing the evolved system with the prior system

4.3 100% Completion

This milestone is similar to the 90% milestone, but applied to an open-source software system in a larger scale. We expect this to be significantly more challenging than the 90% target due to the fact that not only do we have to learn AspectJ, but we also have to become familiar enough with a foreign system to refactor and evolve the system appropriately.

4.4 Poster

This is the part of the project which other CPSC311 classmates will have the most exposure to. Therefore, we plan to take a high-level approach to describing AspectJ. We will detail the primary language features of AspectJ, such as the *Aspect*, *Pointcut*, *Joinpoint*, and the different types of *Advice* made possible. In addition to the features above, we will also enumerate some common cross-cutting concerns which are targeted by AspectJ, and provide a simple demo of AspectJ in action using an IDE.

5 SELECT REFERENCES

This section details a few sources which we will utilize as a "jumping-point" for our project. Note that these are not the complete set of sources which we will consult, but are a subset which are especially appropriate for getting familiar with AspectJ and AOP.

(1) Eclipse Foundation: *Getting Started with AspectJ*

This is a webpage which contains some key information about the AspectJ language, containing information about Pointcuts, Joinpoints, and the different types of Advice which are available as part of the language. It is especially appropriate as a primary language specification due to the fact that the Eclipse foundation is strongly associated with AspectJ.

(1) Kiselev, Ivan: *Aspect-Oriented Programming with AspectJ*

This is a book which is accessible to people with a general knowledge of the Java programming language. It eases readers into AspectJ with concrete code examples which are small enough to be written into an IDE at the same time.

(1) Murphy et al: *Identifying, Assigning, and Quantifying Cross-cutting Concerns*

This paper will deal with researching how we can measure the amount of cross-cutting which is present in a system. This is important as we will require a metric in order to measure the efficacy of AspectJ in mitigating cross-cutting.

6 CITATIONS

(1) Murphy et al: *Identifying, Assigning, and Quantifying Cross-cutting Concerns*
(2) Baeldung, Intro to AspectJ: *https://www.baeldung.com/aspectj*
(3) Eclipse Foundation: *Getting Started with AspectJ*
(4) Kiselev, Ivan: *Aspect-Oriented Programming with AspectJ*
(5) o7planning, Java Aspect Oriented Programming Tutorial: *goo.gl/BDVAEk*