*Appears in *Object-Oriented Technology: ECOOP '98 Workshop Reader*, LNCS 1543

Assessing Aspect-Oriented Programming: Preliminary Results

Robert J. Walker, Elisa L. A. Baniassad, and Gail C. Murphy

Department of Computer Science
University of British Columbia
201-2366 Main Mall, Vancouver B.C., Canada V6T 1Z4
{walker, bani, murphy}@cs.ubc.ca

Abstract. The aspect-oriented programming approach claims to make it easier to reason about, develop, and maintain certain kinds of application code while maintaining highly efficient code. To better understand the usefulness and usability of the aspect-oriented approach, we have been conducting a series of experiments. These experiments are designed to investigate such characteristics of aspect-oriented development as the creation and ease of debugging programs built in this style. This paper provides an overview of the experiments we have conducted to date.

1 Introduction

Aspect-oriented programming[1] is in its infancy. The approach claims to make it easier to reason about, develop and maintain certain kinds of application code while maintaining highly efficient code. To better understand the usefulness and usability of the aspect-oriented approach, we are currently conducting three "experiments"[1]: one to investigate the ease of creating aspect-oriented programs, another to investigate the ease of debugging, and a third to investigate the ease of change. The experiments investigate aspect-oriented design and programming as represented in AspectJ[2], an aspect-oriented variant of Java[3] developed at Xerox PARC.

We present here a brief overview of the first two experiments. For further details, the reader is referred to a technical report [2].

2 Pilot Study

To understand how difficult a problem we could realistically ask a participant to tackle in a period of no more than four hours, we set-up the first experiment as a pilot study. The experimental question was whether, in the context of AspectJ, the combination of JCore for component programming and COOL as a

[1] To overcome constraints, such as a small participant pool, we set our experiments up as semi-controlled empirical studies rather than statistically valid experiments.
[2] AspectJ is a trademark of Xerox Corporation.
[3] Java is a registered trademark of Sun Microsystems.

synchronization aspect language eases the creation of multi-threaded programs compared to programming in the Java object-oriented language.

We had three programmers attempt a solution to a small programming problem with concurrency in Java; another three programmers attempted a solution in AspectJ. We video-taped and later analyzed these programming sessions.

None of the six participants in the experiment were able to produce a solution to the programming problem in the time provided, although two came close (one Java, one AspectJ). The major impediment to the programmers seemed to be that the task was more difficult than we expected for the time available. We used our experiences with the pilot study to refine our subsequent experiments.

3 Experiment 1: Ease of Debugging

The intent of the second experiment we conducted was to learn whether the combination of JCore for component programming and COOL as a synchronization aspect language eased the debugging of multi-threaded programs, compared to the ability to debug the same program written in Java.

In this experiment, we had pairs of programmers attempt to fix three bugs related to synchronization that we seeded into an approximately 600-line program. Three of the pairs worked with AspectJ, three with Java. All of the pairs were able to find and correct all three of the bugs.

We compared the performance of the pairs by comparing such factors as the time it took them to fix each of the bugs and the number of times they examined the semantics of the core program. Our analysis found that the AspectJ pairs were able to complete debugging tasks with fewer instances of semantic analysis which seemed to lead directly to less switching between files, indirectly to fewer builds, and ultimately to quicker completion times.

4 Acknowledgments

Thanks to the Xerox PARC Embedded Computation Area group, for their the anonymous participants who took part in the sessions, and Robert Rekrutiak and Paul Nalos for their work on experiment setup. Funding was provided by Xerox Corporation, a UBC Graduate Fellowship, and NSERC.

References

1. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *ECOOP'97—Object-Oriented Programming, 11th European Conference*, LNCS 1241, pages 220–242, 1997.
2. R.J. Walker, E.L.A. Baniassad, and G.C. Murphy. An initial assessment of aspect-oriented programming. Technical Report UBC-TR-98-12, Department of Computer Science, University of British Columbia, 1998.