Research Statement

Marc Eaddy
155 E 49th Street #6B
New York, NY 10017

+1 (212) 593-3583
me133@columbia.edu
www.columbia.edu/~me133

I plan to make software easier to develop and maintain by enabling architects, developers, and maintainers to better understand and modularize programs. The inability to effectively modularize the requirements, features, etc. (*concerns*) of a program is called the *crosscutting concern problem*—so named because the implementation of the concern is scattered across (crosscuts) the program structure. When concerns are poorly modularized, programs become more complex, resulting in increased maintenance effort and defects. I plan to better understand—and try to solve—the crosscutting concern problem, thereby making software development less costly and more reliable.

To better understand the nature of crosscutting, I created a *concern model* that models the relationship between the concerns and the program and a suite of *concern metrics*. To quantify the impact of crosscutting, my colleagues and I conducted several empirical studies and found that *as crosscutting increases so do defects* [EZS+07]. This provides a strong motivation to determine why crosscutting occurs and find ways to reduce it.

Crosscutting can be reduced with better languages and development tools. I am designing a language called *Wicca#*—an extension of C# that unifies constructs from object-oriented programming, aspect-oriented programming [KIL+96], and open classes—to better modularize crosscutting concerns. I have also developed compiler technologies and tools for modularizing crosscutting concerns.

Developers should proactively avoid implementing concerns in a crosscutting fashion. However, when concerns cannot be modularized, we need better ways to manage them. I created the *ConcernTagger* and *DependencyViewer* tools to provide ways for developers to specify and visualize the relationship between concerns and code, pinpoint poorly modularized (crosscutting) concerns, and automatically refactor concern code to improve modularity. Ultimately, *I plan to make development tools more concern aware*, so that developers are always aware of the concerns, documents, emails, change history, bugs, etc. associated with a piece of code. This will improve program understanding, which will in turn reduce the likelihood of defects—and improve productivity since a developer will no longer need to waste their or another developer's time to obtain this information [KDV07].

Research

A Theory of Concerns

An overarching goal of my research is to *formulate a theory of concerns* that explains what concerns are, how they relate to code and other artifacts, why some concerns are crosscutting, and how this impacts software quality. This is needed before we can properly understand the crosscutting concern problem and propose effective solutions. I created a *model of concerns* that formalizes the concepts of "concern" and "crosscutting," and the relationships between concerns and program elements. I created *concern metrics* to operationalize these concepts and precisely quantify terms such as "highly crosscutting" [EAM07].

Gail Murphy from the University of British Columbia, my advisor, Alfred Aho, and I performed a study to test the feasibility of my model and metrics, and to obtain a complete picture of the concerns of several programs. To associate concerns with code, we created the *pruning dependency rule*, which overcomes ambiguities inherent in prior techniques [EAM07]. We successfully applied the rule on five case studies. We used *statement-level concern annotations* [EA06] to tag each statement in a 13,000 line C# program with the names

of the implemented concerns. We found that 95% of the software requirements were crosscutting to some extent, and that our concern metrics were better at describing the amount of crosscutting than other metrics.

Are Crosscutting Concerns Harmful?

Very little is known about the impact of crosscutting concerns on code quality. Some studies have shown that reducing crosscutting leads to improved *internal* quality indicators such as reduced coupling, but, until now, there has been no evidence that this leads to improved *external* quality indicators such as reducing maintenance effort and defects—the "holy grail" of software engineering. Murphy, Nachiappan Nagappan from Microsoft Research, Tom Zimmerman from the University of Calgary, Aho and I performed three case studies to gather empirical evidence of a statistical correlation between crosscutting concerns and defects. To perform our studies, we created the most rigorous methodology to date for assessing the impact of crosscutting concerns. We found that the more scattered the implementation of a concern is, the more defects it has. Aho has dubbed this phenomenon "Eaddy's Law." Our results provide convincing evidence that crosscutting concerns are harmful and motivate finding ways to reduce crosscutting. Our paper [EZS+07] was recently accepted for publication by the IEEE Transactions on Software Engineering Journal, with much acclaim from the reviewers:

> *"of paramount importance to the software engineering literature,"*
> *"a very interesting empirical paper addressing an important question,"*
> *"the authors have done an excellent job defining appropriate metrics and carrying out a careful case study,"*
> *"a very well-written paper that is extremely interesting and makes a valuable contribution to the field,"*
> *"an exemplary contribution to empirical software engineering."*

Concern Location

Concern analysis requires locating the parts of the program that implement each concern. The problem of associating concerns with code (i.e., the concept assignment problem) is hard, and I have researched it extensively. I developed *ConcernTagger*, a plug-in for Eclipse, that allows an analyst to manually associate concerns with code (methods, classes, etc.) [Ead07a]. Ramesh Viswanathan from Bell Labs and I are extending ConcernTagger to *semi-automate* concern location using program dependency analysis and theorem proving. The idea is that if a program element (e.g., method) has a pruning dependency on a concern, then *uses* of that element (e.g., calls to the method) also have that dependency. I am also working with Giuliano Antoniol and Yann-Gaël Guéhéneuc from Polytechnique Montréal to use information retrieval and static and dynamic analyses techniques to *fully automate* concern location [EAA+08], which is essential for large rapidly evolving systems.

I created *DependencyViewer* during an internship at Microsoft Research, where I was hosted by Manuel Fähndrich. Dependency Viewer visualizes program dependencies to aid the comprehension of large code bases. It also refactors the code base to (1) eliminate dependency cycles to simplify removing functionality that is not supported by a particular target platform (i.e., *software pruning*), and (2) better modularize concerns using open classes.

Programming Language Design for Modularizing Crosscutting Concerns

In addition to understanding the crosscutting concern problem, I have proposed solutions. I created *Phx.Morph*, a static byte code weaver that supports aspect-oriented programming and open classes for .NET, during an internship with the Phoenix research group at Microsoft [Ead06a]. Phx.Morph is unique in that it is built on top of Phoenix, Microsoft's backend compiler infrastructure, and supports *statement-level annotations* to enable fine-grained advising [EA06]. I also developed a framework called *Wicca*, which uses the .NET debugging APIs to support dynamic aspect-oriented programming, i.e., the ability to update the code of a running program [Ead06b]. I showed how Wicca can reduce crosscutting [Ead07b] and aid debugging [EAH+07] by injecting code at runtime.

Many of the proposed approaches to reducing crosscutting make programs even more confusing [Ste06]. I

am designing a *language called Wicca#* that seeks to unify concepts from aspect-oriented programming, open classes, and object-oriented programming. The goal of Wicca# is to support modularizing crosscutting concerns to make the program more modular and easier to understand. After I implement Wicca#, I plan to use empirical assessment to show tangible benefits.

Research Plan

The crosscutting concern problem is widespread and significant. The problem must be solved to allow us to manage the ever-growing complexity of software and improve reliability. I plan to improve our understanding of the crosscutting concern problem, and develop and assess techniques, technologies, and tools for minimizing its effects. I intend to answer the following questions:

> *"Can we improve modularity with new languages and tools?"*
> *"Are crosscutting concerns caused by inadequate technology, developer aptitude, or inherent complexity?"*
> *"Can reducing and/or managing crosscutting concerns improve program understandability, reliability, and maintainability?"*
> *"How can we automatically determine the relationship between code and concerns, emails, bugs, and other artifacts, and how can we effectively present this information?"*

In conclusion, my research, if successful, will improve software reliability—by reducing or eliminating defects related to crosscutting concerns—and make software easier to develop, understand, and maintain.

References

[Ead06a] M. Eaddy, "Phx.Morph - Weaving using the Microsoft Phoenix compiler back-end," Demonstration, in *Conference on Aspect-Oriented Software Development (AOSD 2006)*, Bonn, Germany, March 20-25 2006.

[Ead06b] M. Eaddy, "Wicca Project Website," 2006, http://www.cs.columbia.edu/~eaddy/wicca.

[Ead07a] M. Eaddy, "ConcernTagger Project Website and Dataset Repository," 2007, http://www.cs.columbia.edu/~eaddy/concerntagger.

[Ead07b] M. Eaddy, "Wicca 2.0 - Dynamic Weaving using the .NET 2.0 Debugging APIs," Demonstration, in *Aspect-Oriented Software Development (AOSD 2007)*, Vancouver, British Columbia, Canada, March 12–16 2007.

[EA06] M. Eaddy and A. Aho, "Statement Annotations for Fine-Grained Advising," in *Workshop on Reflection, AOP, Meta-Data for Software Evolution (RAM-SE 2006)*, Nantes, France, 2006.

[EAH+07] M. Eaddy, A. Aho, W. Hu, P. McDonald, and J. Burger, "Debugging Aspect-Enabled Programs," in *International Symposium on Software Composition (SC 2007)*, Braga, Portugal, March 25-26 2007.

[EAM07] M. Eaddy, A. Aho, and G. C. Murphy, "Identifying, Assigning, and Quantifying Crosscutting Concerns," in *Workshop on Assessment of Contemporary Modularization Techniques (ACoM 2007)*, Minneapolis, Minnesota, USA, May 22 2007.

[EAA+08] M. Eaddy, A. V. Aho, G. Antoniol, and Y.-G. Guéhéneuc, "Cerberus: Tracing Requirements to Source Code Using Static, Dynamic, and Semantic Analysis," in *International Conference on Program Comprehension (submitted) (ICPC 2008)*, Amsterdam, The Netherlands, June 10–13 2008.

[EZS+07] M. Eaddy, T. Zimmerman, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho, "Do Crosscutting Concerns Cause Defects?," *IEEE Transactions on Software Engineering (to appear),* 2007.

[KIL+96] G. Kiczales, J. Irwin, J. Lamping, J.-M. Loingtier, C. V. Lopes, C. Maeda, and A. Mendhekar, "Aspect-oriented programming," *ACM Computing Surveys,* 28(4es):154, 1996.

[KDV07] A. J. Ko, R. DeLine, and G. Venolia, "Information Needs in Collocated Software Development Teams," in *International Conference on Software Engineering (ICSE 2007)*, Minneapolis, Minnesota, USA, May 20-26 2007.

[Ste06] F. Steimann, "The paradoxical success of aspect-oriented programming," *SIGPLAN Not.,* 41(10):481-497, 2006.