

# Capturing Concerns with Conceptual Modules

Albert Lai and Gail C. Murphy  
Department of Computer Science  
University of British Columbia  
2366 Main Mall  
Vancouver BC Canada V6T 1Z4  
{alai,murphy}@cs.ubc.ca

## Abstract

*Separating a concern in an existing system with currently available modularization mechanisms requires an understanding of how the concern interacts with the rest of the system. Specifically, a developer must be able to trace and understand dependencies between the concern and other code. Tracing such dependencies is often a tedious and time-consuming task. To help a developer with this task, we are developing a conceptual modules tool that allows the developer to logically describe the code related to a concern and to analyze the interaction of that concern with the system prior to the actual separation of the concern. This tool uses data-flow information extracted from compiled Java classes.*

## 1. Introduction

Various approaches have been used to support the identification of concerns in software systems, including minimal subsets [3, 4], pattern matching [5], and user-level features [6].

Once identified, it is helpful to modularize a concern of interest to help maintainers identify and change the concern amongst other reasons. Newer modularization mechanisms, such as Hyper/J<sup>TM</sup> and AspectJ<sup>TM</sup> are intended to help in the modularization process. However, applying these mechanisms requires an understanding of how the concern of interest interacts with other concerns. For example, in exploratory work we conducted in separating a graphic user interface (GUI) concern [6], we found that in one class, the GUI concern was present along with three other concerns. To properly separate the GUI concern from the other ones, the developer needed to examine the dependencies between it and the other concerns. Tracing dependencies proved to be not only a tedious task, but also a task that distracted from

the more important goal of deciding on an appropriate modularization for the GUI concern.

We believe that the task of modularizing a concern can be simplified if a developer can express the concern logically and then reason about it in the context of the existing structure. Specifically, we are investigating whether the conceptual module [2] approach can be adapted to provide the desired support. The conceptual module approach allows a logical structure to be overlaid on top of the existing structure of a system. Using information about the data-flow of the code identified in a logical module, the impact of making a potential modularization change can be elucidated because dependencies for a given concern can be efficiently identified and addressed.

## 2. Overview of Conceptual Modules

Previous work in conceptual modules was targeted towards the reengineering of C code bases. This previous work defined a conceptual module (CM) as a set of developer-specified lines of (C) code that were to be treated as a logical unit. Each CM represented a logical procedure. A developer could then use either data-flow or cross-reference information [8] to determine the interfaces of CMs. It also allowed developers to determine the interactions between one CM and another, or one CM and the rest of the source code. Queries provided by the tool determined whether or not two conceptual modules shared common code or were mutually exclusive.

## 3. Applying Conceptual Modules to Concern Modularization

Once a concern has been lexically (or otherwise) identified in the source code, a developer can create a CM from the corresponding lines of source code. While

modularizing the concern, the developer can query the CM to determine all the dependencies with the remaining code that need to be addressed. By tracking these dependencies, the developer's task is eased. Moreover, the developer can elucidate how the concern interacts with other concerns captured as CMs. This approach is especially appropriate for concerns that have more complex interactions with other concerns.

### 3.1. Work in Progress

We are currently adapting the conceptual modules approach to work with object-oriented structures – more specifically, the structures supported by Java [1]. Our initial target is a tool that aids in the reengineering of Java code. This tool will allow a developer to create a CM from a set of lines of code related to a concern, and will support queries related to the use and definition of fields, methods and classes.

The main benefits of the tool are related to how it supports a developer in reasoning about a source code base. Developers can reason about how a concern interacts with the rest of the code without actually performing any changes to the code. For any potential change, developers can reason about the existing structure and the desired structure at the same time.

The tool can also help identify concerns. Once a CM is created, the tool can determine the dependencies between the CM and the rest of the code. This may include dependencies such as variables that are required as input to methods within the CM, methods that expect return values from methods within the CM, or fields that are defined in the CM but referenced outside the CM. A developer can consider whether these dependencies suggest that there is more to the concern than has been captured by the CM. For example, a CM may contain many uses of a field. This might indicate that the field belongs to the concern in question.

Eventually, we will enhance the tool for use with newer modularization mechanisms such as Hyper-spaces [7, 9]. The tool will suggest modifications to a CM to make it easier to apply some of the modularization mechanisms found in Hyper/J. It will also export CM's to various Hyper/J modularization units such as *concern mappings*.

Although the tool will initially support only Hyper/J, it should be possible to adapt it to work with other advanced separation of concern mechanisms. This will make it possible for a developer to encapsulate a single concern in a CM and attempt to modularize it using any of the supported mechanisms. A developer can then note the modifications required to support each of the mechanisms. This makes the tool ideal for use in comparing mechanisms.

There is potential for the tool to be adapted as a

framework for concern metrics. For example, developers could use the tool to determine the definitions or uses shared by two concerns. This would indicate the degree of entanglement between two concerns and suggest to a developer the amount of work required to separate the concerns.

### Acknowledgements

We would like to thank Martin Robillard for comments on a draft of this paper. This work was supported in part by an IBM University Partnership Program award and a UBC University Graduate Fellowship.

### References

- [1] K. Arnold and J. Gosling. *The Java™ Programming Language*. Addison-Wesley, 1996.
- [2] E. Baniassad and G. Murphy. Conceptual module querying for software reengineering. In *Proc. of ICSE*, pages 64–73, 1998.
- [3] L. Carver and W. G. Griswold. Sorting out concerns. Position Paper for the 1999 OOPSLA Workshop on Multi-dimensional Separation of Concerns in Object-Oriented Systems, October 1999.
- [4] E. Dijkstra. Design and specification of the minimal subset of an operating system family. *Communications of the ACM*, SE-2(4):301–307, 1976.
- [5] W. G. Griswold, Y. Kato, and J. J. Yuan. Aspect browser: Tool support for managing dispersed aspects. Position Paper for the 1999 OOPSLA Workshop on Multi-dimensional Separation of Concerns in Object-Oriented Systems, October 1999.
- [6] A. Lai and G. C. Murphy. The structure of features in Java code: An exploratory investigation. Position Paper for the 1999 OOPSLA Workshop on Multi-dimensional Separation of Concerns in Object-Oriented Systems, October 1999.
- [7] H. Ossher and P. Tarr. Hyper/J: Multi-dimensional separation of concerns for Java. In *Proc. of ICSE*, pages 734–737, 2000.
- [8] S. Reiss. Connecting tools using message passing in the Field program development environment. *IEEE Software*, 7(4):57–66, 1990.
- [9] P. Tarr, H. Ossher, W. Harrison, and S. Sutton. N degrees of separation: Multi-dimensional separation of concerns. In *Proc. of ICSE*, pages 107–119, 1999.