ConcernMorph: Metrics-based Detection of Crosscutting Patterns

Eduardo Figueiredo
Computing Department
Lancaster University, UK
e.figueiredo@lancs.ac.uk

Jon Whittle
Computing Department
Lancaster University, UK
whittle@comp.lancs.ac.uk

Alessandro Garcia
Informatics Department
PUC-Rio, Brazil
afgarcia@inf.puc-rio.br

ABSTRACT
Crosscutting concerns can hinder maintainability of a design because they do not adhere to a system's underlying modular structure. Developers, therefore, may wish to refactor designs to improve modularisation or to implement crosscutting concerns as aspects. However, few tools currently exist that assist developers in detecting and classifying crosscutting concerns in their code. Classification is important because, as recent studies have shown, crosscutting concerns are not always harmful. This paper describes a tool, ConcernMorph, for identifying crosscutting concerns and classifying them into one of a number of predefined *crosscutting patterns*.

Categories and Subject Descriptors
D.2.3 [**Software Engineering**] Coding Tools and Techniques.

General Terms: Measurement, Design, Experimentation.

Keywords: Crosscutting concerns, metrics, crosscutting patterns.

1. INTRODUCTION
A concern is anything of interest to one or more stakeholders that affects the design and implementation of program modules [8] – features, non-functional requirements and design decisions are some examples of concerns. A concern is crosscutting if it cannot be well modularised in a particular design or implementation [6]. Recent studies have confirmed that crosscutting concerns can lead to design flaws which hinder software maintainability [3, 4, 5]. However, there is also some evidence that not all types of crosscutting concerns are harmful [3, 4]. Therefore, when considering refactoring of a design to factor out crosscutting concerns, the effort of doing so must be balanced against the level of harm caused by the crosscutting nature of the concern.

There already exist a number of tools for concern representation [8] and concern measurement [5]. However, developers should also be given information not only on whether a concern is crosscutting or not, but also on the nature of the crosscutting. This is important, for instance, to avoid expending effort in refactoring a crosscutting concern which is, in fact, not harmful.

In previous work [4], we identified thirteen *crosscutting patterns* capturing characteristics of crosscutting concerns that have been found to occur widely in a number of different systems. These patterns capture the fact that crosscutting concerns may manifest themselves in significantly different ways, ranging from a few minor elements of the concern scattered across a single object hierarchy, to many elements littered across the entire system. We have shown that there are correlations between certain types of crosscutting patterns and design flaws [4]. In particular, some crosscutting patterns are more harmful than others.

Of course, the identification of crosscutting patterns is a tedious task without proper tool support. This paper, therefore, describes ConcernMorph[1], a tool to support automatic identification of crosscutting patterns in Java source code. ConcernMorph is implemented as an Eclipse plug-in and relies on ConcernMapper [1]. ConcernMorph identifies crosscutting patterns using metrics that are defined at the level of design concerns rather than the level of code modules (as is the case with traditional metrics). Each crosscutting pattern is defined as a *detection strategy* [7], that is, a rule combining a number of related metrics with thresholds determining whether a case of crosscutting should be classified as an instance of the pattern.

2. CROSSCUTTING PATTERNS
The current version of ConcernMorph supports the detection of twelve crosscutting patterns (in five categories) [4]: the *basic crosscutting pattern*; 3 *flat crosscutting patterns* (Black Sheep, Octopus, and God Concern); 2 *structural concerns* (Data Concern and Behavioural Concern); 2 *inheritance-wise concerns* (Climb-ing Plant and Hereditary Disease); and 4 *communicative concerns* (King Snake, Tree Root, Tsunami, and Neural Network).

Condition A - *Little Dedication*: (CA / NOA < 0.33) **and** (CO / NOO < 0.33)

Condition B - *High Dedication*: (CA / NOA ≥ 0.67) **and** (CO / NOO ≥ 0.67))

DS1 - *Black Sheep*:
if (*Little Dedication*) **for every class associated with** CONCERN
then CONCERN is BLACK SHEEP

DS2 - *Octopus*:
if (*Little Dedication*) **for at least 1 class and** (*High Dedication*) **for at least 1 class associated with** CONCERN **then** CONCERN is OCTOPUS

Figure 1. Detection strategies for flat crosscutting patterns

Each crosscutting pattern is defined as a rule, or detection strategy [7], that combines a number of related metrics. Figure 1 gives examples of rules for two flat crosscutting patterns: Black Sheep and Octopus. Black Sheep is a concern that crosscuts the system but affects very few elements in distinct places. Octopus, on the other hand, is a crosscutting concern which is partially well modularised by one or more classes, but is also spread across a number of other classes. Detection strategies for Black Sheep (DS1) and Octopus

Copyright is held by the author/owner(s).
ESEC/FSE'09, August 24–28, 2009, Amsterdam, The Netherlands.
ACM 978-1-60558-001-2/09/08.

[1] Available at http://www.lancs.ac.uk/postgrad/figueire/concern/plugin/

(DS2) in Figure 1 use auxiliary definitions – *Little Dedication* and *High Dedication* – which are themselves defined in terms of metrics (CA, NOA, CO, and NOO).

It is important to note that some of these metrics are *concern metrics*, that is, they measure properties of concerns which may be implemented by multiple modules. Traditional object-oriented metrics, by contrast, measure properties of modules. ConcernMorph implements five concern metrics [3, 5] and three conventional size metrics. Supported concern metrics includes Concern Attributes (CA) and Concern Operations (CO) - used in Figure 1 - which count, respectively, the number of attributes and methods assigned to a concern. Moreover, the conventional metrics NOA and NOO count, respectively, the number of attributes and operation of a system.

3. CONCERNMORPH

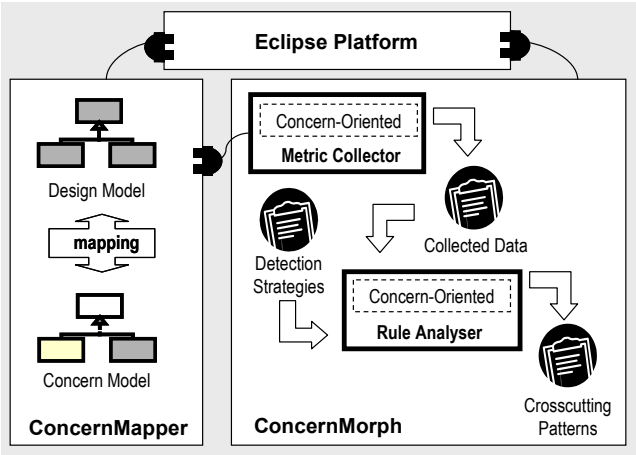

Figure 2. The ConcernMorph Architecture

Concern metrics require users to specify a mapping between concerns and syntactic elements, such as classes, methods, and attributes. An existing tool, ConcernMapper [1], offers good support for this and is used by ConcernMorph. Figure 2 shows the architecture of ConcernMorph and its relationship to ConcernMapper and the Eclipse Platform. Both tools, ConcernMorph and ConcernMapper, are coupled to the Eclipse Platform. ConcernMorph has two main modules (Figure 2): (i) *Metric Collector* that is responsible for computing the concern metrics and (ii) *Rule Analyser* which applies detection strategies to identify crosscutting patterns.

Figure 3 presents two views of ConcernMorph: Concern Metrics (View A) and Crosscutting Patterns (View B). The former shows concern measurements and the latter shows all instances of crosscutting patterns found in an illustrative system, called MobileMedia. The Crosscutting Patterns view also shows the number of classes (between parentheses) taking part in each pattern instance. For example, the Album concern is classified as Octopus and 8 classes are forming this Octopus instance.

4. CONCLUDING REMARKS

This paper presented ConcernMorph, a tool for detecting recurring patterns of crosscutting concerns based on concern metrics and heuristic analysis. The usability of ConcernMorph has been validated by analysing code bases of iBATIS, MobileMedia, and Health Watcher [4]. Ongoing work is empirically exploring the relationship between crosscutting patterns and design flaws.

5. REFERENCES

[1] ConcernMapper: http://www.cs.mcgill.ca/~martin/cm/

[2] Chidamber, S. and Kemerer, C. "A Metrics Suite for Object Oriented Design". IEEE Trans. on Software Eng., 1994.

[3] Eaddy, M. *et al*. "Do Crosscutting Concerns Cause Defects?" IEEE Trans. on Software Engineering, 34(4), 497-515, 2008.

[4] Figueiredo, E. *et al*. "Crosscutting Patterns and Design Stability: An Exploratory Analysis". Proc. of Int'l Conf. on Program Comprehension (ICPC). Vancouver, Canada, 2009.

[5] Figueiredo, E. *et al*. "On the Maintainability of Aspect-Oriented Software: A Concern-Oriented Measurement Framework". Proc. of the 12th European Conference on Software Maintenance and Reengineering (CSMR), 2008.

[6] Kiczales, G. *et al*. "Aspect-Oriented Programming". Proc. of the ECOOP Conference, 220-242, 1997.

[7] Marinescu. R. "Detection Strategies: Metrics-Based Rules for Detecting Design Flaws". Proc. of the Int'l. Conf. on Soft. Maintenance (ICSM), 350-359, Chicago, 2004.

[8] Robillard, M. and Murphy, G. "Representing Concerns in Source Code". ACM TOSEM, 16(1), Article No. 3, 2007.

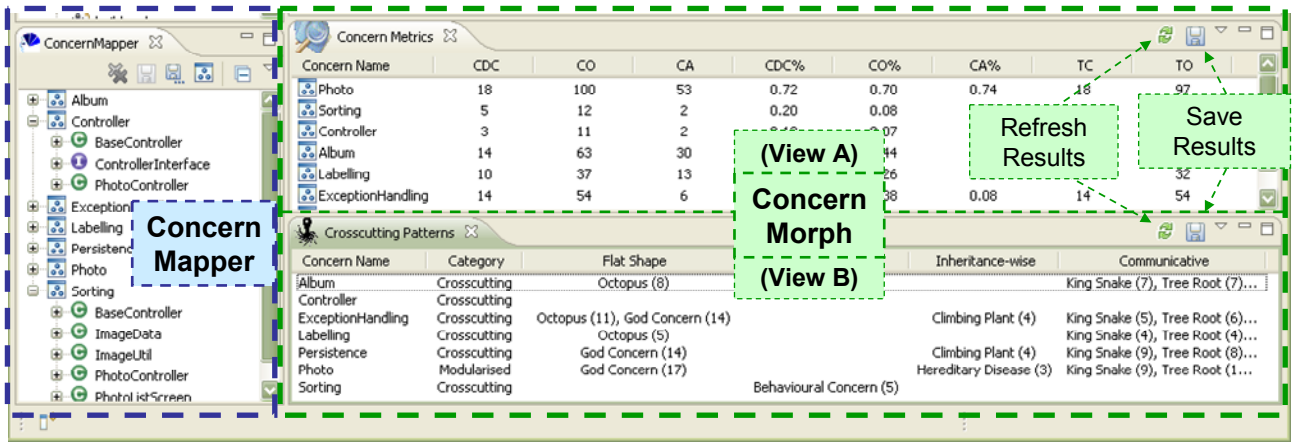

Figure 3. Screenshot showing concern measurements (View A) and crosscutting patterns (View B).