UFMG

lab-soft
software engineering laboratory

Identifying Code Smells with Multiple Concern Views

G. Carneiro, M. Silva, L. Mara, E. Figueiredo,
C. Sant'Anna, A. Garcia, and M. Mendonca

Kattiana Constantino
Prof.: Eduardo Figueiredo

Outline

- Introduction
- A Multiple View Approach to Visually Represent Concerns in Source Code
- The Exploratory Study Design
- Results and Data Analysis
- Conclusion

Introduction

- A **concern** is anything a stakeholder wants to consider as a conceptual unit, including domain-specific features, nonfunctional requirements, and design patterns.

- **Code smells** are anomalies in the source code that contribute to the degeneration of software design maintainability.
  - Classical examples are ***God Class***, ***Divergent Change*** and ***Feature Envy***.

Goals

- To understand to which extent the multiple views environment supports programmers in the detection of certain code smells
- To uncover successful detection strategies, based on the use of concern properties by the participants

Four Categories of Code Views with Concern Properties

- Concern's package-class-method structure
- Concern's inheritance-wise structure
- Concern dependency
- Concern dependency weight

A Multiple View Approach to Visually Represent Concerns in Source Code

The Concern's Package-Class-Method Structure View

- How a concern is realized through modularity units of a system, such as packages, classes, and methods;

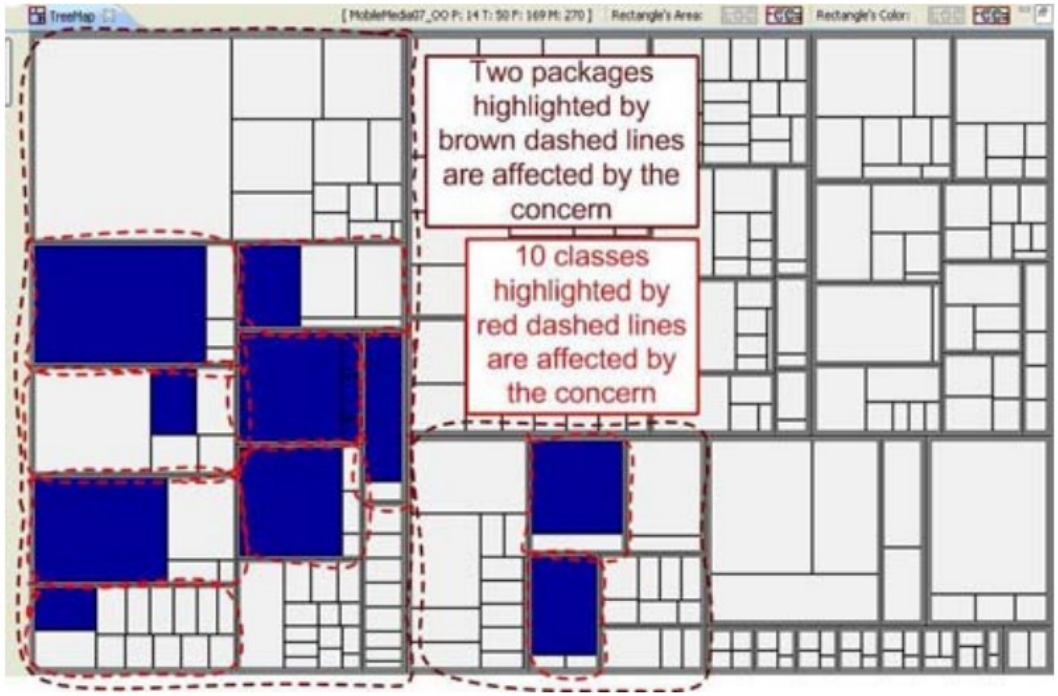


Figure 1: The Treemap View in the concern's package-class-method structure view 2

The Concern's Inheritance-Wise Structure View

- How a concern is dispersed through one or more inheritance trees;

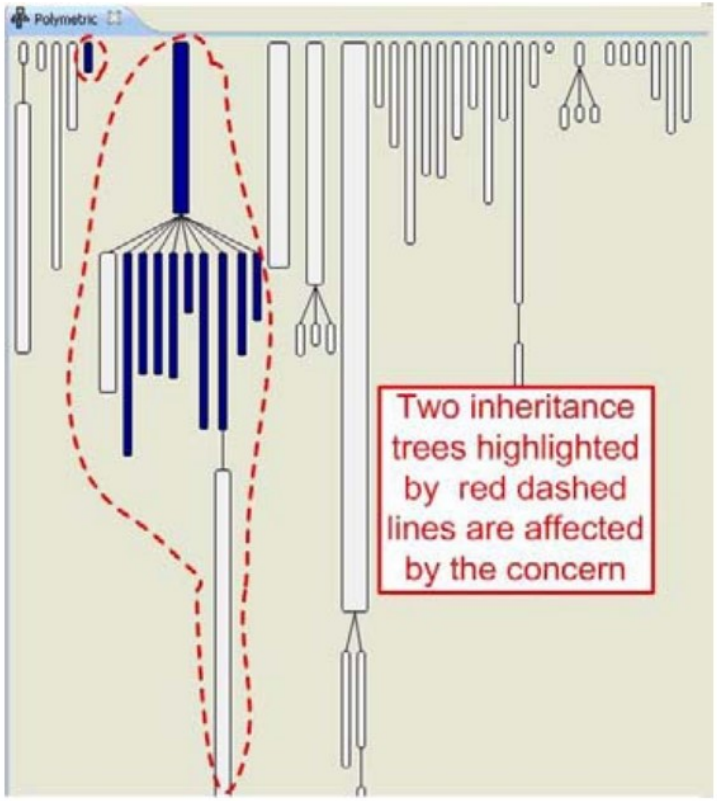


Figure 2: The Polymetric represeting the Concern's Inheritance-wise View

The Concern Dependency View

- How a concern affects the relationships among modules;

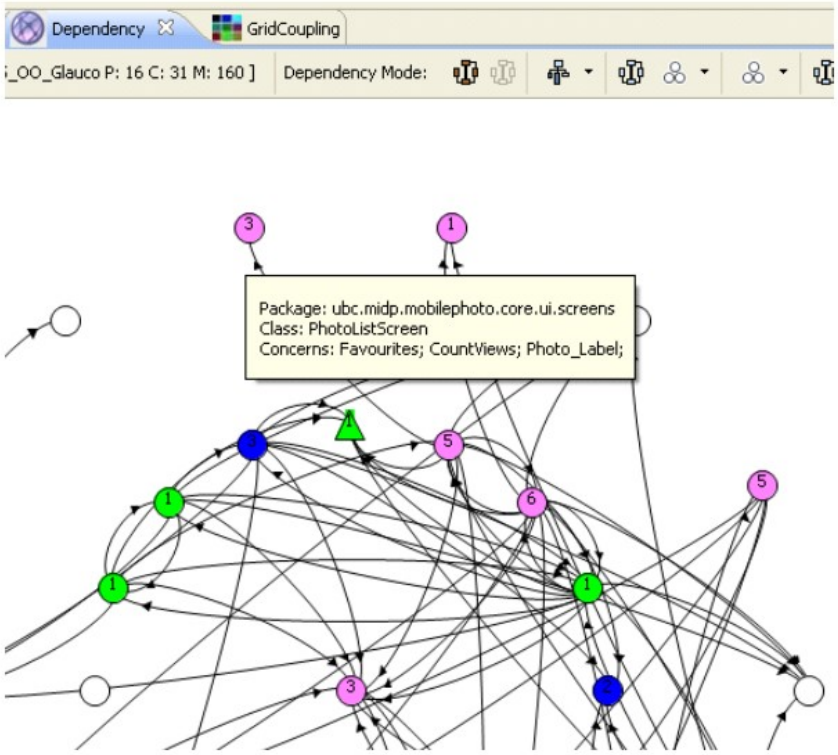
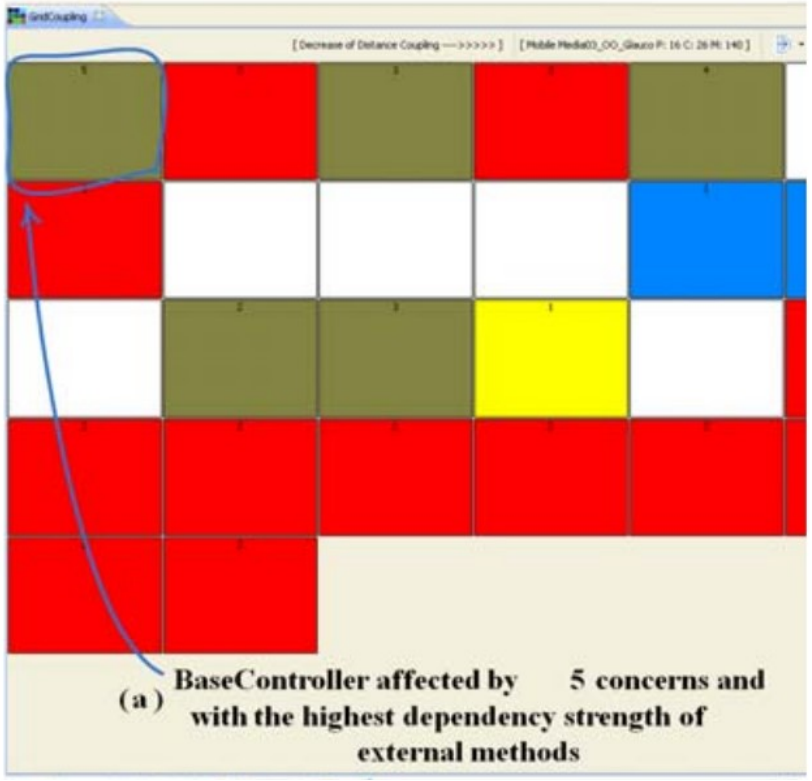


Figure 3: The Graph View representing the Concern Dependency View

The Concern Dependency Weight Views (a)

- How a concern can be perceived as affecting the weight with which modules are coupled to each other.



The Concern Dependency Weight Views (b)

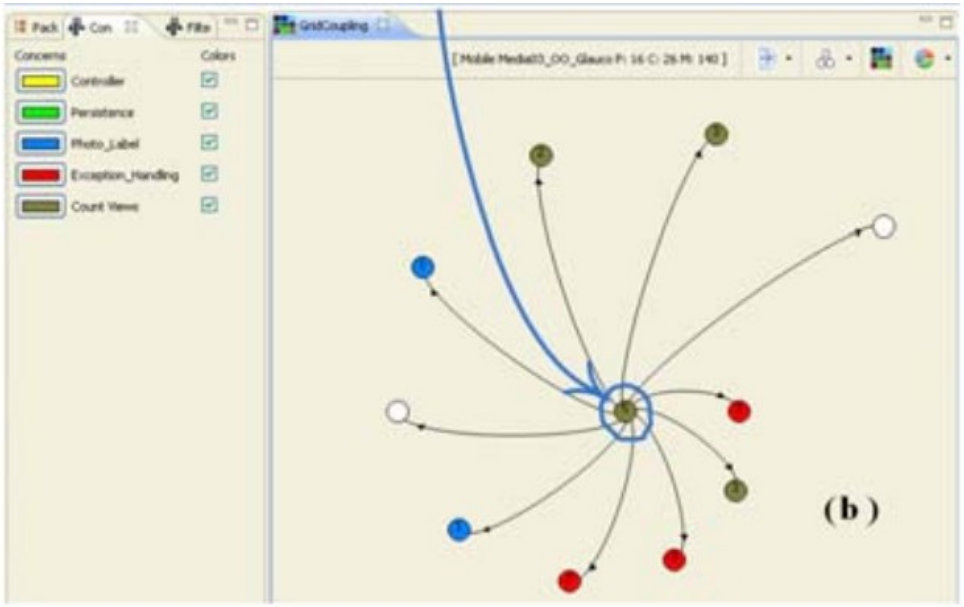


Figure 4: The Chess Board-like and the Spiral Representations in the Concern Dependency Weight

The Exploratory Study Design

Research Questions

- ***RQ1:*** To which extent the multiple views concern visualization supports the identification of code smells?

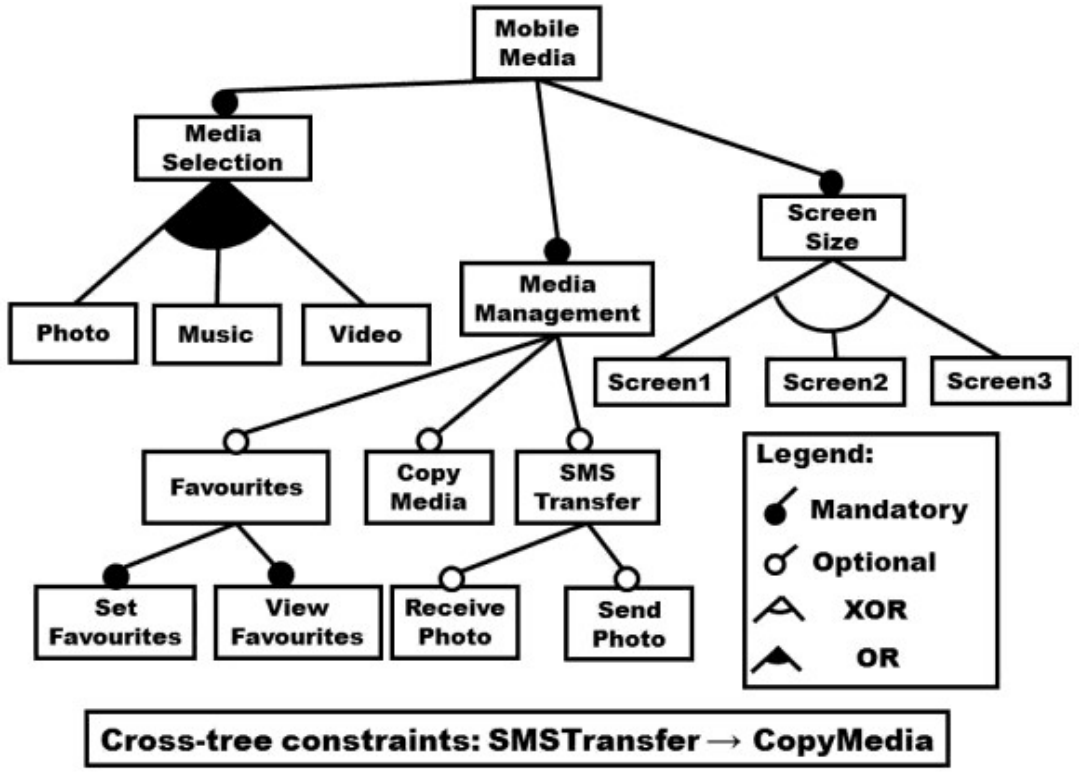
- ***RQ2:*** What are the possible identification strategies for code smells using the multiple views approach?

Code Smells Identification

- Feature Envy (FE)
- God Class (GC)
- Divergent Change (DC)

The Target System

- Mobile Media (4 KLOC distributed in 18 packages and 50 classes)



The Target System

- Its Java implementation is available and served as object in many studies.
- Its key concerns were previously identified by the developers and mapped to the source code
- Code smells had already been detected in this system using a heuristic approach.

The Tutorial Session

- Training Session – 24 hours – SourceMiner
- To analyze a program, called Health Watcher
- 28 basic questions regarding the tool functionalities
- Two experimenters were available online (email and chat) to provide complementary guidance and detailed explanation on how to use SourceMiner.

Results and Data Analysis

Collecting Data for Analysis

- Direct data collection:
  - Questionnaires performed by participants
- Indirect data collection
  - Logs, information, details from tool

Data Analysis

- Analysis of Precision and Recall (RQ1)

$$\text{Precision} = \frac{|\{\text{existing code smells}\} \cap \{\text{detected code smells}\}|}{|\{\text{detected code smells}\}|}$$

$$\text{Recall} = \frac{|\{\text{existing code smells}\} \cap \{\text{detected code smells}\}|}{|\{\text{existing code smells}\}|}$$

RQ1: Multiple Views Concern Visualization Support to Code Smell Detection

TABLE 1: PARTICIPANT-ORIENTED EVALUATION

	P1		P2		P3		P4		P5	
	Recall	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*
GC	**0,7	**0,9	0,5	0,4	0,4	0,4	0,4	0,8	1	0,4
DC	0,4	0,8	**0,8	**0,7	0,4	0,8	0,2	0,4	1	0,4
FE	**0,5	**0,5	*Not Performed		0,0	0,0	0,6	0,2	1	0,2

* Participant 2 did not performed FE code smell detection
** Best precision-recall pairs for each code smell

TABLE 2: CODE SMELL ORIENTED EVALUATION

	Recall	Precision
GC	0,6	0,6
DC	0,5	0,5
FE	0,4	0,2

RQ2: Uncovering Identification Strategies

TABLE 3: STRATEGIES ADOPTED TO IDENTIFY CODE SMELLS

	God Class	Divergent Change	Feature Envy
P1	** Inheritance + Concern	Dependency	** Dep. Weight (Grid) + Dep. Weight (Spiral)
P2	Inheritance + Pack-class-method + Concern	** Pack-class-method + Inheritance + Concern	Not Perfomed
P3	Dependency + Concern	Dependency + Grid + Concern	Pack-class-method + Concern
P4	Inheritance + Concern	Pack-class-method + Concern	Dep. Weight (Grid) + Dependency + Inheritance
P5	Inheritance + Grid + Concern	Grid	Dependency

** Best precision-recall pairs for each 1

Conclusion

- We analyzed how the tool supports the visual detection of recurring code smells. To this end, we conducted an exploratory study with two main goals.
  - First, we aimed at understand to which extent the multiple views environment supports programmers in the detection of certain code smells. Our analysis was driven by the computation of precision and recall in the answers given by programmers who participated in the study. The results show that the multiple views concern visualization provided useful support to identify the God Class and Divergent Change code smells.
  - Second, we aimed to uncover successful detection strategies, based on the use of concern properties by the participants. The low number of false negatives is an indication that, even having no access to the source code, participants found useful strategies to spot the code smells.

SourceMiner

- http://www.sourceminer.org/

UFMG

lab-soft
software engineering laboratory

Identifying Code Smells with Multiple Concern Views

G. Carneiro, M. Silva, L. Mara, E. Figueiredo,
C. Sant'Anna, A. Garcia, and M. Mendonca

Kattiana Constantino
Prof.: Eduardo Figueiredo