On the Impact of Crosscutting Concern Projection on Code Measurement

Eduardo Figueiredo (UFMG), Alessandro Garcia (PUC-Rio),
Marcelo Maia (UFU), Gabriel Ferreira (UFU),
Camila Nunes (PUC-Rio) and Jon Whittle (Lancaster University)

23 March 2011

Agenda

Metrics for Separation of Concerns

Study Settings

- Experiment Hypotheses
- The Participants
- Evaluation Procedures

Results and Key Findings

Ongoing and Future Work

Motivation

Many metrics for separation of concerns have been proposed

- We call **Concern Metrics**

Concern Metrics rely on the projection of concerns onto software artifacts

- **Source code** in this study

Some Concern Metrics

Who proposed them?

Sant'Anna [2]

- Concern Diffusion over Components (CDC)
- Concern Diffusion over Operations (CDO)
- Concern Diffusion over Lines of Code (CDLOC)

Ducasse, Girba, and Kuhn [3]

- Size, Touch, Spread, and Focus

[2] Sant'Anna, C. et al. **On the Reuse and Maintenance of. Aspect-Oriented Software: an Assessment Framework.** Proceeding of the Brazilian Symposium on Software Engineering (SBES), 2003.

[3] Ducasse, S.; Girba, T.; and Kuhn, A. **Distribution Map.** Proceeding of the International Conference on Software Maintenance (ICSM), 2006.

Other Concern Metrics

Eaddy, Aho, and Murphy [4]

- Lines of Concern Code (LOCC)
- Concentration and Dedication

Lopez-Herrejon and Apel [5]

- Number of Features (NOF)
- Feature Crosscutting Degree (FCD)

NOCA, NCC, …

[4] Eaddy, M., Aho, A., and Murphy, G. **Identifying, Assigning, and Quantifying Crosscutting Concerns.** Proceeding of the Workshop on Assessment of Contemporary Modularization Techniques (ACoM), 2007.

[5] Lopez-Herrejon, R. and Apel, S. **Measuring and Characterizing Crosscutting in Aspect-Based Programs: Basic Metrics and Case Studies.** Proc. of the Int'l Conf. on Fundamental Approaches to Soft. Eng. (FASE), 2006.

Quantifying Crosscutting

Source code of the MediaController class

MobileMedia [1]

```
public class MediaController extends AbstractController {

  public MediaController(...) {
    super(midlet, albumData, albumListScreen);
  }

  public boolean handleCommand(Command command) {
    String label = command.getLabel();
    String password;
    if (label.equals("Select Album")) {
      setCurrentAlbumName( getDisplay(midlet).getSelectedIndex() );
      password = getAlbumData().getPassword( getCurrentAlbumName() );
      if(password == null) {
        showMediaList(getCurrentAlbumName(), false, false);
      } else {
        PasswordScreen pwScreen = new PasswordScreen("Password", 1);
        pwScreen.setCommandListener(this);
        setCurrentScreen(pwScreen);
        return true;
      }
      return true;
    }

    if(label.equals("Password Given")){
      PasswordScreen pwScreen = (PasswordScreen) getCurrentScreen();
      password = getAlbumData().getPassword( getCurrentAlbumName() );
      if(pwScreen.getPassword().equals(password)) {
        showMediaList(getCurrentAlbumName(), false, false);
      }else{
        Alert alert = new Alert( "Error", "Invalid Password", AlertType.ERROR);
        getDisplay(midlet).setCurrent(alert, getDisplay(midlet));
      }
      return true;
    }
    return false;
  }

  public void showMediaList(String albumName, boolean sort, boolean favorite) {
```

[1] Figueiredo, E. et al. **Evolving software product lines with aspects: an empirical study on design stability.** Proceeding of the International Conference on Software Engineering (ICSE), 2008.

Quantifying Crosscutting

Security

```
public boolean handleCommand(Command command) {
    String label = command.getLabel();
    String password;
    if (label.equals("Select Album")) {
      setCurrentAlbumName( getDisplay(midlet).getSelectedIndex() );
      password = getAlbumData().getPassword( getCurrentAlbumName() );
      if(password == null) {
        showMediaList(getCurrentAlbumName(), false, false);
      } else {
        PasswordScreen pwScreen = new PasswordScreen("Password", 1);
        pwScreen.setCommandListener(this);
        setCurrentScreen(pwScreen);
        return true;
      }
      return true;
    }
    ...
```

Measures for Security

Security

CDC = 1
CDO = 1
CDLOC = 8
LOCC = 20
...

```
public class MediaController extends AbstractController {
  public MediaController(...) {
    super(midlet, albumData, albumListScreen);
  }
  public boolean handleCommand(Command command) {
    String label = command.getLabel();
    String password;
    if (label.equals("Select Album")) {
      setCurrentAlbumName( getDisplay(midlet).getSelectedIndex() );
      password = getAlbumData().getPassword( getCurrentAlbumName() );
      if(password == null) {
        showMediaList(getCurrentAlbumName(), false, false);
      } else {
        PasswordScreen pwScreen = new PasswordScreen("Password", 1);
        pwScreen.setCommandListener(this);
        setCurrentScreen(pwScreen);
        return true;
      }
      return true;
    }
    if(label.equals("Password Given")){
      PasswordScreen pwScreen = (PasswordScreen) getCurrentScreen();
      password = getAlbumData().getPassword( getCurrentAlbumName() );
      if(pwScreen.getPassword().equals(password)) {
        showMediaList(getCurrentAlbumName(), false, false);
      }else{
        Alert alert = new Alert( "Error", "Invalid Password", AlertType.ERROR);
        getDisplay(midlet).setCurrent(alert, getDisplay(midlet));
      }
      return true;
    }
    return false;
  }
  public void showMediaList(String albumName, boolean sort, boolean favorite) {
    ...
  }
  ...
}
```

Research Question / Hypothesis

Research Question (RQ1)

Can different developers identify the same code fragments for a concern?

Hypothesis (H1)

The projection of crosscutting concerns into source code does not depend on individual differences between developers

Research Question / Hypothesis

Research Question (RQ2)

Do measurements significantly vary depending on who performed the concern projection?

Hypothesis (H2)

Concern metrics can be precisely quantified regardless of who projected the crosscutting concern into the system

Study Settings

The Selected Concerns

Participants and Institutions

Experimental Procedures

The Selected Concerns

We selected 10 concerns from two applications

- 6 concerns from Health Watcher [6]
- 4 concerns from MobileMedia

Eight crosscutting concerns

- Concurrency, Distribution, Persistence, Exception Handling (2x), Security, Sorting, and Favourites

Two non-crosscutting concerns

- Business and View (GUI)

[6] Greenwood, P. et al. On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study. Proceeding of the 21st European Conference on Object-Oriented Programming (ECOOP), 2007.

Participants per Replication

5 **replications** with 80 **participants** from 4 **institutions**

1st Replication (FRB - Health Watcher)

- 6 interns and young developers with less than 3 years experience in software development

2nd Replication (Lancs - Health Watcher)

- 13 undergraduate Computer Science students

3rd Replication (PUC-Rio - Health Watcher)

- 16 graduated Master and PhD students
- Organized in groups of two or three people

Participants per Replication

4th Replication (PUC-Rio - MobileMedia)

- 16 graduated Master and PhD students
- None of them participated in the 3rd replication

5th Replication (UFMG - MobileMedia)

- 32 undergraduate Computer Science students
- Organized in groups of two or three people

Experimental Procedures

1. Each replication was limited by 1,5 hour
   - Including a 15-minute training session
2. We give the participants
   - The source code of four classes
   - The description of the concerns
3. We asked them to project (**by hand**) the concerns onto the given code
4. After each experiment, we measure the rate of **hits**, **false positives** and **false negatives**

Measure of Hits

Concern Name: **Hits** | False Positives | False Negatives

Hits % = Correctly Tagged + Correctly Not Tagged / # Lines of Code

Correctly Tagged = 2 | # LOC = 10

Correctly Not Tagged = 8 | **Hits % = 100 %**

```
15  public void insert(Employee employee)
16    throws ObjectNotValidException, ObjectAlreadyInserted
17    manager.beginExecution(employee.getLogin());   Concurrency
18    if (employeeRepository.exists(employee.getLogin())) {
19      throw new ObjectAlreadyInsertedException();   Not Concurrency
20    } else {
21      employeeRepository.insert(employee);
22    }
23    manager.endExecution(employee.getLogin());   Concurrency
24  }
```

Measure of False Positives

Concern Name: Hits | **False Positives** | False Negatives

FP % = # False Positives / # LOC not implementing concern

False Positives = 1

LOC not implementing concern = 8

FP % = 12.5 %

```
15  public void insert(Employee employee)
16    throws ObjectNotValidException, ObjectAlreadyInserted
17    manager.beginExecution(employee.getLogin());   Concurrency
18    if (employeeRepository.exists(employee.getLogin())) {
19      throw new ObjectAlreadyInsertedException();
20    } else {
21      employeeRepository.insert(employee);   False Positive
22    }
23    manager.endExecution(employee.getLogin());
24  }
```

Measure of False Negatives

Concern Name: Hits | False Positives | **False Negatives**

FN % = # False Negatives / # Concern LOC

False Negatives = 1

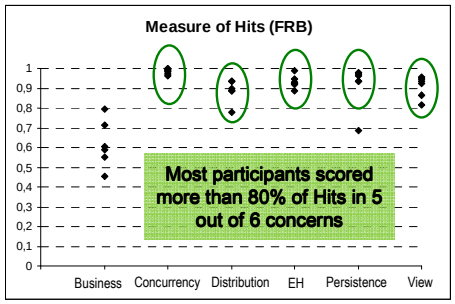
Concern LOC = 2

FN % = 50 %

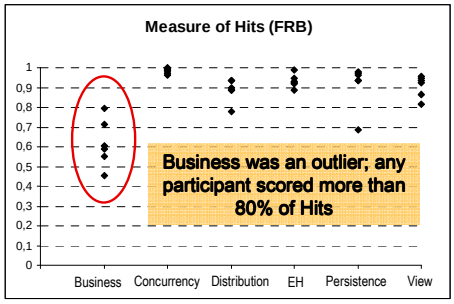
```
15  public void insert(Employee employee)
16    throws ObjectNotValidException, ObjectAlreadyInserted
17    manager.beginExecution(employee.getLogin());   Concurrency
18    if (employeeRepository.exists(employee.getLogin())) {
19      throw new ObjectAlreadyInsertedException();
20    } else {
21      employeeRepository.insert(employee);
22    }
23    manager.endExecution(employee.getLogin());   False Negative
24  }
```

Results

Results of the 1st Replication



Results of the 1st Replication



The Issue of Business

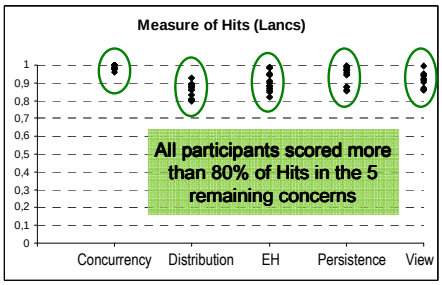
Table 1. Comparison of Business to other concerns (FRB)

		S1	S2	S3*	S4	S5	S6
Business	Hits	60.5%	59.8%	55.2%	79.4%	45.5%	71.3%
	False Positives	0%	0%	0%	4.9%	0%	0%
	False Negatives	55.1%	56.1%	62.4%	26.8%	76.1%	40.0%
Average Others	Hits	86.0%	94.3%	93.4%	95.2%	91.0%	94.6%
	False Positives	9.7%	1.0%	2.5%	1.2%	2.9%	1.0%
	False Negatives	44.1%	39.0%	35.6%	29.9%	50.8%	36.7%

Business is also an issue in the measurement of False Negatives

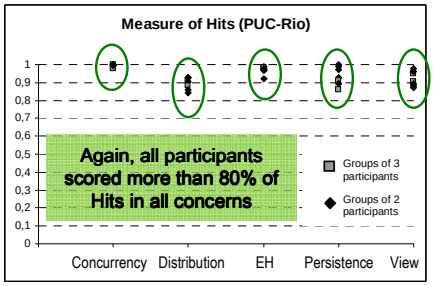
Results of the 2nd Replication

Due to time constraints we remove Business from the 2nd study replication



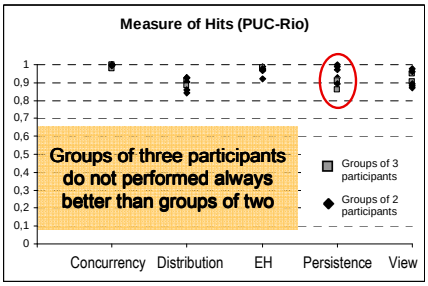
Results of the 3rd Replication

The participants were organized in groups of two or three people



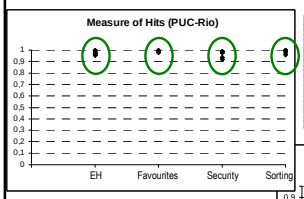
Results of the 3rd Replication

The participants were organized in groups of two or three people

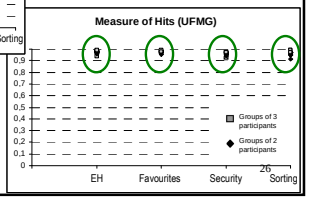


Groups of three participants do not performed always better than groups of two

4th and 5th Replications



Confirmed previous observations
All participants scored more than 80% of Hits for all concerns



General Observations

In general, participants score about the same for every concern

- All participants scored more than 95 % for Concurrency
- Most subjects scored between 80% and 90% for Distribution …

Distribution seems the hardest crosscutting concern to be projected

The measure of false positives was generally low

- < 10 % for all crosscutting concerns

The measure of false negatives was generally high

- > 20 % for all crosscutting concerns

General Observations

In general, participants score about the same for every concern

- All participants scored more than 95 % for Concurrency
- Most subjects scored between 80% and 90% for Distribution …

Distribution seems the hardest crosscutting concern to be projected

Hypothesis (H1)

The projection of crosscutting concerns into source code does not depend on individual differences between developers.

The measure of false positives was generally low

- < 10 % for all crosscutting concerns

The measure of false negatives was generally high

- > 20 % for all crosscutting concerns

General Observations

In general, subjects score more or less the same for every concern

- All participants scored more than 95 % for Concurrency
- Most subjects scored between 80% and 90% for Distribution …

Distribution seems the hardest crosscutting concern to be projected

Additional Finding (1)

Developers usually do not assign a line of code to a concern if they are unsure about it

The measure of false positives was generally low

- < 10 % for all crosscutting concerns

The measure of false negatives was generally high

- > 20 % for all crosscutting concerns

General Observations

In general, subjects score more or less the same for every concern

- All participants scored more than 95 % for Concurrency
- Most subjects scored between 80% and 90% for Distribution …

Distribution seems the hardest crosscutting concern to be projected

Additional Finding (2)

Developers usually miss out code fragments that are realizing a concern

The measure of false positives was generally low

- < 10 % for all crosscutting concerns

The measure of false negatives was generally high

- > 20 % for all crosscutting concerns

The Impact on Concern Metrics

Procedures

Hypothesis

Concern metrics can be precisely quantified regardless of who projected the crosscutting concern into the system

1. Apply a set of metrics into concern projections from all participants
2. Verify the variance of a given metric across projections from different participants

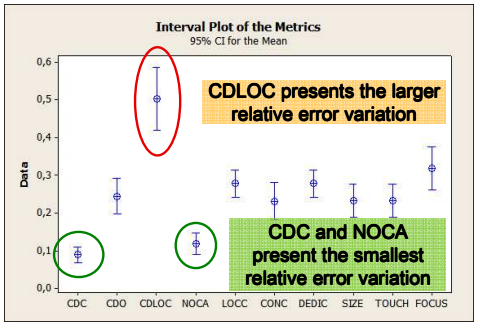
Average (and Reference Value)

	CDC	CDO	CDLOC	LOCC	NOCA
Business	3 (3)	18 (30)	24 (12)	65 (205)	3 (7)
Concurrency	1 (1)	2 (2)	8 (10)	6 (5)	0.8 (1)
Distribution	2 (2)	8 (14)	21 (42)	34 (53)	0.5 (0)
EH	2 (2)	8 (4)	23 (14)	39 (44)	0 (0)
Persistence	2 (1)	8 (8)	14 (6)	37 (37)	1.4 (2)
View	1 (1)	1 (1)	11 (2)	10 (38)	0 (0)

For some metrics, the average matches the reference value

But, this is not the case for other metrics

Relative Error per Concern Metric



Study Constraints

The concern metrics were not automated

- Hand-count measurement might be error prone

We have not taken any special care to select the subjects and institutions

- We consider random choices

The selected systems may not be representative of the industrial practice

- Two systems from different domains
- Heavily based on industry-strength technologies

Conclusions

This study aimed to quantify the impact of concern projections on measurement of concern properties

The selected crosscutting concerns could be projected with more than 80% precision

The conservative behaviour makes developers miss code where concerns should be projected

CDLOC can be considered unreliable since its values highly varies across projections

CDC and NOCA metrics were found to be the most reliable ones

Ongoing and Future Work

Additional concerns

- Our results are limited to 10 concern instances

Different systems and domains

- We rely on the source code of only two systems

Other research questions

- What recurring mistakes developers make when projecting crosscutting concerns?
- What kind of code fragments developers usually consider relevant to a crosscutting concern?

On the Impact of Crosscutting Concern Projection on Code Measurement

Eduardo Figueiredo (UFMG), Alessandro Garcia (PUC-Rio),
Marcelo Maia (UFU), Gabriel Ferreira (UFU),
Camila Nunes (PUC-Rio) and Jon Whittle (Lancaster University)

23 March 2011