On the Effectiveness of Concern Metrics to Detect Code Smells: An Empirical Study

Presented by Vagner Clementino

Department of Computer Science
Federal University of Minas Gerais (UFMG)
Software Quality and Measurement - 2015

Outline

About the Paper

Context

Motivation

Proposed Work

Results

Threats to Validity

Conclusions

About the Paper

- Who:
  - Juliana Padilha, Juliana Pereira, Eduardo Figueiredo, Jussara Almeida[1]
  - Alessandro Garcia, Cláudio Sant'Anna[2]
- When: 2014
- Where:
  - 26^{th} International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014, Proceedings.

[1] UFMG

[2] PUC-RIO

Software Maintainability Measurement

- Software Measurement has been proposed to find symptoms of particular design flaw.
- Software Metrics are the key means for assessing the system maintainability[1].
- Traditionally software metrics have been used to evaluate the maintainability of software programs through the identification of code smells.

Traditional vs concern Metrics

- Traditional metrics quantify properties of software modules.
- Concern metrics quantify concern properties, such as *scattering* and *tangling*.
- Some *code smells* may be motivated by a poor separation of Concerns.

Concerns' Definition

- A Concern is any important property or area of interest of a system that we want to treat in a modular way[8].
- Examples:
  - Functional concerns.
  - Quality of service.
  - Organizational concerns.
  - Global system concerns.

A Less Formal Definition

- A concern can be defined as something that is of interest or significance to a stakeholder or a group of stakeholders[9].

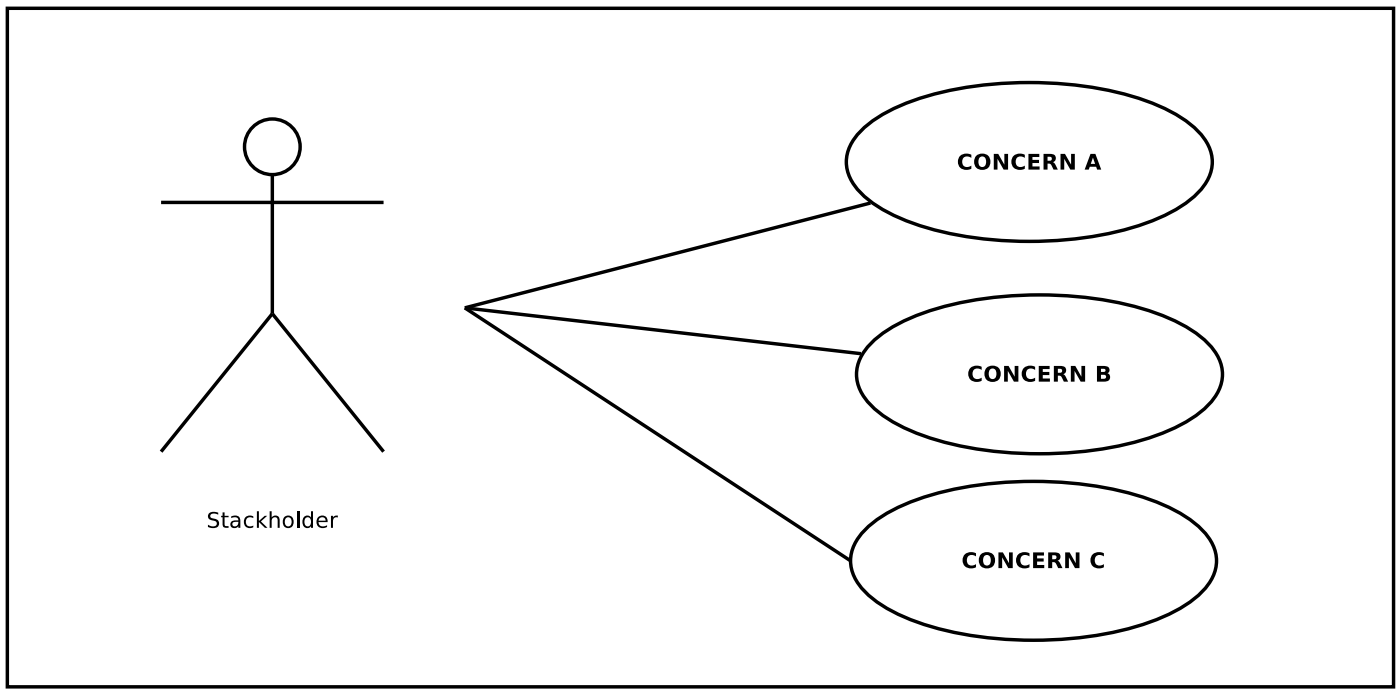

Figure 1 : Stakeholder with yours concerns

Crosscutting Concerns

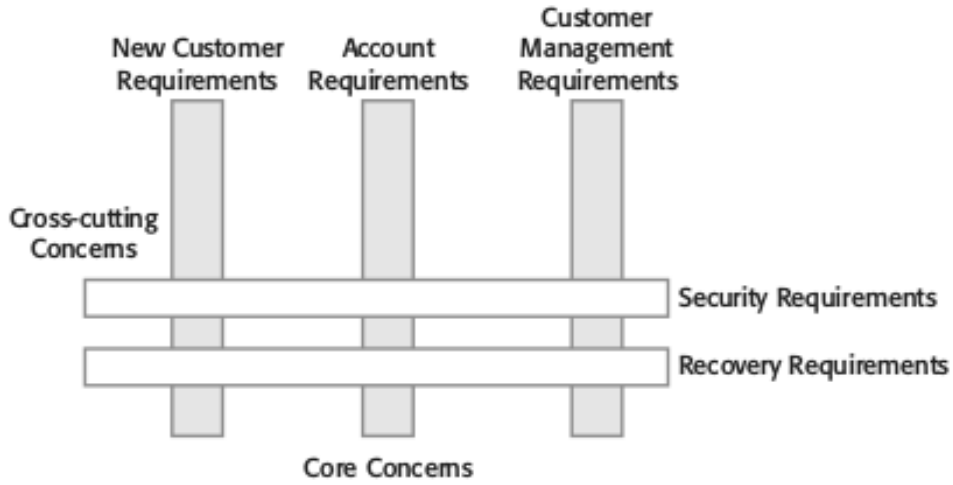

Figure 2 : Adapted from [9].

- Two typical problems:
  - *Scattering*
  - *Tangling*

Scattering

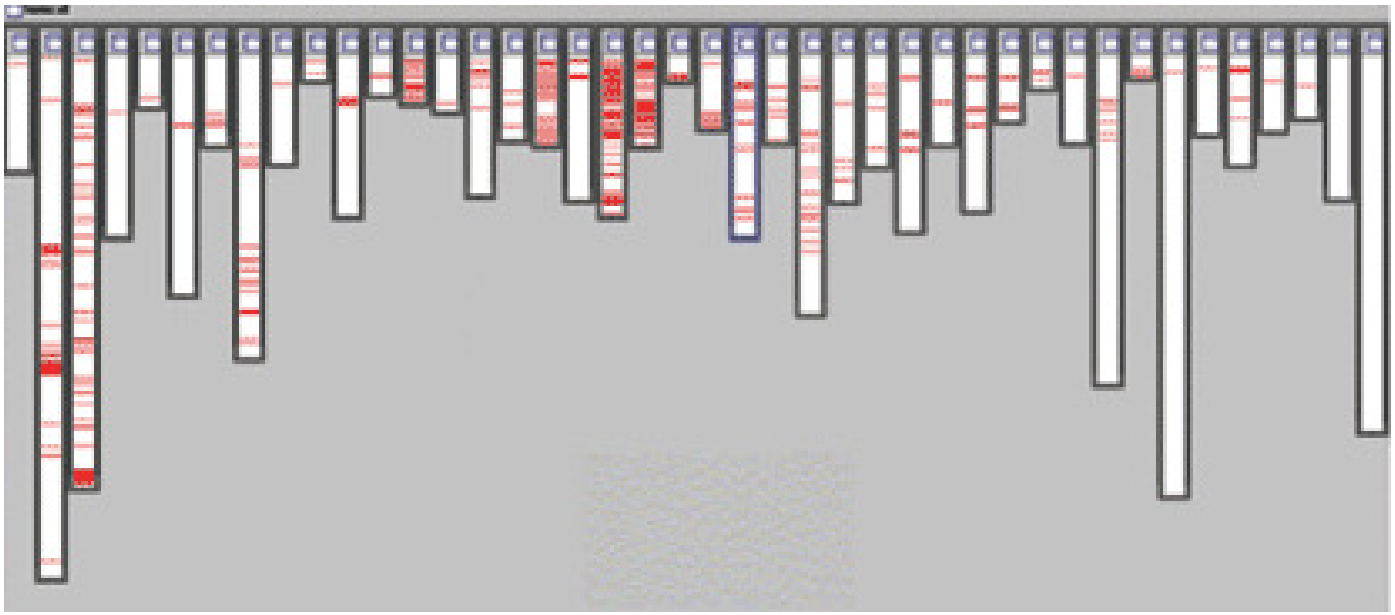

Figure 3 : Logging concern in Tomcat.

Tangling

```
synchronized void put (SensorRecord rec )
{
   // Check that there is space in the buffer; wait if not
   if ( numberOfEntries == bufsize)
      wait () ;
   // Add record at end of buffer
   store [back] = new SensorRecord (rec.sensorId, rec.sensorVal) ;
   back = back + 1 ;
   // If at end of buffer, next entry is at the beginning
   if (back == bufsize)
      back = 0 ;
   numberOfEntries = numberOfEntries + 1 ;
   // indicate that buffer is available
   notify () ;
} // put
```

Figure 4 : Tangling of buffer management and synchronization code. Adapted from [9].

Motivation

- Concern metrics have been used in several empirical studies:
  - Comparing aspect-oriented and object-oriented programming techniques
  - Identifying crosscutting concerns that should be refactored
- However, there is a lack of studies about the *effectiveness of concern metrics to support code smell detection*.

Proposed Work

- It has been proposed an empirical investigation of the effectiveness of concern metrics compared with traditional metrics on the identification of code smells.
- That study compares the trade-offs on the *recall* and *time efficiency* of code smell detection.

Traditional Metrics

- It was selected a set of the most widely used metrics to be a baseline.
- *Chidamber-Kemerer (CK) Suite*[1].
- These metrics has been largely used in literature.[2, 3, 7]

Traditional Metrics

Metric	Definition
Coupling between Objects (CBO)	Number of classes from which a class calls methods or accesses attributes.
Lack of Cohesion in Methods (LCOM)	It divides pairs of methods that do not access common attributes by pairs that access common attributes.
Lines of Code (LOC)	Total number of lines of code.
Number of Attributes (NOA)	Number of attributes defined in a class.
Number of Methods (NOM)	Number of methods defined in a class.
Weighted Methods per Class (WMC)	Number of methods and their parameters in a class

Concern Metrics

- Concern Metrics goals are the identification of specific design flaws or design degeneration caused by poor modularization of concerns.
- These metrics have been successfully used in a number of studies related to software maintainability[3, 4, 5].

Concern Metrics

Metric	Definition
Concern Diffusion over Components (CDC)	Number of classes whose main purpose is to contribute to the implementation of a concern and the number of other classes that access them.
Concern Diffusion over Operations (CDO)	Number of methods whose main function is to implement a concern.
Concern Diffusion over LOC (CDLOC)	Number of transition points for each concern through the lines of code. Transition points are points in the code where there is a “concern switch”.
Number Concerns per Component (NCC)	Number of concern in each class.

Code Smells

- **Divergent Change:** A class is often changed in different ways for different reasons.
- **Shotgun Surgery:** This code smell is somehow the opposite of Divergent Change. Making a kind of change in a class leads to a lot of small changes in many different classes.
- **God Class:** An object that knows too much or does too much.

Study Settings

- Target System
  - **Health Watcher:** Web-based information system that supports the registration and management of complaints to the public health system.
  - **MobileMedia:** It is a *software product line* (SPL) for applications that manipulate photo, music and video on mobile devices, such as mobile phones.
  - They have been previously used in other maintainability-related studies.
  - The authors have access to developers and experts.

Study Settings

- Code Smells Reference List
  - Software's experts make a list of detected code smells from each system.
  - It was promoted a discussions among experts in order to achieve a consensus.

System	Code Smell	Classes in the Reference List
Health Watcher	Divergent Change	EmployeeRecord, HealthWatcherFacade, HealthUnitRecord, IFacade, PersistenceMechanism, IPersistenceMechanism, ServletInsertEmployee, ServletSearchComplaintData, ServletUpdateComplaintData, ServletUpdateHealthUnitData, ComplaintRecord, HealthWatcherFacadeInit
	Shotgun Surgery	EmployeeRepositoryRDB, IEmployeeRepository, ComplaintRecordRDB, IComplaintRepository, IPersistenceMechanism, PersistenceMechanism, IHealthUnitRepository, HealthUnitRepositoryRDB
	God Class	HealthWatcherFacade, HealthWatcherFacadeInit, PersistenceMechanism
Mobile Media	Divergent Change	ImageMediaAccessor, MediaController, MediaAcessor, MediaListController
	Shotgun Surgery	ControllerInterface, MediaAccessor, ScreenSingleton

Study Settings

- Background of Subjects
  - 54 subjects, named S_1 to S_{54}
  - Federal University of Minas Gerais (UFMG)
    - 11 IT professional.
    - 4 PhD candidates.
    - 12 undergraduate students.
  - Lancaster University (UK)
    - 14 PhD candidate.
    - 13 undergraduate student.

Study Settings

- Background of Subjects
  - Before running the experiment the subjects answered a background questionnaire.

Divergent Change		*Traditional*	*Concern*	*Hybrid*	*No Answer*
Knowledge	Class Diagram	S5 - S6	S9 - S11	S14 - S24	S1, S2, S3, S7, S8, S12, S13, S18
	Java Programming	S5 - S6	S9 - S11	S14 - S24	
	Measurement	-	S9	S16, S20, S22, S24	
	Academic Experience	S4, S6	S9	S19, S21-S24	
	Work Experience	S5	S10,S11	S14 - S17, S20	
Shotgun Surgery					
Knowledge	Class Diagram	S28, S29	S31, S32	S34 - S37	S25, S26, S30, S33, S38
	Java Programming	S28, S29	S31, S32	S34 - S37	
	Measurement	-	S31	S35, S36	
	Academic Experience	S27, S29	S31	S39, S41-S44	
	Work Experience	S28	S32	S33 - S37, S40	
God Class					
Knowledge	Class Diagram	S46	S48 - S50	S51 - S54	-
	Java Programming	S45, S46	S48 - S50	S51 - S54	
	Measurement	-	S49 - S50	S52 -S54	
	Academic Experience	S46, S47	S49, S50	S52-S54	
	Work Experience	S45	S48	S51	

Study Settings

- Subjects' Skills:
  - *60%* have *moderate to high* knowledge in *Class Diagram* and *Java Programming*.
  - *71%* have *moderate to high knowledge* in at least one topic.
  - *31%* have *low to none* knowledge in all topics.

Study Settings

- Experimental Tasks:
  - The study was preceded by a 30-minute training session to allow subjects' familiarization with the evaluated metrics and the target code smells.
  - The subjects receive a document with a brief explanation and a partial view of the system design as a Class Diagram, and a description of the concerns involved in the respective analyzed system.

Evaluation Metrics

- It was defined three metrics based on the reference lists:
  - *True Positive (TP):* number of *correctly* identified code smells by a subject.
  - *False Positive (FP):* number of *wrongly* identified code smells by a subject.
  - *False Negative (FN):* number of code smells a subject *missed out*.

Recall and Precision

- $Recall(R) = \frac{TP}{TP+FN}$
- $Precision(P) = \frac{TP}{TP+FP}$
- The focus was mainly on *recall* because it is a measure of completeness.
- High *recall* means that the subject was able to identify most code smells in the system.

Results Divergent Change

Group	Traditional						Concern						
Subject	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11		
R(%)	17	17	17	33	25	25	100	100	33	25	50		
P(%)	67	50	40	50	17	25	63	100	100	25	29		
T(min)	15	15	40	38	41	36	26	29	29	15	33		
Group	Hybrid												
Subject	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
R(%)	75	8	25	50	100	25	50	0	50	25	50	25	50
P(%)	100	50	75	25	67	33	40	0	67	17	40	17	50
T(min)	40	31	23	36	27	39	24	11	18	19	13	13	12

Results Divergent Change

- Concern and hybrid groups achieved better results than traditional group.
- The average recall of the concern group was 62%.
- The best achievement from the traditional metric was 33%.
- *Concern metrics* are an effective means to detect Divergent Change.

Results Shotgun Surgery

Group	Traditional					Concern		
Subject	S25	S26	S27	S28	S29	S30	S31	S32
R(%)	13	13	0	67	33	75	25	33
P(%)	25	33	0	25	25	35	40	25
T(min)	6	10	27	12	14	13	28	14

Group	Hybrid											
Subject	S33	S34	S35	S36	S37	S38	S39	S40	S41	S42	S43	S44
R(%)	13	50	67	33	33	33	0	0	33	0	0	0
P(%)	25	80	6	33	25	33	0	0	20	0	0	0
T(min)	35	14	19	15	4	10	14	9	21	3	7	5

Results Shotgun Surgery

- No group achieved good results.
- The used metrics cannot properly indicate Shotgun Surgery.
- The spent time (on average) was less than in others code smells detection.

Results God Class

Group	Traditional			Concern			Hybrid			
Subject	S45	S46	S47	S48	S49	S50	S51	S52	S53	S54
R(%)	33	33	67	100	67	100	33	100	100	100
P(%)	33	33	67	75	100	75	50	100	60	75
T(min)	18	25	27	37	66	43	22	53	51	35

Results God Class

- Traditional metrics when used in isolation do not so good to detect God Class.
- The join of concern and traditional metrics (hybrids) seems to be the best choice.

Results: Research Questions

- **RQ1.** *How accurate do concern metrics perform in comparison with traditional metrics to detect code smells?*
  - It was performed an *unpaired t-test* with 90% confidence level[6].
  - The *unpaired t-test* can be used to determine if two sets of data are significantly. different

Results: Research Questions

- For concerns metrics two confidence intervals computed *do* overlap.
- This mean that statistically the results for both systems *are not* different.
- In another way the concern metrics are system-independent.

Systems	Health Watcher (HW)			Mobile Media (MM)		
Groups	*Traditional (T)*	*Concern (C)*	*Hybrid (H)*	*Traditional (T)*	*Concern (C)*	*Hybrid (H)*
All	(9,1;22,4)	(37,5;95,7)	(10,9;57,5)	(27,8;53,1)	(38,5;86,5)	(27,1;52,7)
DC	(11,6;30,4)	(12,5;142,9)	(-22,7;94,8)	(23,9;26,5)	(-41,4;116,4)	(27,2;57,9)
SS	(-4,0;21,3)	(-107,9;207,9)	(-85,3;148,3)	(-57,3;157,3)	(28,9;37,9)	(6,4;33,4)
GC	-	-	-	(11,2;77,4)	(56,9;121,1)	(43,8;123)

Results: Research Questions

- Concern metrics produce significantly higher recall, compared to traditional metrics for the Health Watcher system.
- For the MobileMedia system there was a statistical tie, but results are better for the concern metric.

Results: Research Questions

- **RQ2.** *Does background of subjects impact the efficiency of the detected code smell?*
  - It was applied a 2^k full factorial design with $k = 2$.[6]
  - Developers' work experience *vs* the time spent in code smells detection.
  - Low Experience: $\leq$ *6 months* High Experience: $>$ *6 months*.
  - Short Time: $\leq$ *33 minutes* Long Time: $>$ *33 minutes*.

Results: Research Questions

- **RQ2.** *Does background of subjects impact the efficiency of the detected code smell?*
  - Results shows that the recall tends to increase with the work experience and the time spent in the detection.
  - *96%* can be attributed to the time spent in the detection.
  - *04%* is due to variations in the subject's work experience.
  - *01%* for the interaction of these two factors.

Results: Research Questions

- **RQ3.** *Is there a combination of metrics that increases recall of code smell detection?*
  - Subjects reported the metrics they considered useful for each code smell.
  - The result was based in the metrics that were considered useful by at least three subject.
  - It was analyzed to metrics with average of recall higher than 30%.

Results: Research Questions

- Divergent Change
  - Number Concern per Component (NCC)
  - Lack of Cohesion in Methods (LCOM)
  - Concern Diffusion over Components (CDC)
  - Line of Code (LOC)

Metrics	NCC	LCOM	CDC	LOC
Subjects who used this metric	S7, S8, S9, S11, S12, S14, S15, S16, S20, S23, S24	S1, S2, S4, S6, S12, S13, S14, S15, S17, S22, S24	S8, S10, S23	S2, S17, S20
Average of recall	60%	34%	50%	31%

Results: Research Questions

- Shotgun Surgery
  - Coupling between Object (CBO)
  - Concern Diffusion over Components (CDC)
  - Number Concern per Component (NCC)

Metrics	CBO	CDC	NCC
Subjects who used this metric	S25-S29, S37, S39, S40, S42-S44	S30, S31, S33, S40, S43	S32, S35, S36, S37
Average of Recall	15%	23%	42%

Results: Research Questions

- God Class
  - Coupling between Object (CBO)
  - Lack of Cohesion in Methods (LCOM)
  - Weighted Methods per Class (WMC)
  - Concern Diffusions over LOC (CDLOC)

Metrics	CBO	LCOM	WMC	LOC	CDLOC
Subjects who used	S46, S47, S51, S54	S45, S51, S53, S54	S47, S52, S53	S52, S53, S54	S48, S49, S53
Average of recall	67%	67%	89%	100%	89%

Threats to Validity

- The conclusions are restricted to the involved metrics, code smells, and target software systems.
- Language-specific.
- *Quasi*-Experiment.

Conclusions

- The time spent in code smell detection is more relevant than the developer's expertise.
- Concern metrics are clearly useful to detect Divergent Change and God Class.
- The concern metric *Number of Concerns per Component* is a reliable indicator of *Divergent Change*.

Questions?

References I

[1] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, Jun. 1994. [Online]. Available: http://dx.doi.org/10.1109/32.295895

[2] F. Ferrari, R. Burrows, O. Lemos, A. Garcia, E. Figueiredo, N. Cacho, F. Lopes, N. Temudo, L. Silva, S. Soares, A. Rashid, P. Masiero, T. Batista, and J. Maldonado, “An exploratory study of fault-proneness in evolving

References II

aspect-oriented programs," in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 65–74. [Online]. Available: http://doi.acm.org/10.1145/1806799.1806813

References III

[3] E. Figueiredo, N. Cacho, C. Sant'Anna, M. Monteiro, U. Kulesza, A. Garcia, S. Soares, F. Ferrari, S. Khan, F. Castor Filho, and F. Dantas, "Evolving software product lines with aspects: An empirical study on design stability," in *Proceedings of the 30th International Conference on Software Engineering*, ser. ICSE '08. New York, NY, USA: ACM, 2008, pp. 261–270. [Online]. Available: http://doi.acm.org/10.1145/1368088.1368124

References IV

[4] A. Garcia, C. Sant'Anna, E. Figueiredo, U. Kulesza, C. Lucena, and A. von Staa, "Modularizing design patterns with aspects: A quantitative study," in *Proceedings of the 4th International Conference on Aspect-oriented Software Development*, ser. AOSD '05. New York, NY, USA: ACM, 2005, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/1052898.1052899

References V

[5] P. Greenwood, T. Bartolomei, E. Figueiredo, M. Dosea, A. Garcia, N. Cacho, C. Sant'Anna, S. Soares, P. Borba, U. Kulesza, and A. Rashid, "On the impact of aspectual decompositions on design stability: An empirical study," in *ECOOP 2007 – Object-Oriented Programming*, ser. Lecture Notes in Computer Science, E. Ernst, Ed. Springer Berlin Heidelberg, 2007, vol. 4609, pp. 176–200. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73589-2_9

References VI

[6] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, ser. Wiley professional computing. Wiley, 1991. [Online]. Available: http://books.google.com.br/books?id=HetQAAAAMAAJ

References VII

[7] R. Marinescu, “Detection strategies: metrics-based rules for detecting design flaws,” in *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*, Sept 2004, pp. 350–359.

[8] M. P. Robillard and G. C. Murphy, “Representing concerns in source code,” *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 1, Feb. 2007. [Online]. Available: http://doi.acm.org/10.1145/1189748.1189751

References VIII

[9] I. Sommerville, *Software Engineering*, ser. International Computer Science Series. Pearson, 2011. [Online]. Available: http://books.google.com.br/books?id=l0egcQAACAAJ