A Systematic Review of Bad Smells Metrics

Luiz Paulo Coelho Ferreira

Motivation

One of the main goals in Software Engineering is to **transform software development** in a process **predictable** and **controlled**.

In this context studying **Bad Smells** and **metrics** to determine them is a interesting way to achieve this goal.

Bad Smells

It is any symptom in the source code of a program that possibly indicates a problem.

Determining what is and what is not a bad smell is subjective. That is why studying better ways to determine it is so important.

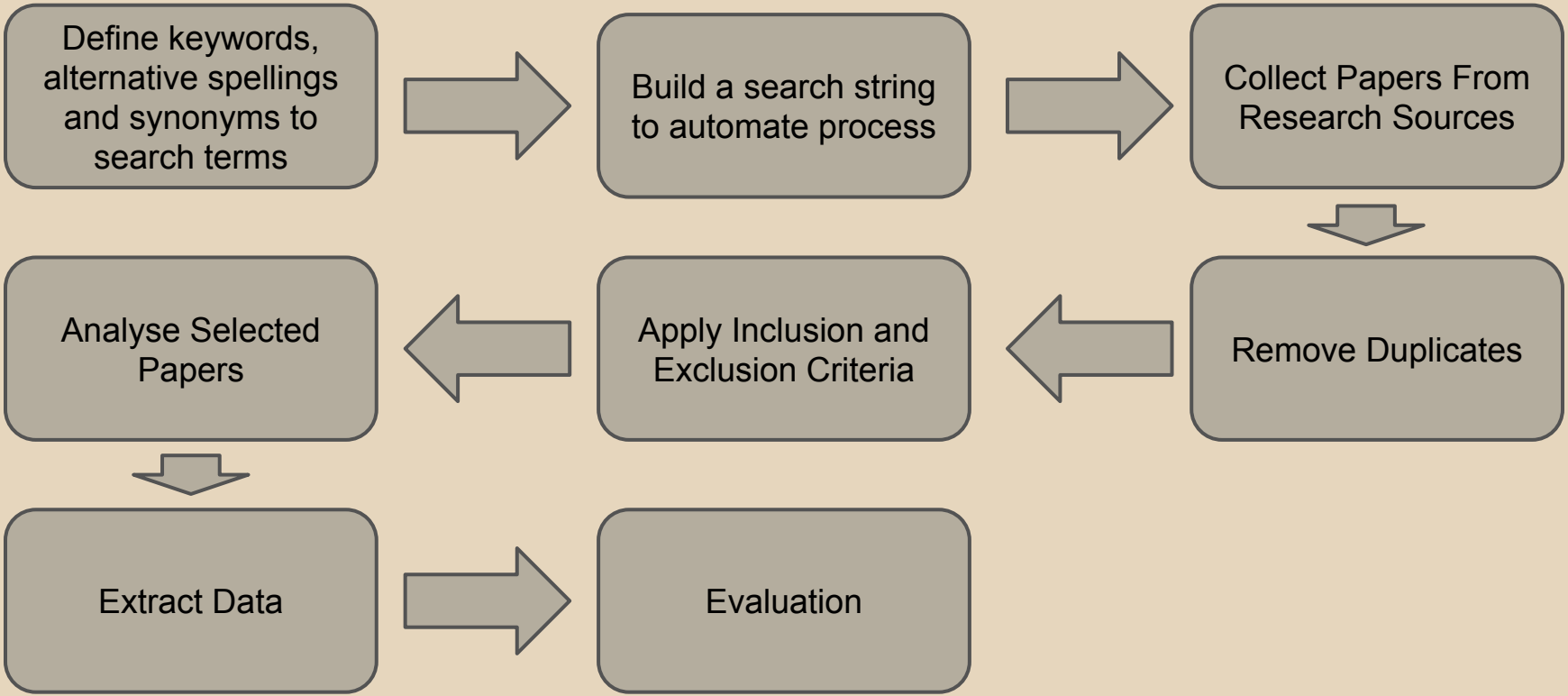
Research Questions

Q1: Which Bad Smell have well-defined metrics and which have not?

Q2: Do the existing metrics works for any object-oriented language?

Q3: What tools exist for collecting software metrics?

Search Strategies



Search String

"**Software Engineering**" AND ("**metric**" OR "metrics" OR "measure" OR "measurement" OR "measuring" OR "**quality**" OR "internal quality") AND ("**bad smell**" OR "bad smells" OR "code smell" OR "code smells" OR "anomaly code" OR "anomalies code”)

Research Sources

Research Sources	# of Papers	# of Duplicated Papers
DBLP	1	0
Science Direct	29	0
Scopus	28	10
ACM Digital Library	0	0
IEE Explorer	55	15

Inclusion and Exclusion Criteria

Inclusion:

- Papers must be from computing journals or conferences
- Papers must be related to Software Engineering
- Papers must be related to software quality metrics and bad smells
- Papers must be available to download

Exclusion:

- Papers published in conferences with Qualis smaller or equal to B2
- Papers published in journals or conferences not related with Computer Science
- Papers unrelated to Software Engineering

After Inclusion and Exclusion criteria only **28 papers** was select.

Study Quality Assessment

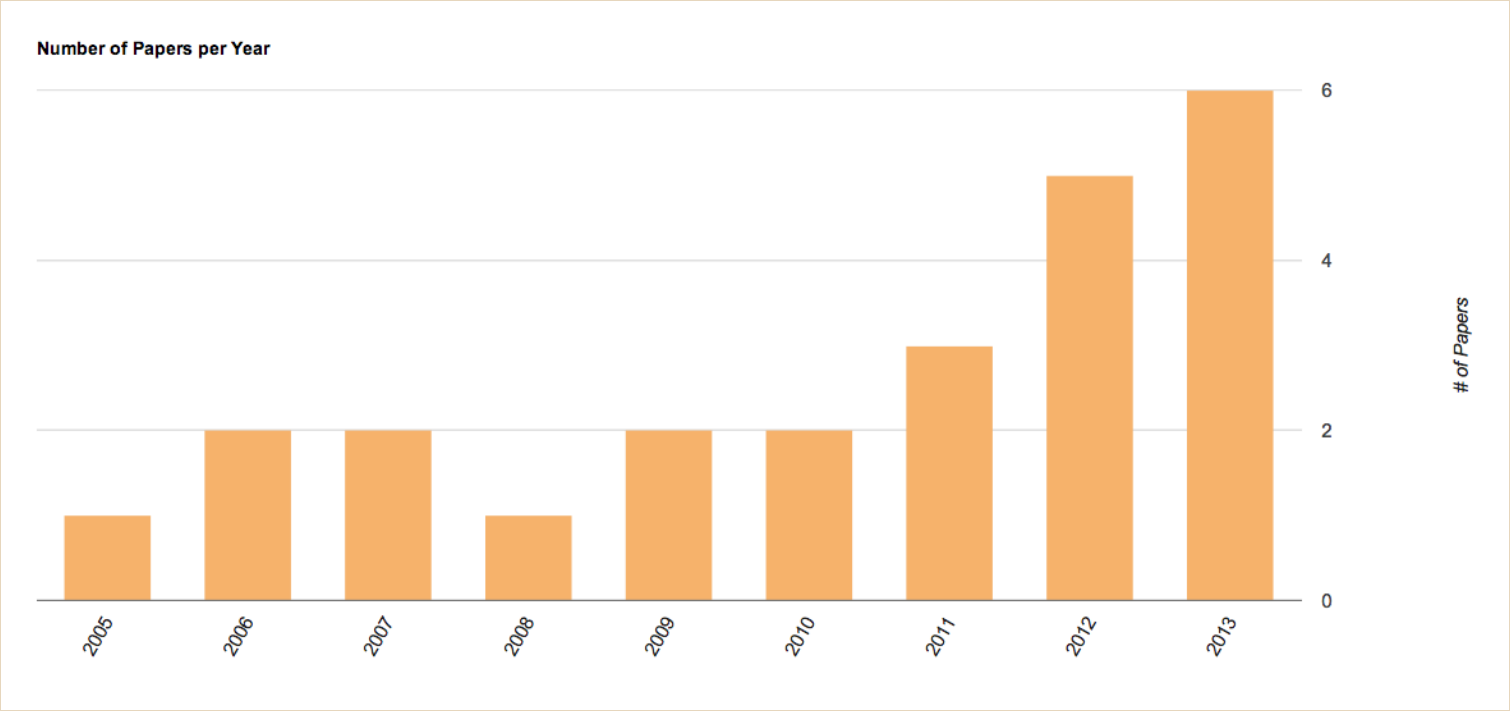
- Are the research question(s) clearly stated for the studies?
- Does the study build upon existing body of knowledge, i.e., does it explicitly discuss its contribution in the light of previous work?
- Are the metrics used in the study clearly defined?
- Are the metrics used in the study the most relevant ones for answering the research questions?
- Are the data collection methods adequately described?
- Do the researchers discuss any problems with the validity/reliability of their results?
- Is the study replicable?
- Are the findings credible?
- Are the links between the data, interpretation, and conclusions clear?
- Is the reporting clear and coherent?

Data Extraction

In each paper we will extract:

- Title;
- Author;
- Year;
- Journal;
- Language used on experiments;
- Bad Smells discussed in the paper;
- Metrics for each Bad Smell;
- Metrics tools discussed in the paper;
- Quality assessment.

Results - Years



Results - Quality Assessment 1/2

Paper	Paper Quality
Product Metrics for Automatic Identification of "Bad Smell" Design Problems in Java Source-Code	7
Search-Based Determination of Refactorings for Improving the Class Structure of Object-Oriented Systems Categories and Subject Descriptors	9.5
An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution	10
An empirical study on students' ability to comprehend design patterns	8.5
DECOR: A Method for the Specification and Detection of Code and Design Smells	9
Identification of refactoring opportunities introducing polymorphism	10
Identifying Extract Class refactoring opportunities using structural and semantic cohesion measures	10
Automated refactoring to the Strategy design pattern	10
Identification and application of Extract Class refactorings in object-oriented systems	10
Identifying thresholds for object-oriented software metrics	10
Schedule of Bad Smell Detection and Resolution: A New Way to Save Effort	9.5
Supporting extract class refactoring in Eclipse: The ARIES project	7.5
A study of cyclic dependencies on defect profile of software components	10
Code smells as system-level indicators of maintainability: An empirical study	10
Monitor-Based Instant Software Refactoring	10
To what extent can maintenance problems be predicted by code smell detection? – An empirical study	10

Results - Quality Assessment 2/2

Paper	Paper Quality
An Investigation of Bad Smells in Object-Oriented Design	6
Bad-Smell Metrics for Aspect-Oriented Software	9.5
The evolution and impact of code smells: A case study of two open source systems	9.5
An Exploratory Study of the Impact of Code Smells on Software Change-proneness	10
Bad-smell prediction from software design model using machine learning techniques	9
An approach for source code classification to enhance maintainability	6
Questioning software maintenance metrics: A comparative case study	5
A novel approach to effective detection and analysis of code clones	5
Agent based tool for topologically sorting badsmells and refactoring by analyzing complexities in source code	5

Results - Q1

Most of papers studies the same group of Bad Smells, and the more recent studies uses tools to detect bad smell, some of them does not explain what metric is been used.

Bad Smell	%	Metrics
God Class/Blob/Large Class	64.00%	WMC, AOFD, ATFD, RFC, ICP, TCC, ICH, NOM, SSM, CDM, CSM, C3, LSI, LCOM, LCOM2, LCOM3, LCOM4, LCOM5, NFD, NOM, CBO, MPC, NOP, Entity Placement, Connectivity, LOC, McCabe Cyclomatic Complexity, Other
Data Class	24.00%	NAM, WOC, NOPA
God Method/Long Method	36.00%	NLOC, LOC, NOLV, NOP, NILI, CC, ILCC, MNOB, NOP, McCabe Cyclomatic Complexity
Feature Envy	36.00%	NIC, ALD, AID
Duplicated Code	24.00%	Minimal Duplicate Chunk
Refused Bequest	16.00%	AIUR, DIT
Shotgun Surgery	20.00%	CM, ChC, CC
Long Parameter List	20.00%	NOP
Switch Statement	12.00%	
Lazy Class	12.00%	DIT, NFD, NOM, WMC, LOC, CBO
30 Others with less than	8.00%	

Results - Q2

Commonly papers explicit says that their approach works on every object-oriented language, but their experient are based in Java projects.

Do we need different approaches for dynamic-typed languages?

Languages	# of Experiments
Java	20
C#	3
C++	1
AspectJ	1
None	3

Results - Q3

While analysing the papers selected we found **30 tools** to identify bad smells and control metrics in a software.

We found the tools: PMD, Borland Toguether, PRODEOOS, DECOR, Aspect, LCLINT, JLINT, Extended Static Checker, SmallLint, FindBugs, Saber, Analyst4J, CheckStyle, FxCop, Hammurapi, SemmleCode, Crocopat, Blast, Mops, POM framework, SORMASA, JDeodorant, Java Source Metric, InCode, InsRefactor, CodeBizard, Recorder API, JBuilder, Duploc and JSmell

Threats to validity

- Quality of our search string can invalidate the results;
- Papers who we didn't had access could give us a different perception;
- Qualis A1, A2 and B1, may remove some relevant work.

Conclusion

We found 115 papers to be analysed in this work and we selected 28 to investigate deeper, thouse 25 showed to be relevant for identifying the state-of-the-art about Metrics to automatic identifying of Bad Smells.

There is a lot of studying about metrics for God Class, but others Bad Smells don't receive the same attention.

In the last years, studies started to use more popular tools to collect metric and identify Bad Smells, this can be explained by the stability of such tools.