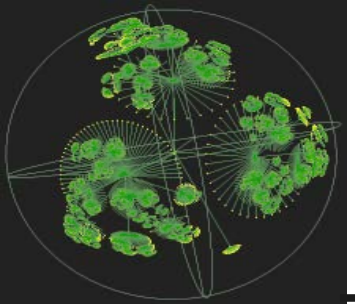


ESSeRE

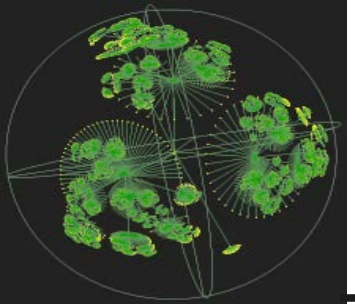
Evolution of Software Systems and Reverse Engineering

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Investigating the Impact of Code Smells Debt on Quality Code Evaluation

Francesca Arcelli Fontana, Vincenzo Ferme, Stefano Spinelli
University of Milano Bicocca, Italy

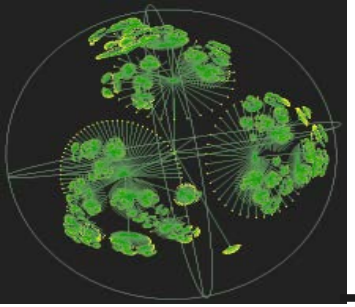
Managing Technical Debt Workshop
ICSE 2012, Zurich
June 5, 2012



Research Focus

UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

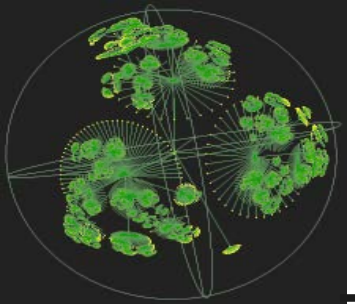
- Design Defects: **Code Smells**,.....
  - symptoms of possible problems.
  - may have a negative impact on software evolution.
  - source of **design debt** that have to be managed in some way.
- eliminate codes smell debt through **refactoring**
- refactoring could sometimes be **too expensive**
- **identify critical smells** whose refactoring cost will be repaid in terms of **ease of maintenance and improvement of system quality**
- consider **code smells and metrics values** relationships



Two Questions

1. What is the impact of removing a particular kind of smell on different software quality metrics? And hence which smells are more critical and should be removed first? In other words, is there a smell which could represent an indicator of design debt more than other smells?

2. Is it possible that a smell detected in a system could not be considered a smell in a system of another domain? Are there some kinds of of domain-dependent smells?

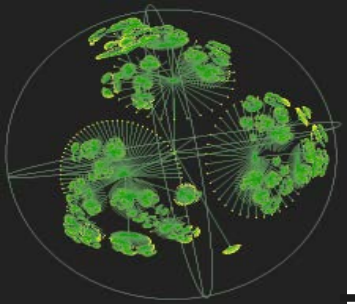


Code Smells

Data Class – classes that have fields, getting and setting methods for the fields, and nothing else. Such classes are dumb data holders and are almost certainly being manipulated in far too much detail by other classes.

God Class – performs too much work on its own, delegating only minor details to a set of trivial classes and using the data from other classes. This has a negative impact on the reusability and the understandability of this part of the system.

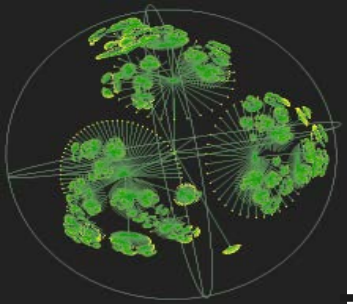
Duplicate Code – is the most pervasive and pungent smell in software. It tends to be either explicit or subtle. Explicit duplication exists in identical code, while subtle duplication exists in structures or processing steps that are outwardly different, yet essentially the same.



Systems and Domains

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Systems	Application Domains	Number of Class / LOC
Columba 1.0	Application Software	1303/71680
Drawswf 1.2.9	Application Software	302/27008
Galleon 2.3.0	Application Software	556/52653
C_jdbc 2.0.2	Client-Server Software	778/81306
Heritrix 1.8.0	Client-Server Software	649/39272
Struts 2.2.1	Client-Server Software	1608/74670
Ganttproject 2.0.9	Diagram generator/Data visualization	801/47051
Jhotdraw 7.5.1	Diagram generator/Data visualization	749/75958
Velocity 1.6.4	Diagram generator/Data visualization	429/26854
Antlr 3.2	Software Development	330/25243
Drjava 20100913-r5387	Software Development	920/62380
Pmd 4.2.5	Software Development	885/60875

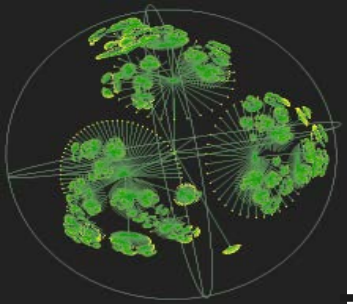


Metrics and Tools

Metrics	Granularity	Tools
Abstracteness (Abstr)	System	Google CodePro Analytics
Distance from Main Sequence (DMS)	System	
CC – Changing Classes	Method	iPlasma
FANOUT – Number of Called Classes	System	
Average Line of Code per Method (ALCM)	System	Google CodePro Analytics
CYCLO – McCabe's Cyclomatic Number	Method	iPlasma
WMC – Weighted Method Count	Class	
AMW – Average Method Weight	Class	
ATFD – Access to Foreign Data	Class, Method	
LAA – Locality of Attribute Accesses	Method	
TCC – Tight Class Cohesion	Class	
LCOM – Lack of Cohesion in Method	Method	Eclipse Metrics

Metrics Classification:

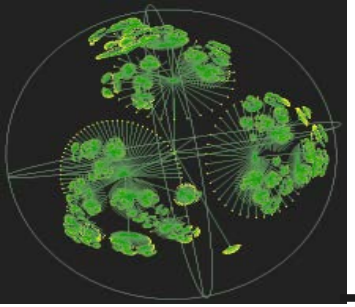
- ✓Coupling
- ✓Size
- ✓Complexity
- ✓Data Abstraction
- ✓Cohesion



Experimental Plan

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

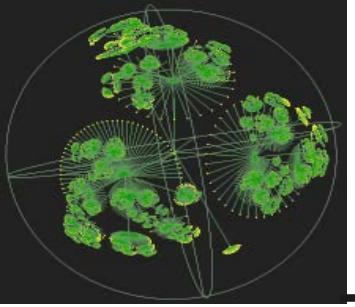
1. Compute the metric values on the systems
2. Detect the Code Smells in the systems
3. Apply refactoring techniques
4. Re-compute metric values
5. Evaluate the differences
6. Verify the removal of Code Smells and the introduction of new ones



Code Smells Detection

UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

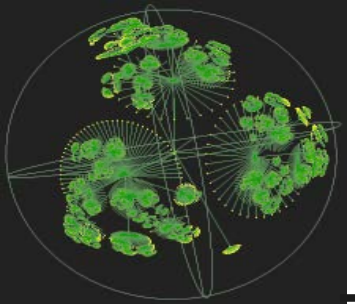
Systems	God Class	Data Class	Duplicate Code (LOC)
Columb	14	42	4209
Drawswf	5	35	1376
Galleon	30	41	11556
C_jdbc	30	47	6972
Heritrix	33	18	1529
Struts	36	176	6192
Ganttproject	22	56	1064
Jhotdraw	17	14	9171
Velocity	3	18	1550
Antlr	27	28	3243
Drjava	22	25	5240
Pmd	17	26	2924



Refactoring Steps

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

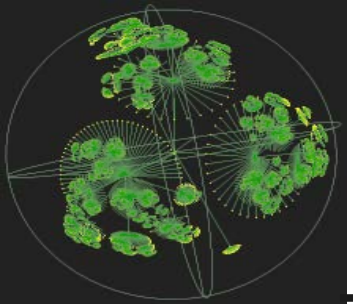
Order	God Class	Data Class	Duplicate Code
1	Extract Class	Encapsulate Field	Extract Method
2	Extract Subclass	Remove Setting Method	-
3	-	Hide Method	-



Metrics Evaluation: Data Class

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Systems	ALCM	CYCLO	WMC	AMW	Abstr	DMS	CC	FANOUT	ATFD	LAA	TCC	LCOM
Columb	= 0%	= 0%	+ 0,85%	- 0,36%	= 0%	= 0%	= 0%	= 0%	= 0%	- 0,07%	+ 1,04%	= 0%
Drawswf	- 1,16%	- 1,10%	+ 1,17%	- 2,65%	= 0%	= 0%	= 0%	= 0%	= 0%	- 0,02%	+ 3,43%	+ 5,59%
Galleon	- 0,45%	- 0,41%	+ 0,27%	- 0,05%	= 0%	= 0%	= 0%	= 0%	= 0%	- 0,08%	- 0,01%	+ 0,50%
C_jdbc	- 1,47%	- 0,93%	+ 0,87%	- 0,35%	= 0%	= 0%	= 0%	= 0%	- 0,34%	- 0,09%	+ 0,76%	+ 1,19%
Heritrix	- 0,42%	- 0,51%	+ 0,26%	- 0,38%	= 0%	= 0%	= 0%	= 0%	= 0%	- 0,04%	+ 0,48%	+ 1,33%
Struts	- 0,25%	= 0%	- 0,01%	- 0,04%	= 0%	= 0%	- 0,67%	- 0,83%	+ 5,00%	- 0,13%	+ 9,68%	+ 0,77%
Ganttproject	- 0,40%	- 0,65%	+ 0,41%	- 0,09%	= 0%	= 0%	= 0%	= 0%	= 0%	- 0,04%	+ 0,63%	+ 0,87%
Jhotdraw	- 0,41%	- 0,52%	+ 0,30%	- 0,05%	= 0%	= 0%	+ 0,08%	= 0%	= 0%	- 0,01%	+ 0,44%	+ 6,10%
Velocity	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%
Antlr	- 0,66%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	+ 4,80%
Drjava	- 0,12%	= 0%	+ 0,18%	- 0,10%	= 0%	= 0%	= 0%	= 0%	+ 4,44%	- 0,02%	+ 0,34%	+ 0,96%
Pmd	- 0,84%	- 0,88%	+ 0,62%	- 0,33%	= 0%	= 0%	= 0%	= 0%	= 0%	- 0,10%	- 0,14%	+ 1,53%

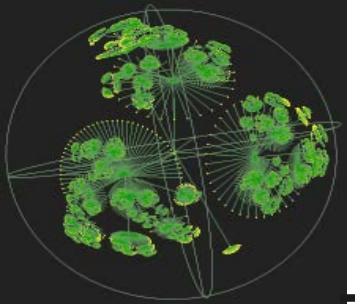


Metrics Evaluation: Data Class

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Systems	ALCM	CYCLO	WMC	AMW	LAA	TCC	LCOM
Columb	= 0%	= 0%	+ 0,85%	- 0,36%	- 0,07%	+ 1,04%	= 0%
Drawswf	- 1,16%	- 1,10%	+ 1,17%	- 2,65%	- 0,02%	+ 3,43%	+ 5,59%
Galleon	- 0,45%	- 0,41%	+ 0,27%	- 0,05%	- 0,08%	- 0,01%	+ 0,50%
C_jdbc	- 1,47%	- 0,93%	+ 0,87%	- 0,35%	- 0,09%	+ 0,76%	+ 1,19%
Heritrix	- 0,42%	- 0,51%	+ 0,26%	- 0,38%	- 0,04%	+ 0,48%	+ 1,33%
Struts	- 0,25%	= 0%	- 0,01%	- 0,04%	- 0,13%	+ 9,68%	+ 0,77%
Ganttproject	- 0,40%	- 0,65%	+ 0,41%	- 0,09%	- 0,04%	+ 0,63%	+ 0,87%
Jhotdraw	- 0,41%	- 0,52%	+ 0,30%	- 0,05%	- 0,01%	+ 0,44%	+ 6,10%
Velocity	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%
Antlr	- 0,66%	= 0%	= 0%	= 0%	= 0%	= 0%	+ 4,80%
Drjava	- 0,12%	= 0%	+ 0,18%	- 0,10%	- 0,02%	+ 0,34%	+ 0,96%
Pmd	- 0,84%	- 0,88%	+ 0,62%	- 0,33%	- 0,10%	- 0,14%	+ 1,53%

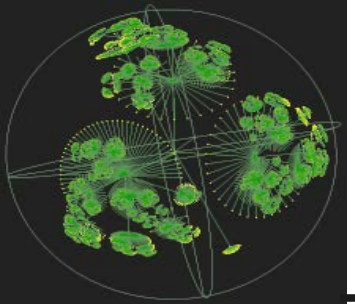
- ✓ Improvement of systems cohesion (TCC and LCOM)
- ✓ Improvement of WMC due to Encapsulate Field refactoring that has often led to the introduction of methods of low complexity
- ✓ Decrement of ALCM due to the Encapsulate Field, which involves the introduction of new methods with few lines of code



Metrics Evaluation: God Class

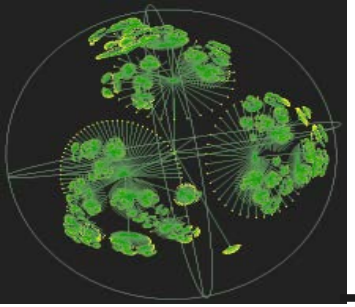
UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Systems	ALCM	CYCLO	WMC	AMW	Abstr	DMS	CC	FANOUT	ATFD	LAA	TCC	LCOM
Columb	+ 0,60%	= 0%	+ 0,85%	- 0,36%	= 0%	= 0%	= 0%	= 0%	= 0%	- 0,07%	+ 1,04%	+ 0,65%
Drawswf	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%
Galleon	- 0,22%	- 0,41%	+ 0,49%	- 0,15%	= 0%	= 0%	+ 0,18%	+ 0,97%	+ 1,28%	- 0,19%	- 3,03%	+ 0,53%
C_jdbc	- 0,81%	- 0,46%	+ 1,26%	- 0,70%	= 0%	= 0%	- 0,53%	- 0,21%	- 0,45%	- 0,08%	+ 1,45%	+ 1,18%
Heritrix	= 0%	- 0,51%	+ 0,62%	- 2,07%	= 0%	= 0%	- 0,14%	- 0,17%	- 0,36%	- 0,06%	+ 0,04%	+ 1,29%
Struts	= 0%	= 0%	- 0,01%	- 0,04%	= 0%	= 0%	- 0,67%	- 0,83%	+ 5,00%	- 0,13%	+ 9,68%	+ 1,15%
Ganttproject	+ 0,40%	= 0%	+ 0,45%	- 0,23%	= 0%	= 0%	- 0,46%	- 0,58%	- 1,33%	= 0%	+ 1,07%	+ 0,70%
Jhotdraw	= 0%	- 0,52%	+ 0,41%	- 0,05%	= 0%	= 0%	- 0,02%	+ 2,39%	+ 7,28%	+ 0,26%	- 10,62%	+ 5,86%
Velocity	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%	= 0%
Antlr	= 0%	+ 1,06%	- 6,47%	- 3,72%	= 0%	= 0%	- 4,25%	- 5,35%	- 4,11%	- 3,05%	+ 1,53%	+ 4,68%
Drjava	+ 0,71%	= 0%	+ 0,23%	- 0,37%	- 1,41%	= 0%	- 0,59%	- 1,41%	- 6,49%	- 0,01%	+ 0,61%	+ 0,88%
Pmd	- 0,36%	- 0,88%	+ 0,74%	- 0,94%	= 0%	= 0%	- 1,10%	- 1,16%	- 1,64%	- 0,09%	- 0,08%	+ 1,51%



Metrics Evaluation: God Class

- ✓ Improvement of systems cohesion (TCC and LCOM) independently from the application domain of the analyzed systems
- ✓ Decrement of AMW due to the introduction of new classes and the movement of some methods and attributes of the refactored classes in these new classes
- ✓ When we apply the refactoring steps for the God Class, it would be useful to immediately know the impact on some metrics values: the wrong choice could lead to the deterioration of other metrics values, as those related to coupling.

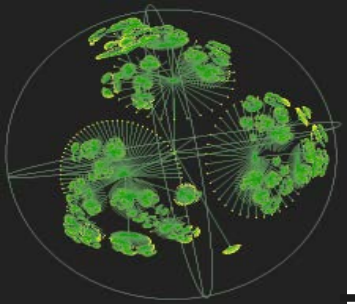


Metrics Evaluation: Duplicate Code

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Systems	ALCM	CYCLO	WMC	AMW	Abstr	DMS	CC	FANOUT	ATFD	LAA	TCC	LCOM
Columb	= 0%	= 0%	+ 0,5%	- 0,2%	= 0%	= 0%	- 0,6%	- 3,4%	= 0%	- 0,4%	+ 3,3%	- 0,9%
Drawswf	= 0%	+ 1,5%	+ 0,6%	= 0%	= 0%	= 0%	- 1,3%	- 0,3%	= 0%	= 0%	+ 1,5%	- 1,6%
Galleon	= 0%	+ 1,2%	+ 1,2%	= 0%	= 0%	= 0%	- 0,3%	- 2,7%	= 0%	= 0%	+ 2,2%	- 2,1%

- ✓ Deterioration of CC and FANOUT because often to remove the duplicate code between two different classes we have to create a method in a class and invoke it from the other one
- ✓ LCOM values decrease in all systems since respect to the initial state of the system, there are more methods that access the same attributes of the refactored class
- ✓ TCC metric improve in all the systems, indicating an improvement of the overall cohesion
- ✓ WMC and CYCLO value improve because the Extract Method improves the code complexity decreasing the number of linearly indipendent paths

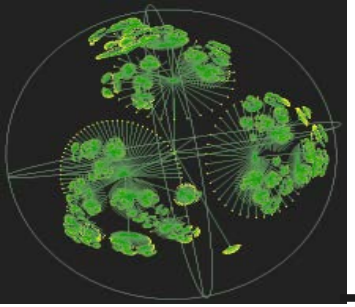


Domain dependent Smells

UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

DATA CLASS

- ✓ Data Structures used by Parser;
- ✓ Application State Classes;
- ✓ Test Classes;
- ✓ Use of Libraries (AWT, SWING);
- ✓ Java Bean;
- ✓ Data Structures used by Logging, Debugging Classes.

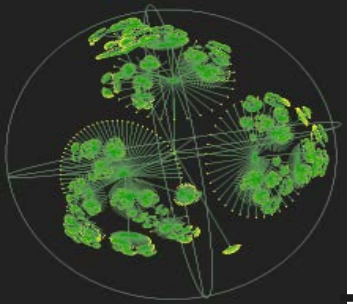


Domain dependent Smells

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

GOD CLASS

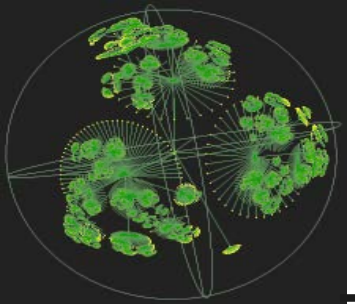
- ✓ Parser;
- ✓ Crawler;
- ✓ Visitors;
- ✓ Interpreter;
- ✓ Writer Classes;
- ✓ Profiler Classes;
- ✓ Errors Handler;
- ✓ Test Classes;
- ✓ Use of Libraries (AWT, SWING);
- ✓ Debugging Classes;
- ✓ Classes that Manage Database Interactions.



Conclusions

- We have identified **smells** related to the improvement/deterioraton of some **metrics**.

- We can **manage the *smell debt*** by condidering:
  - The **most critical smells**
  - If our interest is to mantain or **improve a particular metric value**
    - Remove the smell that allows to improve this metric:
      - i.e. improve cohesion (LCOM, TCC): remove God and Data Class
  - Remove the smells whose **refactoring is easier**:
    - It was easier to remove Data Class respect to God Class
    - Removing Duplicate Code it depends on the required refactoring steps
- We observed that some smells detected by the tools are **not real smells**



Future Developments

- Extend our experiment with other **smells and systems**
- Refine the set of the considered **metrics**
- Check the metric value when **different refactoring** can be applied
- Provide a kind **of prioritization of the smells** to be removed
- Study the **effect of refactoring** on the average time to fix a defect, reduction in bugs,....
- Choice of **the *best* tools** for Metrics, CS Detection and Refactoring
- Suggestions to improve CS Detection tools:
  - **Domain-dependent smells** and **design-dependent smells** (*smell filter*)
  - **Context-based thersholds setting**
  - **Correlations** among smells, smells and anti-patterns (by removing a smell, I could remove/reduce another kind of smell)
  - Automated refactoring