From code similarity detection to model driven similarity detection: first milestones

Jarod BLAIN[1], Corentin GUILLEVIC[1], Alex MOULIN[1], Ali BENJILANY[2]

[1]Université de Nantes, CNRS, LS2N, F-44000, Nantes, France

[2]Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes, Rabat, Maroc

[1] Etudiants en M1 ALMA- [2] Elève ingénieur

Empirical software engineering

Model Driven Software Engineering

Objectives

- Detect similarities in software applications:
  - source code level
  - model level
- Identify the best tool.
- Produce similarity metrics.
- Provide guidelines

Scope

- Object-oriented code similarity is the similarity between syntax, function output, code structure, architecture.
- Comparing code is useful in a lot of situations (see Use Cases) and used in education, engineering, research, etc
- Structure analysis is very effective but difficult due to its specific aspect.
- The state of the art shows that every software has a flaw when it comes to overall comparison (not based on one use case).
- The ultimate goal is to have one tool that can do it all (a do-it-all tool) achieved by building a collaborative toolbox, picking the best from existing soft… compare code.
- Going from source code to models.

Insights

- Clones and duplicates increase software tests and maintenance costs.
- Factoring out duplicates and inconsistencies management needs duplicates detection that can lean on similarity calculation.
- Experimentations done on the code of the Benchmarks according to Use Cases.
- 15 tools identified and 7 tested.
- State of the art: 23 papers studied and spread out into 5 categories: tool comparison, tools, metrics, methods & approaches, and models.
  - "*Plagiarism detection must be integrated into the toolset and activities of MDE instructors.*" [1]
  - "*When source code is copied and modified, which code similarity detection techniques or tools get the most accurate results?*" [3]

Milestone 1 - From code similarity detection...

Tool Comparison

- Different existing tools compared [4].
- The best (overall) tool is JPlag according to its results and ease of use.

"A metric is an information that can be extracted from code resulting in a number (e.g. number of lines)".

Metrics Study

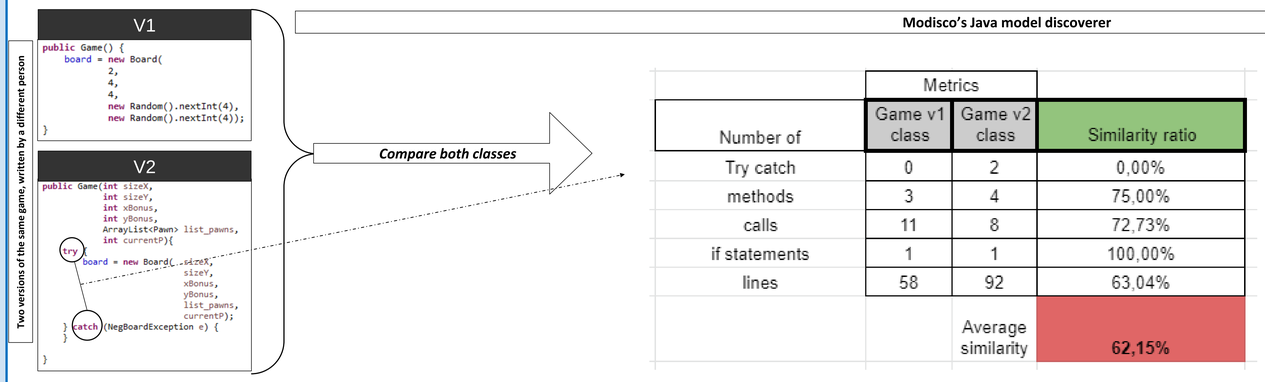
- Similarity ratio = f(software engineering metrics, similarity metrics).
- Metrics help also for abstraction means.
- We experimented a simplified version to validate our computation formula.

%	v1	v2	v3	v4	v5	final
v1		73.93	80.64	19.71	24.31	28.10
v2			89.28	24.32	21.22	19.06
v3				15.48	19.21	22.54
v4					80.38	74.47
v5						92.92

%	v1	v2	v3	v4	v5	final
v1		86.14	87.34	54.64	51.08	48.10
v2			78.57	43.78	41.79	40.12
v3				60.80	55.91	52.02
v4					87.03	78.75
v5						90.10

%	v1	v2	v3	v4	v5	final
v1		83.55	94.18	42.38	39.13	34.92
v2			79.64	28.52	26.35	23.53
v3				47.80	44.13	39.37
v4					89.51	80.40
v5						89.85

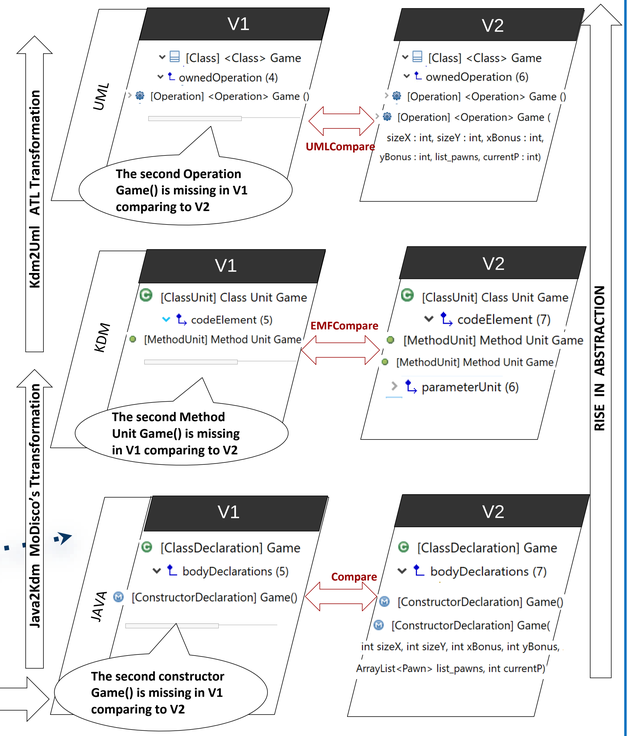
A SimpleGame Example (code similarity)



	Metrics		
Number of	Game v1 class	Game v2 class	Similarity ratio
Try catch	0	2	0,00%
methods	3	4	75,00%
calls	11	8	72,73%
if statements	1	1	100,00%
lines	58	92	63,04%
		Average similarity	62,15%

Plagiarism detection requires code or model mutation to introduce the malicious copy strategy (future work).

Milestone 2 - … to model similarity detection



- Same results, different entities : (Java → Constructor - KDM model → MethodUnit - Operation<--> UML).
- Model similarity detection reveals changes made on the architecture of a program regardless of code details thanks to the rise in abstraction, while the code similarity detection returns a global rate of similarity.
- *Abstraction requires fine heuristics to obtain high-level concepts* [2].

→ Meanwhile, the upcoming milestones of the research might come up with a proof of the efficiency of a certain model-based comparison compared to the others. Much remains to do, especially in reverse engineering and comparison implementation.

References

[1] Salvador Pérez, Manuel Wimmer, and Jordi Cabot. "Efficient plagiarism detection for software modeling assignments". In:Computer Science Education30 (Jan. 2020), pp. 1–29.doi:10.1080/08993408.2020.1711495

[2] Pascal André et al. "JavaCompExt: Extracting Architectural Elements from Java Source Code".In:WCRE. Lille, France: IEEE, Oct. 2009, pp. 317–318.doi:10.1109/WCRE.2009.53.

[3] Chaiyong Ragkhitwetsagul, Jens Krinke, and David Clark. "A comparison of code similarity analysers". In:Empirical Software Engineering23.4 (2018), pp. 2464–2519.

[4] Master TER Report 2021 https://uncloud.univ-nantes.fr/index.php/s/k2Ezp4bJ8bki3Sn

Use Cases

Plagiarism
Licensing
Clone detection
Code refactoring & Code reuse
Pattern detection
Software defects,
Malware insertion

Best tools
JPLAG
MOSS
…

According to our experimentations, JPLAG and MOSS are very good in their comparison results, they both give accurate detailed and easy to understand results. JPlag is used in this poster for it's easiness and simplicity.

Benchmarks

Simple programs
Student applications
Lego EV3+android applications
Git repositories
Plagiarism benchmarks

Further information

Supervisors: ANDRE Pascal, BRUNELIERE Hugo, TAMZALIT Dalila

Students report: https://uncloud.univ-nantes.fr/index.php/s/k2Ezp4bJ8bki3Sn

https://www.ls2n.fr/equipe/aelos/

https://www.ls2n.fr/equipe/naomod/

Poster credit - https://colinpurrington.com/tips/poster-design/