The Use of Large Language Models for Software Refactoring

Henrique Nunes

Laboratory of Software Engineering (Labsoft) - Computer Science Department (UFMG)

August 9th, 2024

Outline

SCAM submitted paper: context

Maintainability is crucial in software engineering, affecting cost, quality, and development.

LLMs show promise in coding, but real-world projects challenge their effectiveness.

The lack of consensus on maintainability tools highlights the potential of LLMs to automate fixes.

SCAM submitted paper: objective

Our study assesses LLMs' effectiveness and limitations in fixing software maintainability in real-world projects.

We use SonarQube to detect and Visual Studio Copilot to fix 127 maintainability issues in 10 GitHub Java repositories.

We conducted a human evaluation with 55 participants to evaluate the readability of 52 LLM-generated solutions.

SCAM submitted paper: research questions

- RQ1. To what extent can an LLM fix maintainability issues?
- RQ2. To what extent do developers find LLM solutions more readable?

SCAM submitted paper: study design

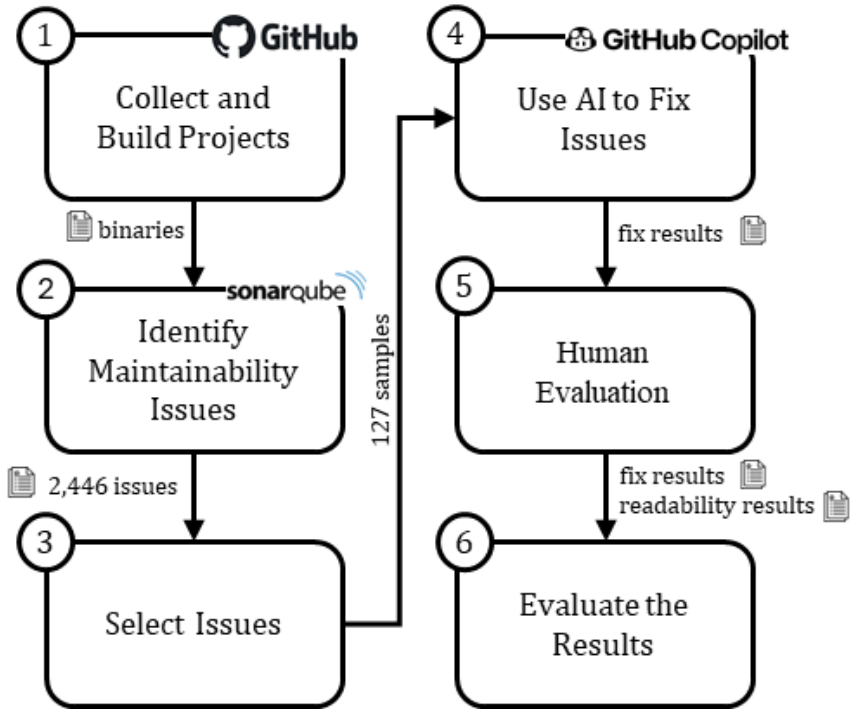

Figure 1: Steps to evaluate if LLM can automatically fix maintainability issues.

SCAM submitted paper: selected projects

Projects and Versions	*#Issues*	*Samples*	*Stars*	*Contributors*
Apollo 2.2.0	119	6	28.8k	149
Byte Buddy 1.14.13	317	11	6.1k	94
Google Java Format 1.21.0	101	11	5.4k	90
Google Jimfs 1.3.0	47	6	2.4k	26
Google Guava 33.0.0	782	16	2.4k	304
Google Guice 7.0.1	173	14	12.4k	78
Jitwatch 1.4.9	365	24	3k	32
Jsoup 1.18.1	171	19	10.7k	107
Zxing 3.5.4	264	16	32.3k	125
Webmagic 0.10.1	107	4	11.3k	54
total	2,446	127		

SCAM submitted paper: selected issues

Acronyms	*Rules*	*Issues*	*Projects Coverage*
CCM	Cognitive Complexity of methods should not be too high	26	9
GET	Generic exceptions should never be thrown	13	6
MIS	Mergeable if statements should be combined	9	5
CCO	Sections of code should not be commented out	13	6
SLD	String literals should not be duplicated	19	7
TON	Ternary operators should not be nested	8	5
TUT	Track uses of "TODO" tags	16	6
TBS	Two branches in a conditional structure should not have exactly the same implementation	5	3
UAR	Unused assignments should be removed	8	3
UMP	Unused method parameters should be removed	10	5
	total	127	

SCAM submitted paper: prompt design

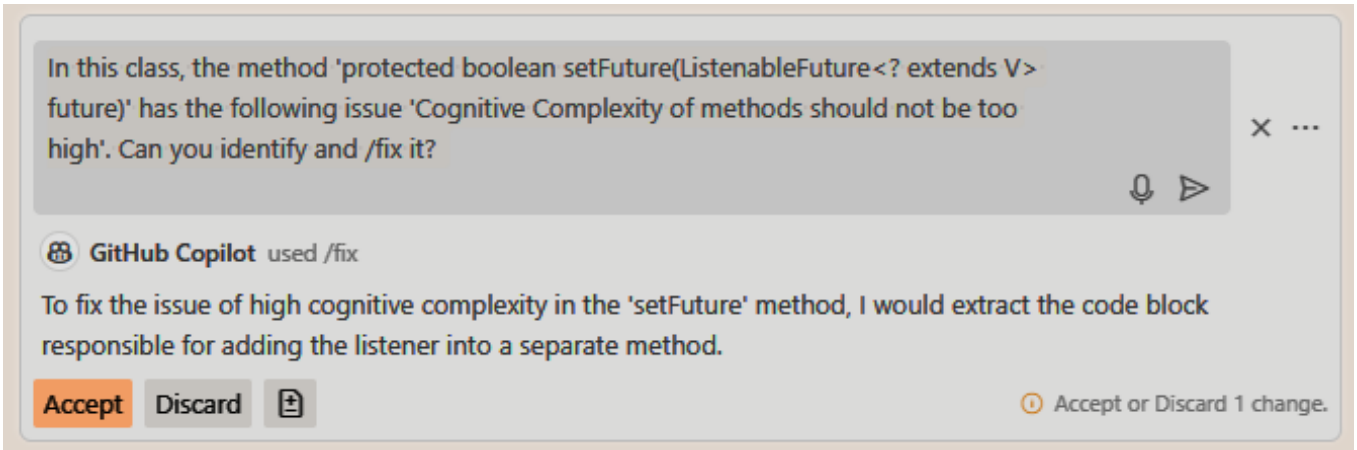

Figure 2: Copilot Chat with a prompt example and solution explanation.

SCAM submitted paper: build errors

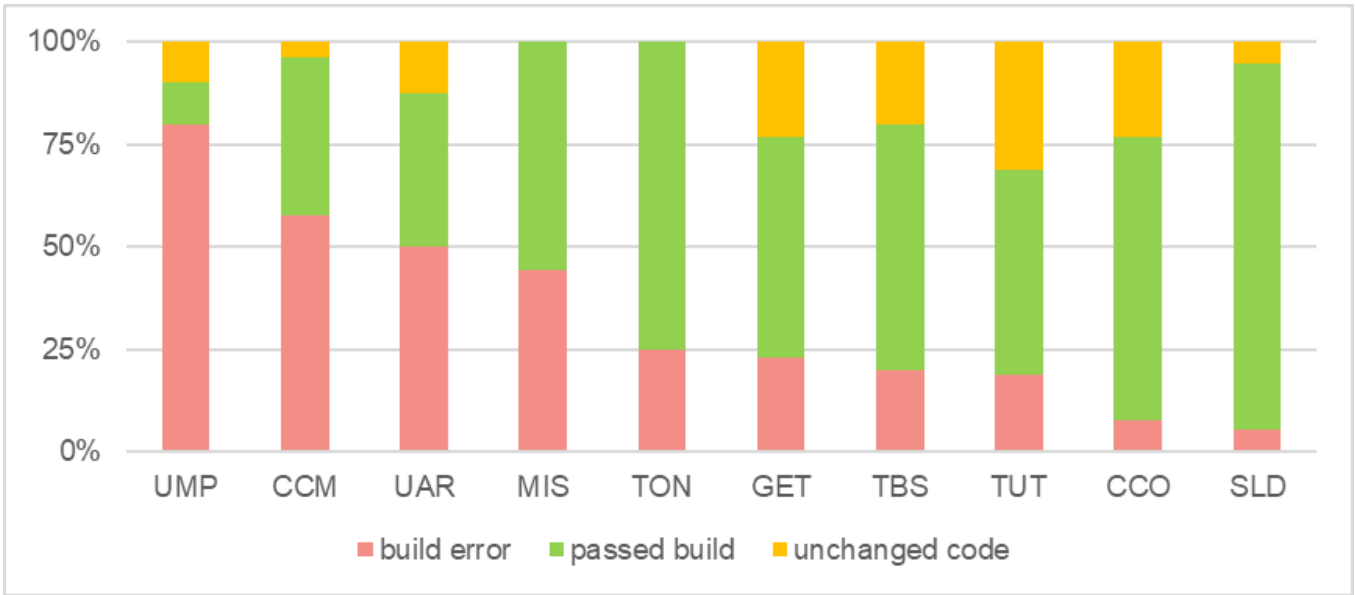

Figure 3: Build errors for the 127 rules violation.

From the 127 samples of the study, 111 methods were changed by LLM.

SCAM submitted paper: type errors

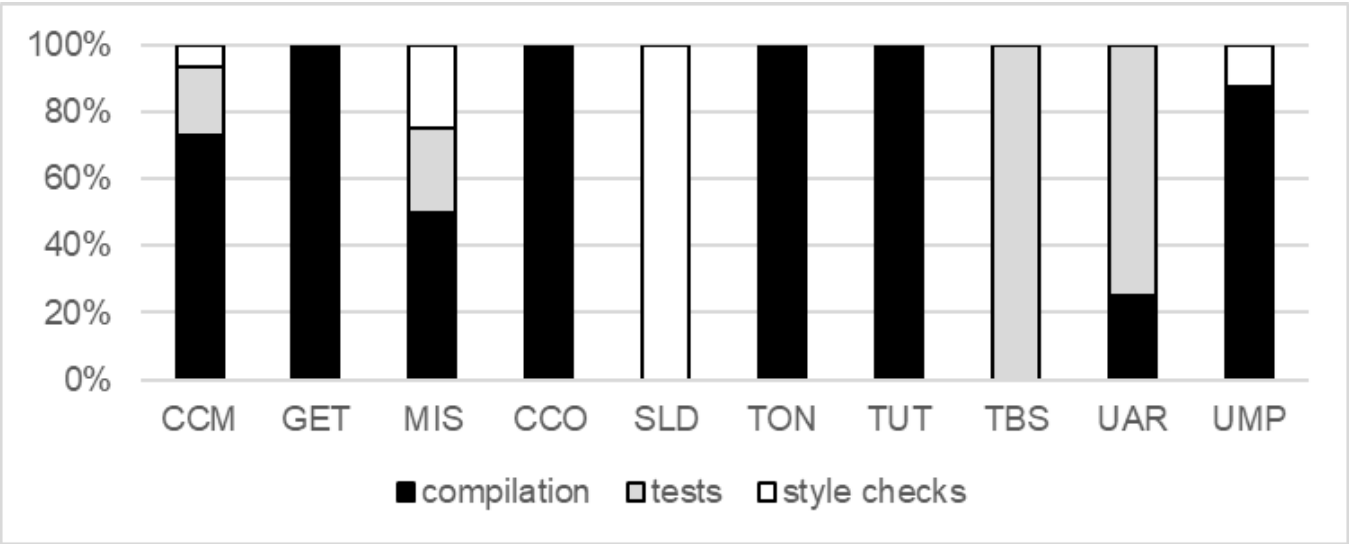

Figure 4: Types of the 42 build errors by the type of error.

SCAM submitted paper: type errors detailed

37.84% of the methods did not pass through the build process:

- 71.43% have incorrect syntax (compilation error)
- 19.05% have semantic problms (tests error)
- 9.52% have code style violation (checkstyle evaluation)

SCAM submitted paper: fixed issues

rules	*fixed*	*not fixed*	*degraded*
CCO	9	0	0
TON	6	0	0
UAR	3	0	0
MIS	4	1	0
CCM	7	2	1
TBS	2	1	0
TUT	4	2	2
GET	3	3	1
SLD	3	12	2
UMP	0	1	0
total	41	22	6

SCAM submitted paper: introduced issues

dataset id	*initial issue*	*inserted issues*
6	TUT	Track uses of TODO tags; The diamond operator ("⟨⟩") should be used.
11	SLD	Unused local variables should be removed; Unused assignments should be removed.
19	CCM	Raw types should not be used.
37	SLD	Local variables should not be declared and then immediately returned or thrown.
67	TUT	Track uses of TODO tags.
74	GET	Generic exceptions should never be thrown.

SCAM submitted paper: 8 methods that failed in tests

dataset id	*rule*	*result*
18	Unused assignments should be removed	not fixed
20	Cognitive Complexity of methods should not be too high	fixed
53	Cognitive Complexity of methods should not be too high	fixed
71	Cognitive Complexity of methods should not be too high	fixed
118	Mergeable if statements should be combined	fixed
125	Two branches in a conditional structure should not have exactly the same implementation	fixed
126	Unused assignments should be removed	not fixed
141	Unused assignments should be removed	fixed

SCAM submitted paper: RQ1. answer

Out of the 111 maintainability issues:

- 41 methods (36.94%) were fixed
- 70 (63.06%) methods that did not have the maintainability issues fixed:
  - 42 (37.84% from the 111) failed in build analysis
  - 22 (19.82%) did not fix the issues but did not have errors or introduced new issues
  - 6 (5.40%) did not have errors and introduced new maintainability issues.

SCAM submitted paper: human evaluation (votes)

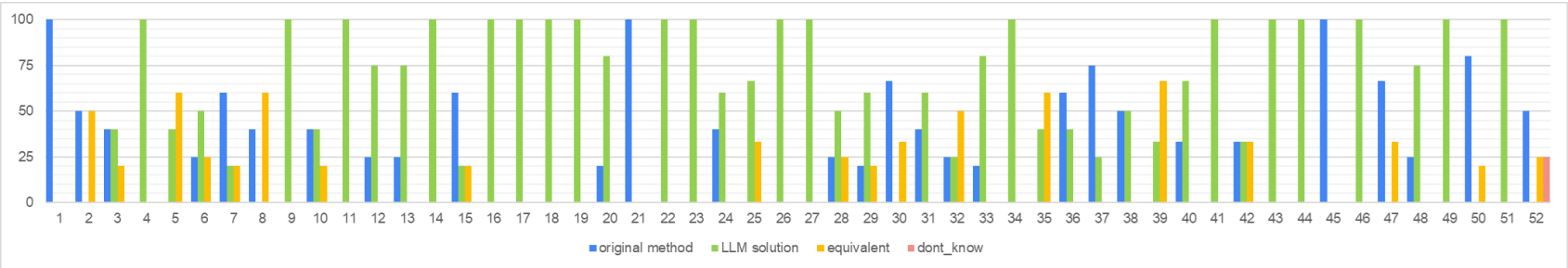

Figure 5: Proportion of answers by methods comparisons

SCAM submitted paper: RQ2. answer

Out of 52 pairs of methods evaluated by 55 developers:

- 31 (59.7%) considered the method with LLM solution more readable than the original method
- 11 (21.1%) found the original method more readable.
- 5 (9.6%) deemed both methods equivalent
- 5 (9.6%) cases, the comparison resulted in a draw

SCAM submitted paper: human evaluation (winners)

Number of times when an option wins a comparison by the rules:

rules	*original method*	*LLM solution*	*equivalent*	*draw*
CCM	0	9	0	0
GET	1	1	1	1
MIS	0	2	0	0
CCO	2	4	1	1
NDL	1	1	0	1
SLD	5	7	1	1
TON	0	2	0	0
TUT	2	3	1	1
TBS	0	1	0	0
UAR	0	1	1	0

Organizing the comparisons by types of rule violations, the LLM solution answer is the most voted for 8 (80%) rules, in the other 2 (20%) rules there was a draw.

SCAM submitted paper: Conclusions

About LLM limitation:

- LLM might **miss method scopes**, even with IDE integration.
- LLM can fix maintenance issues without breaking syntax, but may **change method behavior**.
- LLM might not fix all issues in a method, **even if it resolves the one mentioned**.
- LLM may prioritize fixing an issue over maintainability, potentially **introducing new issues**.

LLMs show potential to improve code readability and fixing maintainability issues, but their effectiveness is limited.

Current Work: read recent LLM papers

Pomian et. al. (FSE 24): A tool that ranks LLM refactoring suggestions and provides high-quality options to developers.

Choi et. al. (SSBSE 24 co-located in FSE 24): proposes an iterative project-level code refactoring to reduce the Cyclomatic Complexity.

Shirafuji et. al. (APSEC 23): The study proposes a method for selecting the best-suited code refactoring examples to be used for few-shot prompts.

Al Omar et. al. (MSR 24): The study evaluates developers refactoring prompts.

White al. (Book 2023): A book that discusses the use of prompt design.

Current Work: read recent surveys

Fan et al. (ICSE 2023) - Future of Software Engineering:

- LLMs could assist developers with the explanation and documentation of design patterns.
- Utilize few-shot learning for code refactoring.

Nyirongo et. al. (Arxiv, 2024) - Survey of Deep Learning for Refactoring:

- Evaluate different method-level refactorings.
- Use industrial datasets for better generalization.
- Assess the effectiveness of refactoring improvements.

Current Work: similarities between studies

Utilize several suggestions from LLMs.

Human intervention is required.

The few-shot technique is very common.

Most proposals focus on reducing complexity.

Most refactoring occurs at the method level.

The Extract Method and Move Method are the most explored techniques.

Current Work: problem

Context

- Developers have **intensively adopted** Large Language Models (LLMs) for different software engineering tasks.
- Most refactoring studies using LLMs are focused on using **prompt engineering techniques**, like few-shot learning, to improve complex methods.

Problems

- These studies are promising but **need many LLM suggestions** to achieve a correct program transformation from syntactic and semantic points of view.
- Furthermore, unlike other techniques, Extract Method and Move Method have been **extensively explored**.

Current Work: objectives

The objective of our study is...

- Option 1: Reduce LLM suggestions to achieve viable refactoring.
- Option 2: Explore different types of refactoring methods.
- Option 3: Evaluate human behavior using prompt techniques.

Thank you!

Henrique Gomes Nunes

henrique.mg.bh@gmail.com