

程序员意图与协作行为模拟制导的自动化程序错误修复

PATCH: Empowering Large Language Model with Programmer-Intent Guidance and Collaborative-Behavior Simulation for Automatic Bug Fixing

张俞炜, 金芝, 邢颖, 李戈, 刘芳, 朱家鑫, 窦文生, 魏峻

ACM Transactions on Software Engineering and Methodology

(Invited to be presented at the FSE 2025 Journal-First Track)

联系方式: 张俞炜 zhangyuwei@iscas.ac.cn

Background

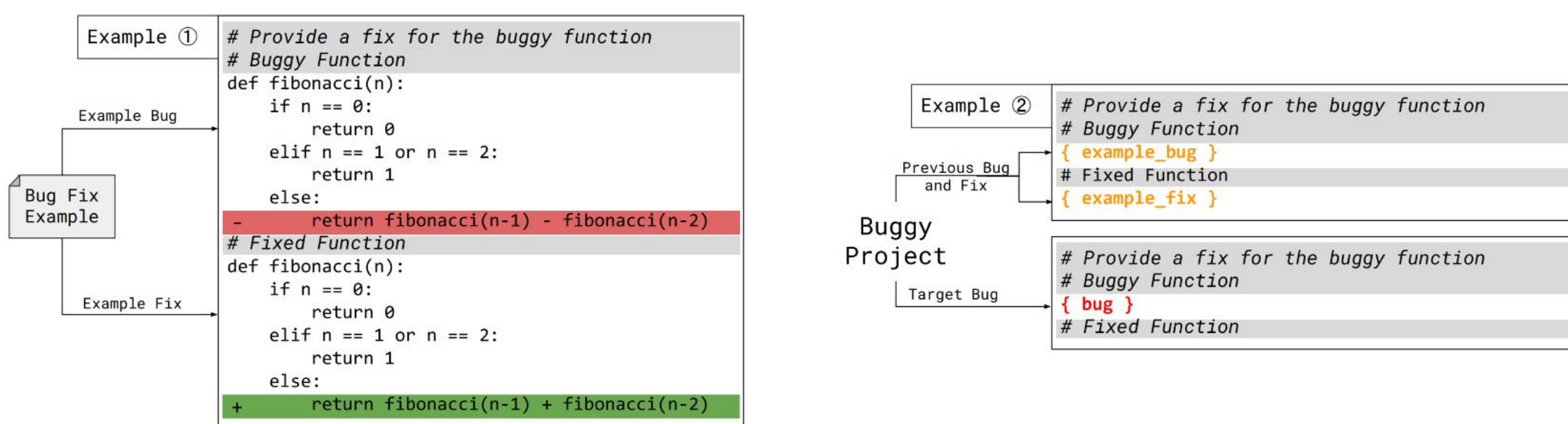
What is Bug Fixing?

- Corrective maintenance process of identifying and resolving bugs in software code
 - Analyzing failure symptoms to locate the bug's root cause
 - Implementing patches to correct unintended behavior
 - Applying verification strategies to ensure patch correctness



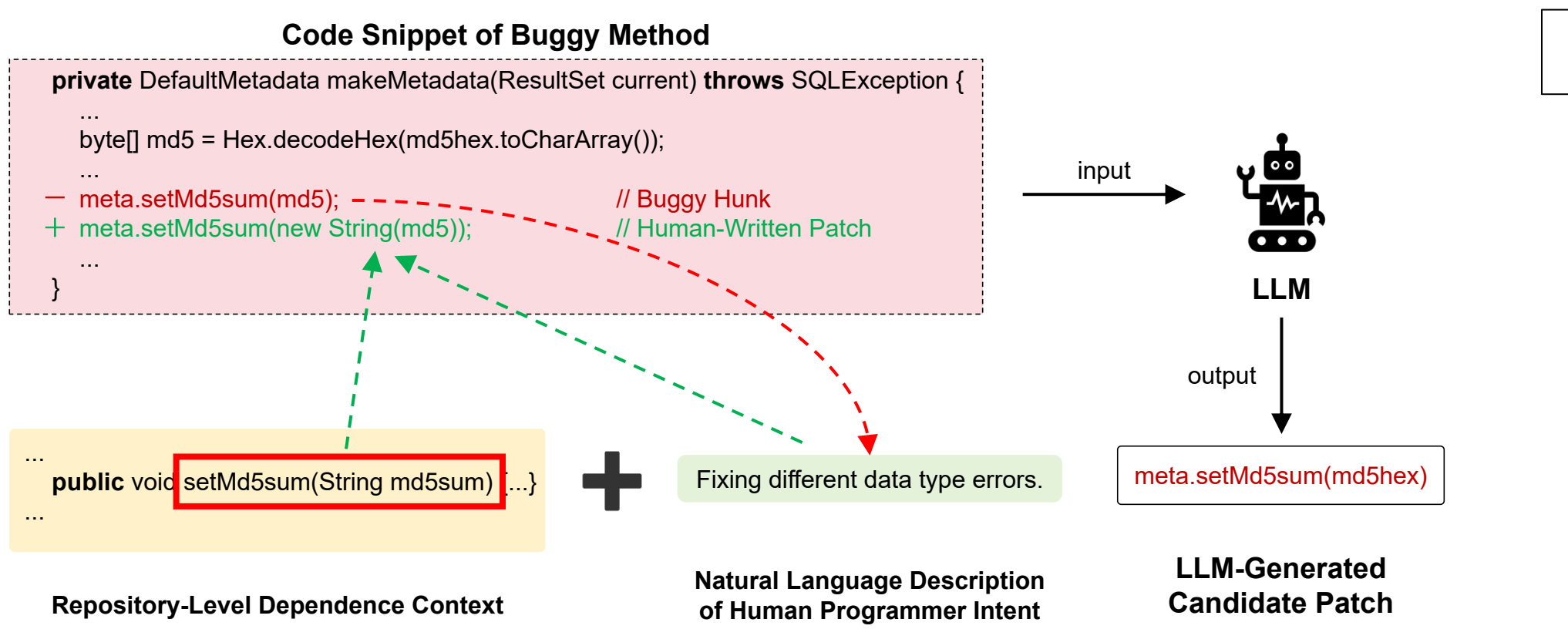
Existing Bug-Fixing Approaches

- Search-based**
 - Utilizes pre-defined patterns mined from historical open-source repositories to generate patches
- Semantic-based**
 - Synthesizes patches by solving repair constraints derived from test suite specifications
- Neural-based**
 - Leverages deep learning models to autonomously learn bug-fixing patterns from code corpora
- LLM-based (in a directly end-to-end manner)**
 - Applies prompt engineering to LLMs using buggy contexts and task-specific examples

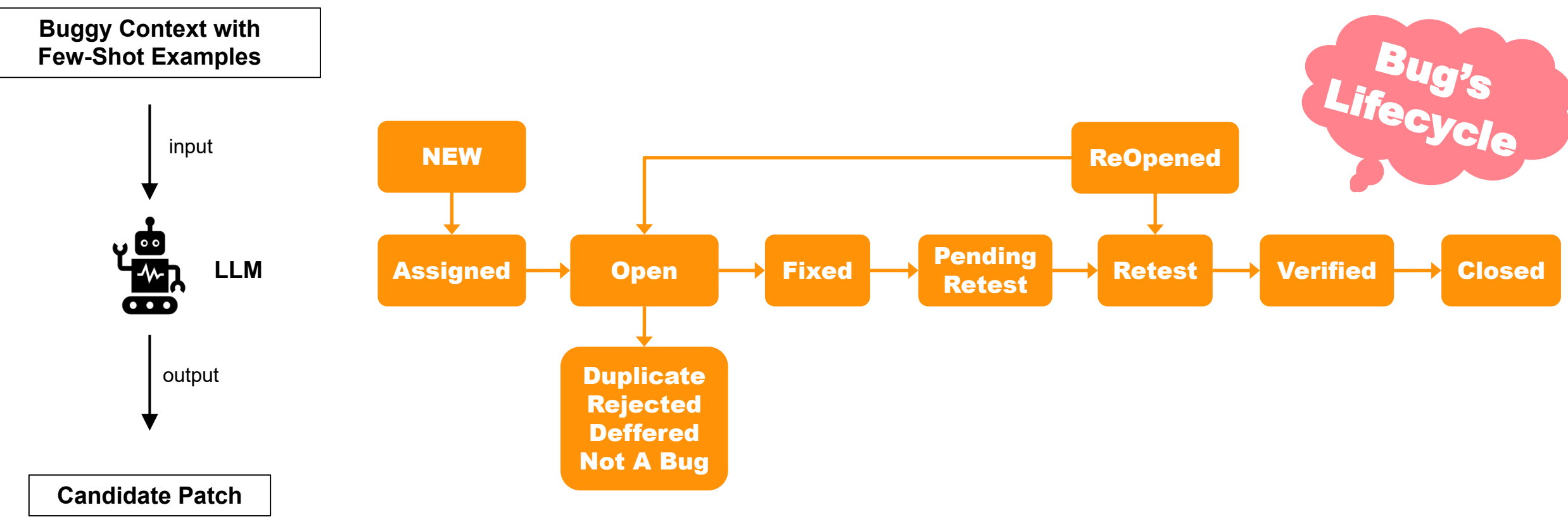


Limitations in LLM-Based Approaches

① Missing additional information as guidance for LLMs



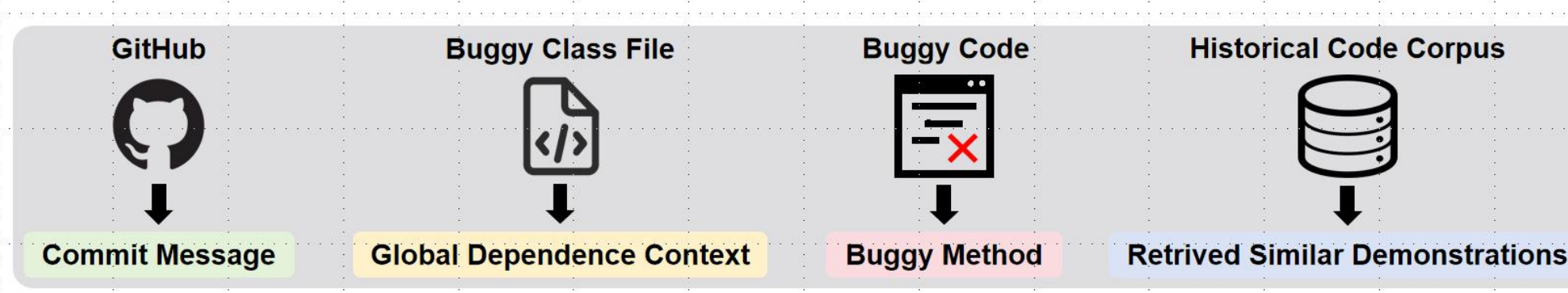
② Treating bug fixing as a single-step, end-to-end process



Key Contributions of PATCH

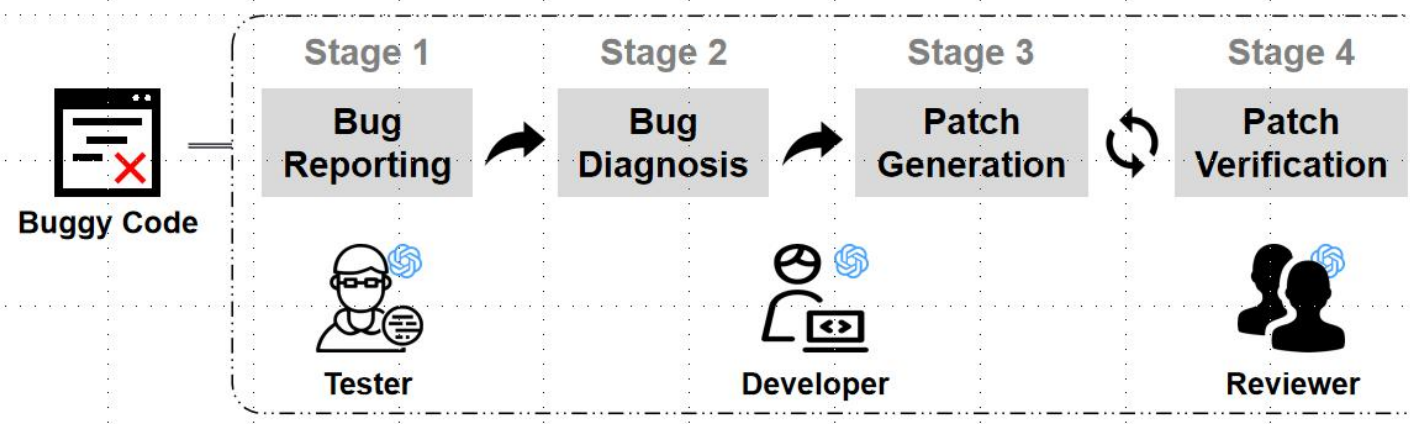
① Context-Enriched Input

- Program Dependencies (class-level / repository-level)
- Formalized Programmer Intent (developer-written commit message)



② Collaborative Simulation

- Enhancing LLM's bug-fixing capabilities via bug management practices
- Breaking down the bug-fixing task into smaller, more manageable subtasks



Incorporates dependence context and the programmer's intent to provide better guidance to LLMs in generating the correct patches for complex bugs

This alignment simulates the interactive behavior of programmers engaged in bug management, enabling LLMs to collaborate and generate correct patches

Experimental Results

Performance Metric Evaluation

- PATCH demonstrates superior performance compared to all the baselines on the augmented BFP benchmark

Model	Evaluation Metric Result			
	Fix@1 (%) ↑	Fix@3 (%) ↑	Fix@5 (%) ↑	Levenshtein Distance ↓
CodeGPT	2.76	5.46	6.65	73.69
DeepSeek-Coder	8.42	11.25	14.14	48.34
CodeGen2	4.31	7.23	10.15	70.13
CodeGeeX2	10.93	15.30	19.41	40.18
InCoder	5.24	8.03	11.60	64.20
Mistral	9.38	14.11	16.68	43.45
CodeLLaMA	9.45	14.46	17.96	43.35
CodeLLaMA	9.80	14.62	17.45	41.20
StarCoder	13.50	17.58	20.89	34.27
GPT-NeoX	8.93	14.56	16.16	57.37
Codex	11.05	-	-	36.33
ChatGPT	14.62	18.99	22.27	33.28
GPT-4	19.96	24.42	26.00	26.07
PATCH	33.97 (14.01% ↑)	37.08 (12.66% ↑)	39.81 (13.81% ↑)	21.44 (17.76% ↓)

Generalizability Evaluation

- PATCH demonstrates a substantial performance advantage over the baselines across different APR benchmarks

Model	# of Exact Match Patches		# of Correct Patches	
	Bugs.jar 1000 bugs	Bears 119 bugs	Time / # Patch	Defects4J 313 bugs
TBar	-	-	3 Hour	63
CoCoNut	66	16	1000	47
SEQUENCER	99	14	300	41
Recorder	61	1	5 Hour	60
CURE	-	-	10000	59
RewardRepair	103	8	200	72
SelfAPR	-	-	-	86
KNOD	-	-	1000	102
CodeBERT	111	12	-	-
GraphCodeBERT	115	12	-	-
CodeT5	150	16	-	-
UniXcoder	164	22	-	-
AlphaRepair	-	-	≤ 5000	92
Replit	-	-	≤ 5000	116
ChatRepair	-	-	≤ 200	142
ThinkRepair	-	-	≤ 125	155
RepairAgent	-	-	Avg. of 117	124
PATCH	180 (9.76% ↑)	28 (27.27% ↑)	≤ 100	169 (9.03% ↑)