FeatX

Editing Software by Editing Features for Repository-Level Code Evolution

Xutian Li[1], Yifeng Zhu[1], Xianlin Zhao[1], Yanzhen Zou[1]*, Lu Zhang[1], Bing Xie[1]

[1]Key Lab of HCST (PKU), MOE; SCS, Peking University, Beijing, China

{xtli25, yifengzhu25}@stu.pku.edu.cn, {zhaoxianlin, zouyz, zhanglucs, xiebing}@pku.edu.cn

Introduction

Background: Software evolution requires translating repository-level feature intent into coordinated code edits across multiple files.

Existing LLM interaction paradigms:

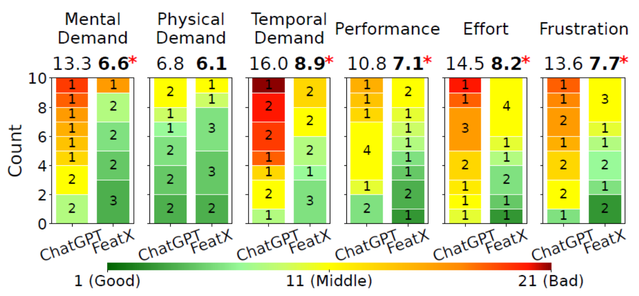
1. Autocompletion (Copilot inline suggestions)
2. Chat-based (Cursor, SWE-Agent)
3. Annotation-based (NL Outlines, Code Shaping)
4. Design-oriented (PAIL, EvoDev)

Gaps:

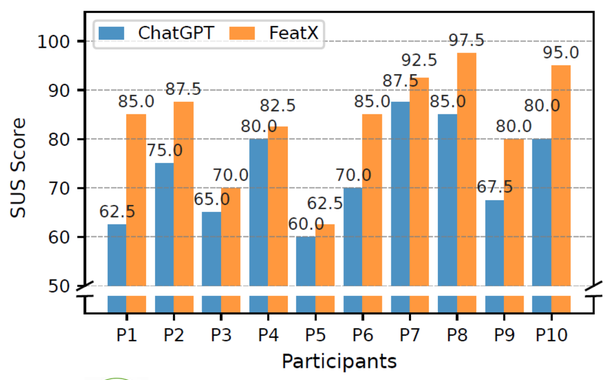
A. Not provide an explicit feature list ⇒ inconsistent!

B. Lack of a feature-to-code mapping ⇒ incur accuracy!

Main Results

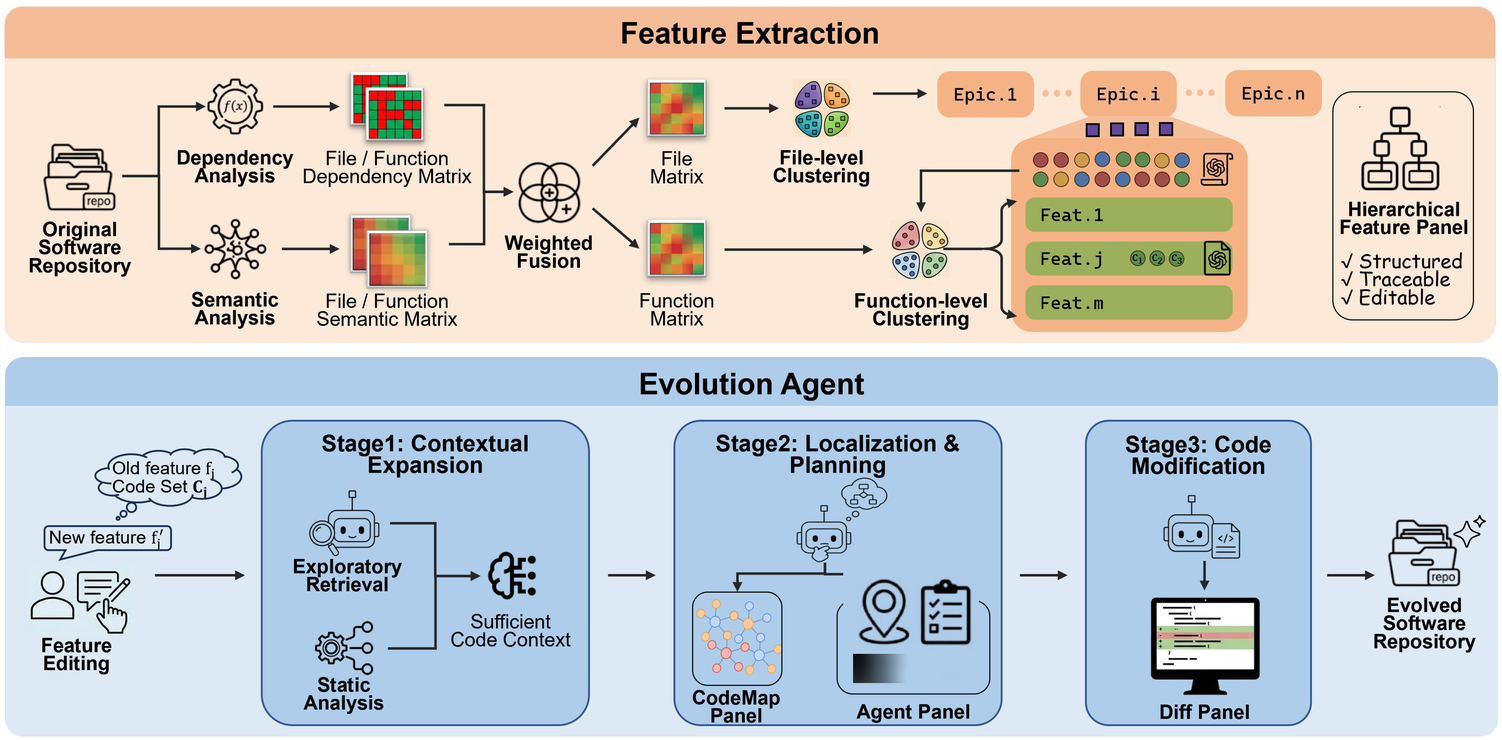


Approach	P (%)	R (%)	F1	Cost ($)
DeepSeek-v3	36.0	9.0	0.143	–
DeepSeek-v3.2-think	29.7	5.5	0.092	–
GPT-4o-mini	37.0	9.7	0.154	1.28
GPT-5.2	30.0	6.0	0.100	17.81
Claude-opus-4.5	**50.7**	*18.4*	*0.270*	45.05
Cursor (Agent)	15.9	13.0	0.143	–
FeatX (Ours)	*41.6*	**35.8**	**0.385**	**0.07**



- **Cognitive Load** (41% ↘)
- **System Usability** (15% ↗)
- **Localization F1** (42.6%↗)
- **Monetary Cost** (only $0.07)

System Design



Feature Extraction

We adapt our previous work *RepoSummary* to build a hierarchical feature list for a whole repository.

Evolution Agent

Three-Stage Agent Flow

I. Context Expansion*

II. Localization & Planning

III. Code Modification

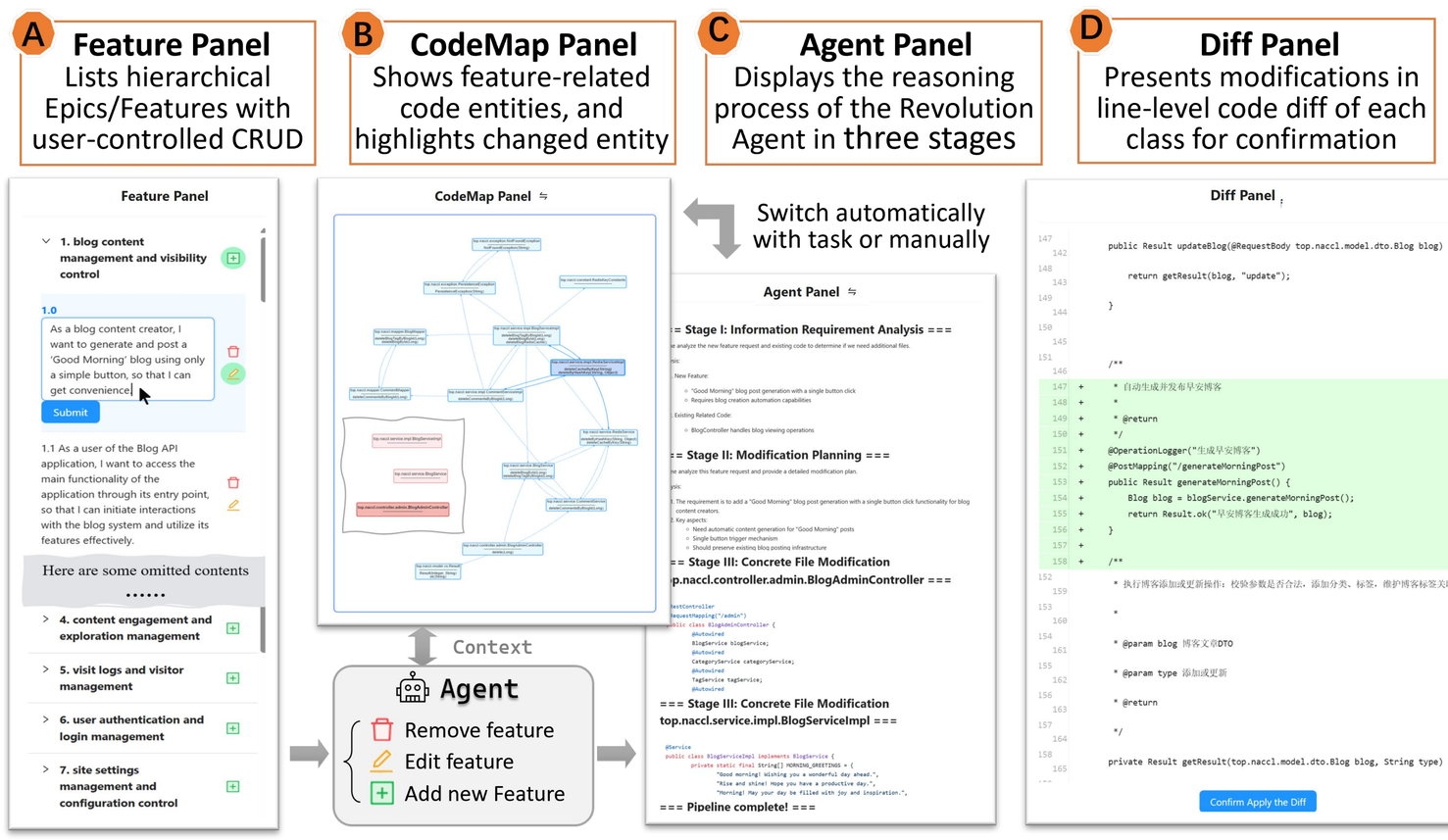
We further apply this idea* to dependency retrieval in *FeatLens*.

Algorithm 1: Iterative Dependency Expansion for Contextual Feature-Code Mapping *

Input: Repository graph G; Entry points method set M_{entry}
Output: Functional subgraph G_{feat}

Initialize $M \leftarrow M_{entry}$, $F \leftarrow \emptyset$, $T \leftarrow \emptyset$
$G_{feat} \leftarrow \emptyset$
repeat
  Trace downstream method calls from M, updating M and G_{feat}
  Trace field, constant, and enum usages in M and F, updating F, T, and G_{feat}
  Trace field definitions and assigning methods from F, updating M and G_{feat}
  Trace declared types referenced by M and F, updating T and G_{feat}
  Trace enclosing classes for all methods and fields, updating T and G_{feat}
  Trace inheritance, nesting, and annotation relations among types, updating T and G_{feat}
until *no new nodes are discovered*
return G_{feat}

Usage Workflow



A Feature Panel
Lists hierarchical Epics/Features with user-controlled CRUD

B CodeMap Panel
Shows feature-related code entities, and highlights changed entity

C Agent Panel
Displays the reasoning process of the Revolution Agent in three stages

D Diff Panel
Presents modifications in line-level code diff of each class for confirmation

More Information

WeChat

Telegram

GitHub

Demo

Video