

SWE-Chain: Benchmarking Coding Agents on Chained Release-Level Package Upgrades

Man Ho Lam, Chaozheng Wang, Hange Liu, Jingyu Xiao, Haau-sing Li, Jen-tse Huang, Terry Yue Zhuo, Michael R. Lyu

ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
The Chinese University of Hong Kong

NEURAL INFORMATION PROCESSING SYSTEMS

Coding Agents for Software Maintenance

- Software projects such as pytest keep evolving after each fix
- Can an agent **build on its own changes** while preserving existing behavior?

2.0.0 → **2.0.1** → **2.0.2**

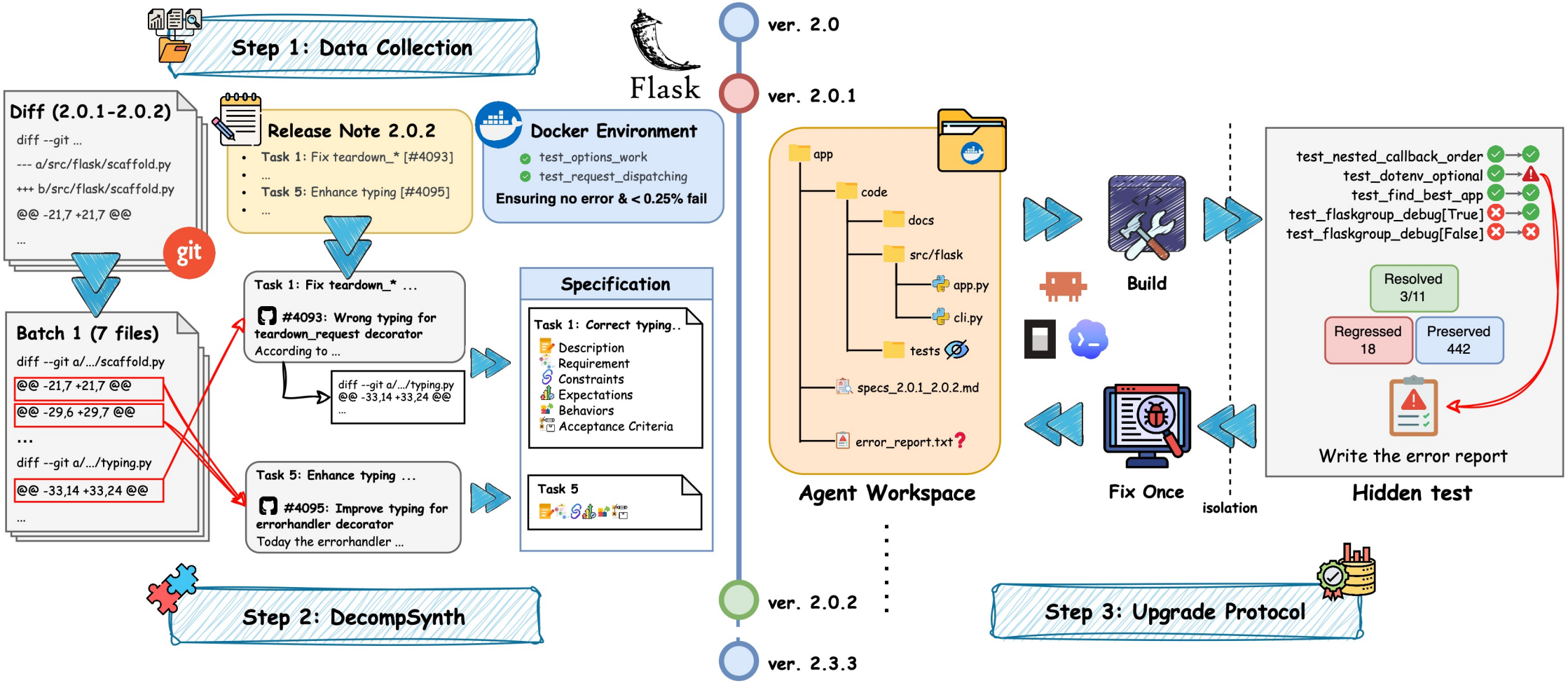
Initial Flask code

Implement the upgrade

Use the agent's 2.0.1 code

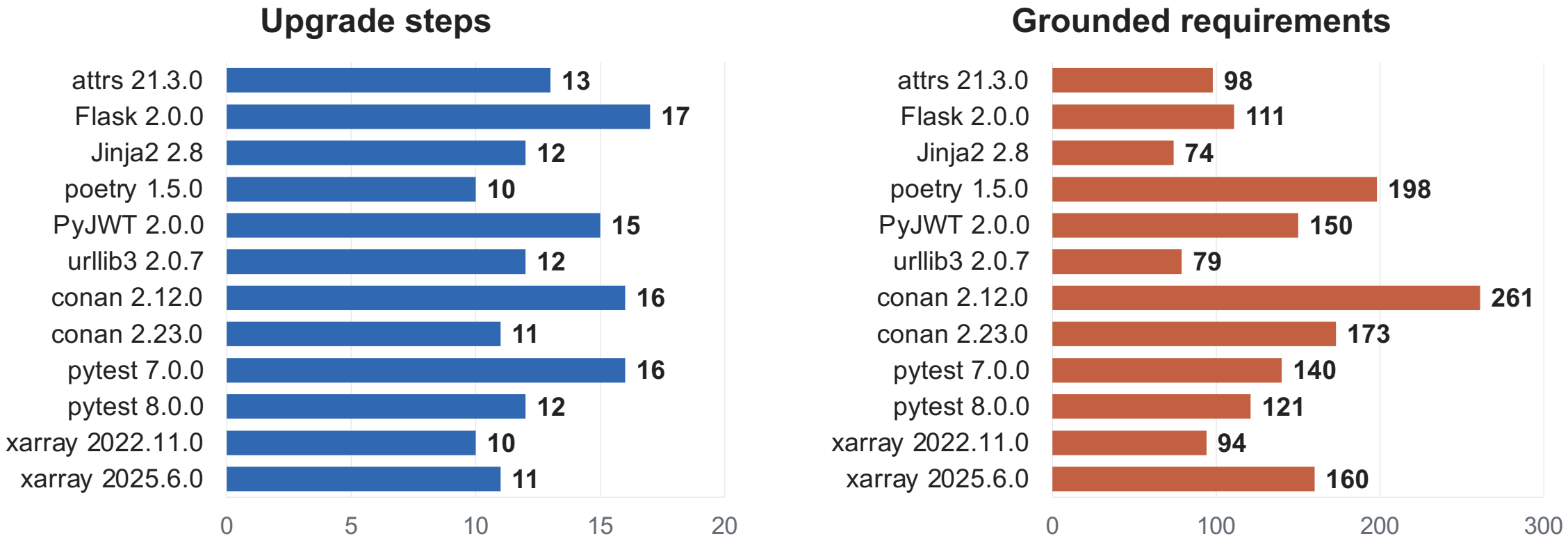
SWE-Chain

- Evaluate **consecutive release upgrades** on the agent's own codebase



Benchmark Scale

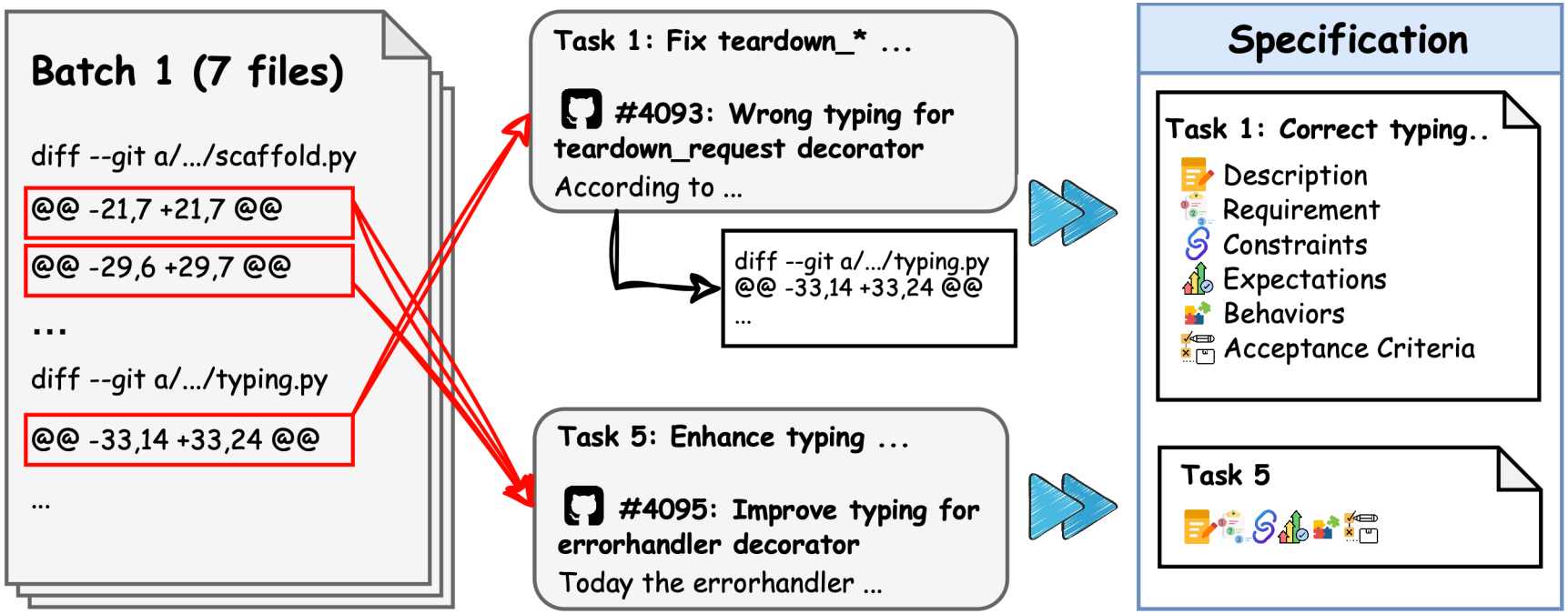
- **12 release chains** across **9 Python packages**
- **155 upgrades** with **1,659 grounded requirements**



Chains are labeled by starting release. Requirement counts include additional synthesized tasks.

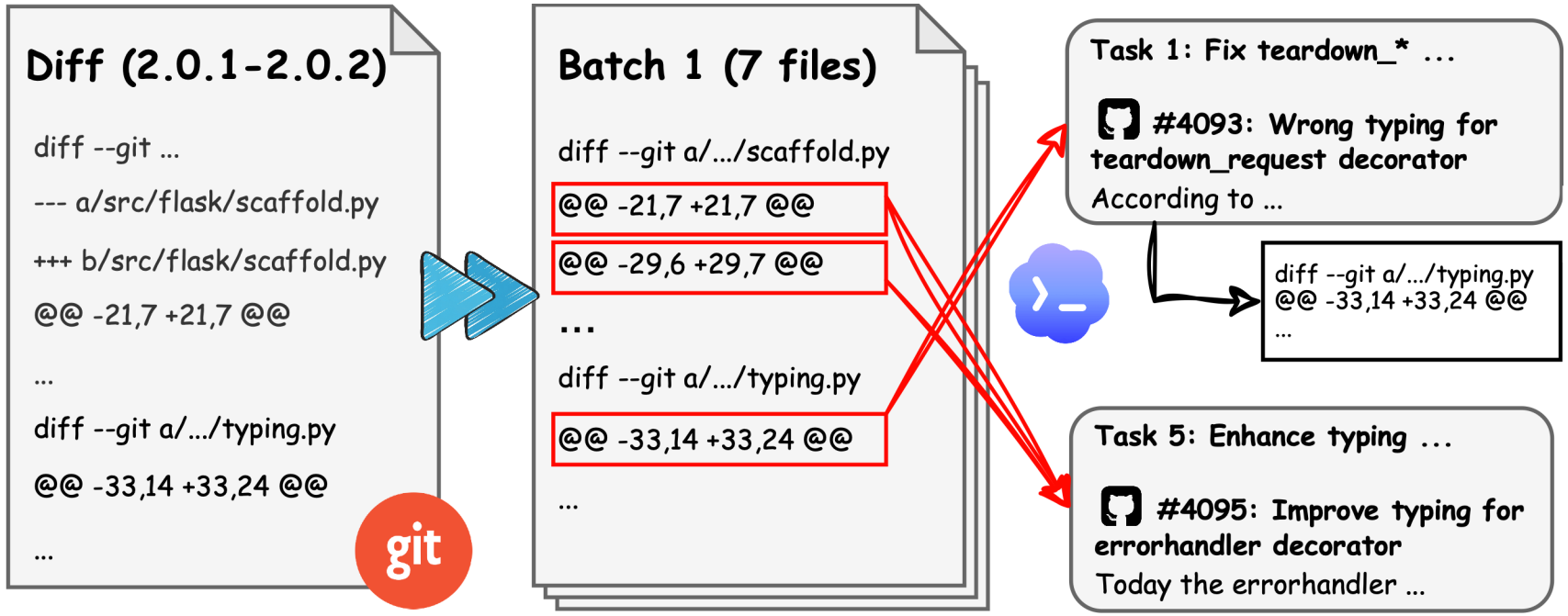
DecompSynth

- **Release notes** describe intent
- **Code diffs** show the actual changes
- **DecompSynth** combines both into **grounded upgrade specifications**



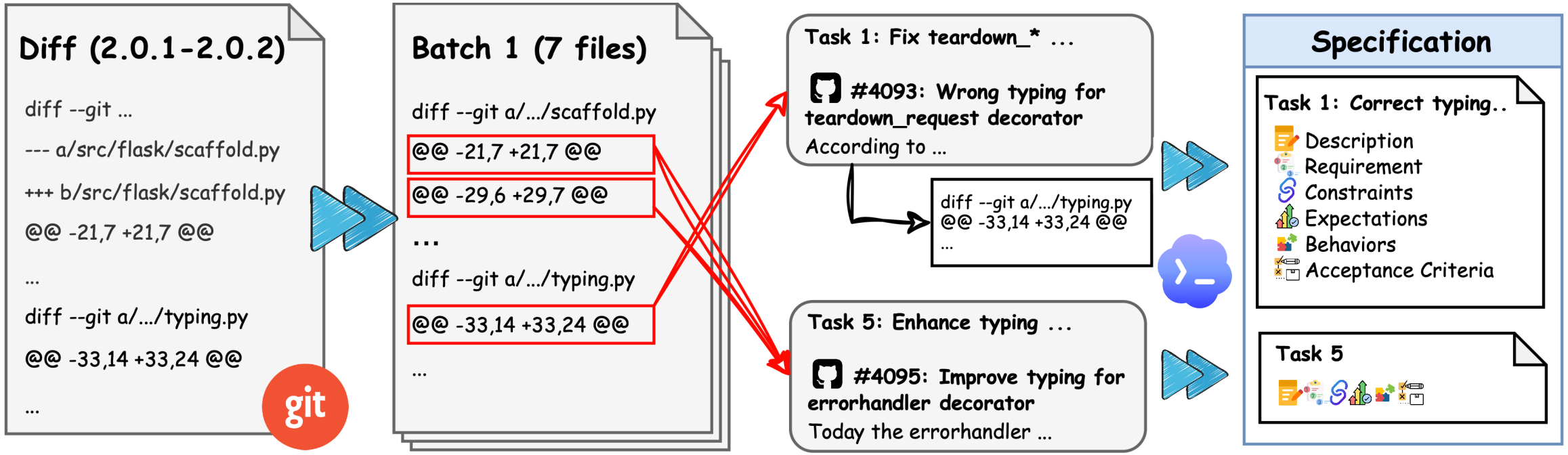
DecompSynth (Divide)

- Split code and test diffs into **manageable batches**
- Match diff hunks to the relevant **release-note tasks** using **GPT-5.4 (Codex)**



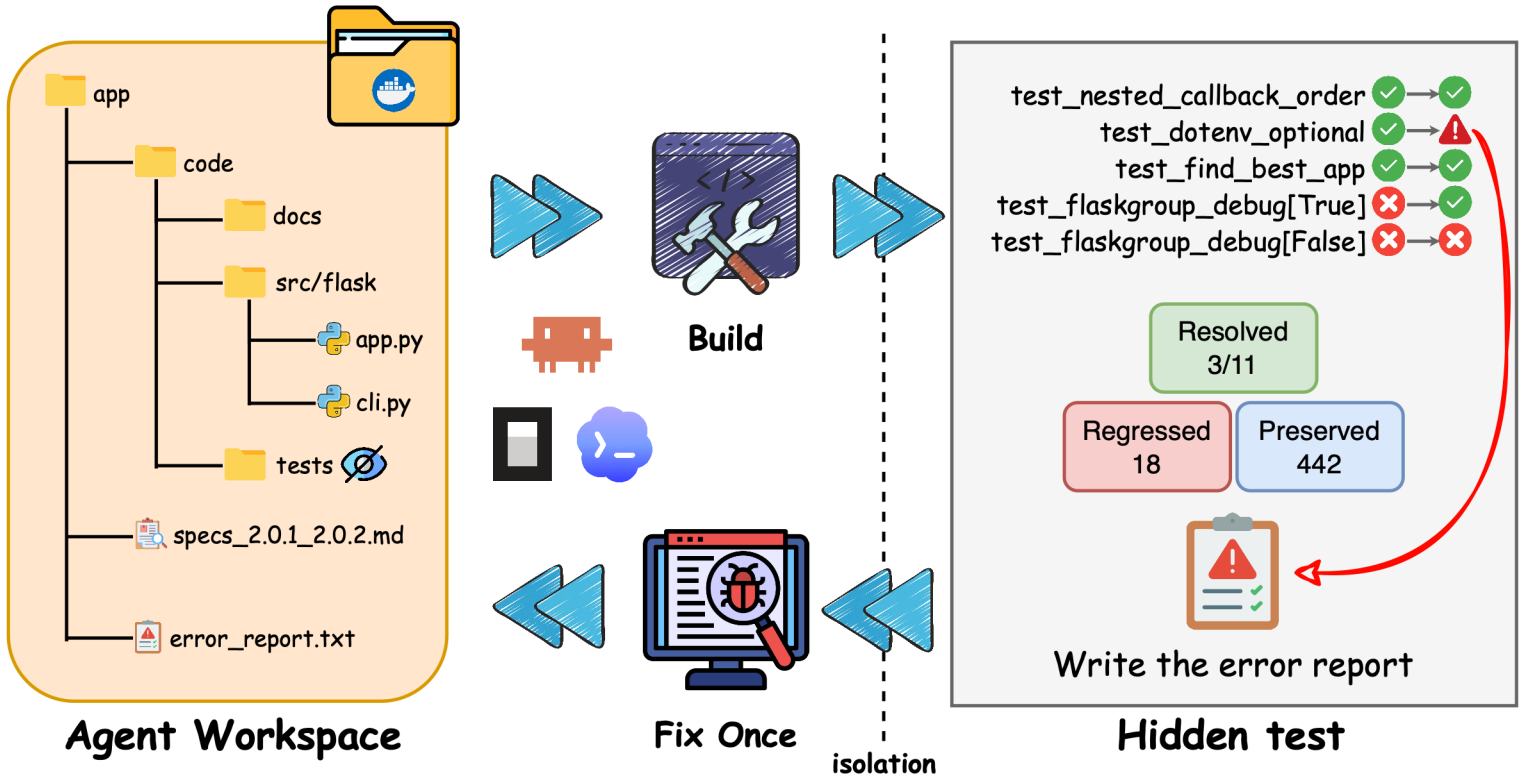
DecompSynth (Conquer)

- Synthesize each task's requirements from its **matched changes**
- Merge task specifications into the **complete release specification**



Upgrade Protocol

- **Build** the release from its specification and the current codebase
- Allow **one Fix** if hidden checks detect execution-level errors
- Carry the resulting codebase into the **next release**



Upgrade Evaluation

- Compare the agent's code before and after each upgrade
  - **Resolving** measures completion of the required changes
  - **Precision** penalizes new regressions on previously passing tests
  - **F1** combines Resolving and Precision

Upgrade-related tests

Before	After	
Any result	**Pass**	**Resolved (TP)**
	Fail	**Unresolved (FN)**

Upgrade-unrelated tests

Before	After	
Pass	**Pass**	**Preserved (TN)**
Pass	**Fail**	**Regressed (FP)**

For previously failing, upgrade-unrelated tests:
fail → pass = recovered; fail → fail = unrecovered. These outcomes are tracked, not scored.

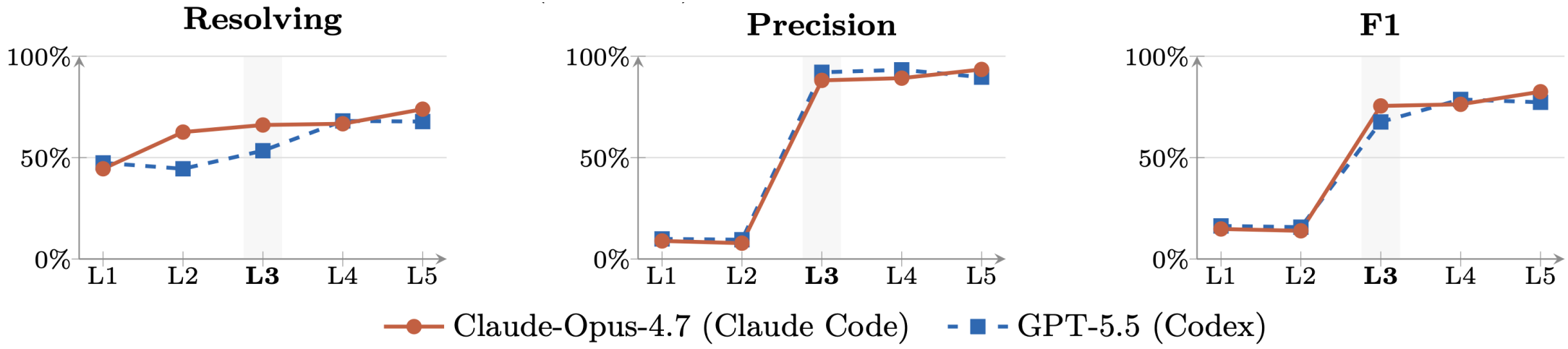
Overall Performance on SWE-Chain

Provider	Scaffold	Model	Resolving		Precision		F1-score	Final Passing
			Build	Build + Fix	Build	Build + Fix		
		Claude-Opus-4.7	**58.3%**	**60.8%**	67.4%	**80.6%**	**68.5%**	91.9%
		Claude-Opus-4.6	38.8%	44.3%	51.3%	64.9%	50.8%	85.8%
		Claude-Sonnet-4.6	34.4%	39.8%	53.4%	66.5%	45.9%	84.5%
		GPT-5.5	52.2%	57.5%	**72.0%**	80.1%	64.8%	**92.2%**
		GPT-5.4	45.3%	47.5%	59.2%	72.2%	54.9%	90.4%
		GPT-5.3-Codex	47.1%	51.8%	70.5%	76.3%	59.2%	90.9%
		GPT-5.4	39.3%	43.4%	54.7%	63.8%	46.5%	89.7%
		GLM-5.1	27.5%	38.1%	31.8%	49.9%	40.1%	84.1%
		MiniMax-M2.7-HS	14.5%	20.2%	25.8%	34.2%	21.2%	70.3%
Average			39.7%	44.8%	54.0%	65.4%	50.2%	86.6%

- 9 model-scaffold configurations across all 12 chains
- **Challenging**: only **44.8%** average Resolving under Build+Fix
- **Discriminative**: F1 ranges from **21.2%** to **68.5%**

Effect of Specification Granularity

- **Expectations and constraints** sharply improve Precision
  - **L1:** Release notes with GitHub issues and PRs, simulating SWE-EVO setting (Le et al.)
  - **L2:** Problem statements only
  - **L3 (default setting):** L2 + conceptual expectations and constraints
  - **L4:** L2 + grounded expectations and constraints
  - **L5:** L4 + behaviors and acceptance criteria



SWE-EVO: Benchmarking Coding Agents in Long-Horizon Software Evolution Scenarios. Le et al.

Key Takeaways

- Chained upgrades remain **challenging**: **44.8% Resolving**, **50.2% F1**
- SWE-Chain is **discriminative** across model-scaffold configurations
- One Fix improves **Precision** more than upgrade coverage
- **Expectations and constraints** sharply improve Precision
- Reliable agents must **complete upgrades** and **preserve functionality**

Webpage

GitHub

Hugging Face