Novelty Assessment Report

Paper: Scalable Supervising Software Agents with Patch Reasoner
PDF URL: https://openreview.net/pdf?id=AXXCo0pOSO
Venue: ICLR 2026 Conference Submission
Year: 2026
Report Generated: 2025-12-30

Abstract

While large language model agents have advanced software engineering tasks, the unscalable nature of existing test-based supervision is limiting the potential improvement of data scaling. The reason is twofold: (1) building and running test sandbox is rather heavy and fragile, and (2) data with high-coverage tests is naturally rare and threatened by test hacking via edge cases. In this paper, we propose R4P, a patch verifier model to provide scalable rewards for training and testing SWE agents via reasoning. We consider that patch verification is fundamentally a reasoning task, mirroring how human repository maintainers review patches without writing and running new reproduction tests. To obtain sufficient reference and reduce the risk of reward hacking, R4P uses a group-wise objective for RL training, enabling it to verify multiple patches against each other's modification and gain a dense reward for stable training. R4P achieves 72.2\% Acc. for verifying patches from SWE-bench-verified, surpassing OpenAI o3. To demonstrate R4P's practicality, we design and train a lite scaffold, Mini-SE, with pure reinforcement learning where all rewards are derived from R4P. As a result, Mini-SE achieves 26.2\% Pass@1 on SWE-bench-verified, showing a 10.0\% improvement over the original Qwen3-32B. This can be further improved to 33.8\% with R4P for test-time scaling. The stable scaling curves in both RL test rewards and test-time accuracy reflect R4P's practical utility for scalable supervision on software agents.

Core Task Landscape

This paper addresses: **Scalable Supervision for Software Engineering Agents Through Patch Verification**

A total of **42 papers** were analyzed and organized into a taxonomy with **24 categories**.

Taxonomy Overview

The research landscape has been organized into the following main categories:

- **Patch Verification and Validation Methods**
- **Agent Architectures and Workflows**
- **Training and Scaling Methodologies**
- **Vulnerability Detection and Exploitation**
- **Bug Fixing and Automated Repair**
- **Evaluation and Benchmarking**
- **Specialized Applications and Domains**
- **Infrastructure and Tooling**
- **Surveys and Conceptual Frameworks**

Complete Taxonomy Tree

- Scalable Supervision for Software Engineering Agents Through Patch Verification Survey Taxonomy
- Patch Verification and Validation Methods
  - Reasoning-Based Patch Verification ★ (2 papers)
  - [0] Scalable Supervising Software Agents with Patch Reasoner (Anon et al., 2026) View paper
  - [23] Dafny as Verification-Aware Intermediate Language for Code Generation (LI Yue-chen, 2025) View paper
  - Test-Based Patch Validation (3 papers)
  - [2] SWE-Dev: Building Software Engineering Agents with Training and Inference Scaling (Wang, 2025) View paper
  - [6] SWE-Synth: Synthesizing Verifiable Bug-Fix Data to Enable Large Language Models in Resolving Real-World Bugs (Minh V. T. Pham, 2025) View paper
  - [17] An analysis of patch plausibility and correctness for generate-and-validate patch generation systems (Zichao Qi, 2015) View paper
  - Hybrid Verification Approaches (2 papers)
  - [9] R2e-gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents (N Jain, 2025) View paper
  - [18] R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents (Jain, 2025) View paper
- Agent Architectures and Workflows
  - Multi-Agent Collaborative Systems (6 papers)
  - [7] PATCH: Empowering Large Language Model with Programmer-Intent Guidance and Collaborative-Behavior Simulation for Automatic Bug Fixing (Yuwei Zhang, 2025) View paper
  - [14] AutoPatch: Multi-Agent Framework for Patching Real-World CVE Vulnerabilities (Seo Min-Jae, 2025) View paper
  - [15] Codepori: Large scale model for autonomous software development by using multi-agents (Zeeshan Rasheed, 2024) View paper
  - [25] Steam: Simulating the interactive behavior of programmers for automatic bug fixing (Zhang Yu-wei, 2023) View paper
  - [32] Architecture of Multi-agent System for Automatic Code Template Maintenance (Elena Akik, 2025) View paper
  - [35] Multi-Agent Code Verification with Compound Vulnerability Detection (Rajan, 2025) View paper

    - Planning-Based Agent Architectures (2 papers)
    - [4] PatchPilot: A Cost-Efficient Software Engineering Agent with Early Attempts on Formal Verification (Li HongWei, 2025) View paper
    - [10] Patchagent: A practical program repair agent mimicking human expertise (Z Yu, 2025) View paper
    - Process-Centric Agent Models (2 papers)
    - [13] SWE-GPT: A Process-Centric Language Model for Automated Software Improvement (Yingwei Ma, 2025) View paper
    - [19] Understanding Software Engineering Agents Through the Lens of Traceability: An Empirical Study (Pujar, 2025) View paper
- Training and Scaling Methodologies
    - Test-Time Compute Scaling (1 papers)
    - [1] Thinking Longer, Not Larger: Enhancing Software Engineering Agents via Scaling Test-Time Compute (Ma YingWei, 2025) View paper
    - Reinforcement Learning-Based Training (1 papers)
    - [40] Multi-Turn Code Generation Through Single-Step Rewards (Jain, 2025) View paper
- Vulnerability Detection and Exploitation
    - Proof-of-Vulnerability Generation (1 papers)
    - [3] Faultline: Automated proof-of-vulnerability generation using llm agents (Nitin, 2025) View paper
    - CVE Reproduction and Dataset Creation (2 papers)
    - [5] From cve entries to verifiable exploits: An automated multi-agent framework for reproducing cves (Ullah Saad, 2025) View paper
    - [16] Eradicating the unseen: Detecting, exploiting, and remediating a path traversal vulnerability across github (Jafar Akhoundali, 2025) View paper
    - Vulnerability Patching (2 papers)
    - [26] Fixing security vulnerabilities with AI in OSS-Fuzz (Zhang, 2024) View paper
    - [39] PATCHEVAL: A New Benchmark for Evaluating LLMs on Patching Real-World Vulnerabilities (Zichao Wei, 2025) View paper
- Bug Fixing and Automated Repair
    - LLM-Based Bug Repair Agents (2 papers)
    - [21] An empirical study on llm-based agents for automated bug fixing (Meng Xiangxin, 2024) View paper
    - [22] Marscode agent: Ai-native automated bug fixing (Liu Yizhou, 2024) View paper
    - Constraint-Based and Path-Sensitive Repair (1 papers)
    - [34] PathFix: Automated Program Repair with Expected Path (He Xu, 2025) View paper
    - Patch Failure Analysis and Improvement (2 papers)
    - [12] PAGENT: Learning to Patch Software Engineering Agents (Xue Haoran, 2025) View paper
    - [42] Patch Overfitting in Program Repair: A Survey (Haoye Tian, n.d.) View paper
- Evaluation and Benchmarking
    - Real-World Task Benchmarks (1 papers)
    - [20] SEC-bench: Automated Benchmarking of LLM Agents on Real-World Software Security Tasks (Lee Hwi-Won, 2025) View paper
    - Patch Quality and Code Analysis (3 papers)
    - [8] How secure is AI-generated code: A large-scale comparison of large language models (Norbert Tihanyi, 2025) View paper
    - [11] Evaluating software development agents: Patch patterns, code quality, and issue complexity in real-world github scenarios (Zhi Chen, 2025) View paper
    - [37] Quality Assurance of LLM-generated Code: Addressing Non-Functional Quality Characteristics (Xin Sun, 2025) View paper
    - Process-Level and Trajectory Evaluation (1 papers)
    - [36] Process-Level Trajectory Evaluation for Environment Configuration in Software Engineering Agents (Li Yinghui, 2025) View paper
- Specialized Applications and Domains
    - Security Patch Detection (1 papers)
    - [31] From LLMs to Agents: A Comparative Evaluation of LLMs and LLM-based Agents in Security Patch Detection (Junxiao Han, 2025) View paper
    - Dataset Adaptation and Migration (1 papers)
    - [27] Multi-Agent Systems for Dataset Adaptation in Software Engineering: Capabilities, Limitations, and Future Directions (Jingyi Chen, 2025) View paper
    - Domain-Specific Code Generation (1 papers)
    - [30] Agent-based code generation for the Gammapy framework (Abhay Mehta, 2025) View paper
- Infrastructure and Tooling
    - Computer-Use and UI Interaction (1 papers)
    - [33] Programming with Pixels: Towards Generalist Software Engineering Agents (P Aggarwal, 2025) View paper
    - Runtime Patching and Self-Healing (2 papers)
    - [24] Self-Healing Autonomous Software Code Development (Jangam, 2022) View paper
    - [41] Runtime programming through model-preserving, scalable runtime patches (Christoph Kirsch, 2011) View paper
    - Quality Assurance Automation (1 papers)
    - [29] Automating quality assurance for industrial web applications: a hybrid approach using API fuzzing and agentic UI testing (Wu, 2025) View paper
- Surveys and Conceptual Frameworks (2 papers)
    - [28] Intelligent Agents for Software Engineering: A Systematic (H RAN, 2025) View paper
    - [38] Trusted Automated Programming (C Pasareanu, 2025) View paper

Narrative

Core task: Scalable supervision for software engineering agents through patch verification. The field has evolved around the challenge of automatically generating, validating, and deploying code changes at scale. The taxonomy reveals several major branches: Patch Verification and Validation Methods explore how to determine whether generated patches are correct and safe, ranging from test-based approaches to reasoning-based verification systems like Dafny Verification[23]. Agent Architectures and Workflows examine the design patterns and execution strategies that enable autonomous software engineering, while Training and Scaling Methodologies address how to build datasets and learning signals that improve agent performance over time. Parallel branches focus on Vulnerability Detection and Exploitation, Bug Fixing and Automated Repair, and domain-specific applications, reflecting the breadth of tasks that software agents

now tackle. Evaluation and Benchmarking provides the measurement infrastructure, and Infrastructure and Tooling supports the practical deployment of these systems.

Within this landscape, a particularly active tension exists between test-driven validation—where patches are accepted if they pass existing test suites—and more rigorous verification approaches that reason about correctness guarantees. Patch Reasoner[0] sits squarely in the reasoning-based verification cluster, emphasizing formal or semantic analysis to validate patches beyond simple test execution. This contrasts with many works in the Bug Fixing branch that rely primarily on test outcomes, and also differs from vulnerability-focused efforts like CVE Verifiable Exploits[5] that prioritize exploit generation over patch correctness proofs. Nearby, Dafny Verification[23] shares the formal verification emphasis, while works like Faultline[3] and PatchPilot[4] blend localization and repair with lighter-weight validation. The central open question remains how to scale rigorous verification without sacrificing the speed and flexibility that make agent-based repair practical for real-world codebases.

Related Works in Same Category

The following **1 sibling papers** share the same taxonomy leaf node with the original paper:

1. Dafny as Verification-Aware Intermediate Language for Code Generation

Authors: LI Yue-chen, Zetzsche, Stefan, Yuekang Li, Somayyajula, et al. (8 authors total) | **Year/Venue**: 2025 | **URL**: View paper

Abstract

Using large language models (LLMs) to generate source code from natural language prompts is a popular and promising idea with a wide range of applications. One of its limitations is that the generated code can be faulty at times, often in a subtle way, despite being presented to the user as correct. In this paper, we explore ways in which formal methods can assist with increasing the quality of code generated by an LLM. Instead of emitting code in a target language directly, we propose that the ...

Relationship Analysis

Both papers belong to the Reasoning-Based Patch Verification category, using formal reasoning methods to validate code without test execution. While the original paper (R4P) uses LLM-based reasoning with group-wise comparison of multiple patches to verify software agent outputs, the candidate paper uses Dafny formal verification as an intermediate language to validate LLM-generated code against specifications before compilation. The key difference is that R4P focuses on post-hoc verification of agent-generated patches through comparative reasoning, whereas the candidate paper integrates formal verification directly into the code generation pipeline to ensure correctness before producing the final output.

Contributions Analysis

Overall novelty summary. The paper introduces R4P, a reasoning-based patch verifier that provides scalable supervision for training software engineering agents without executing tests. It resides in the Reasoning-Based Patch Verification leaf, which contains only two papers including this one. This is a notably sparse research direction within the broader taxonomy of 42 papers across the field. The sibling paper in this leaf focuses on formal verification using Dafny, suggesting that reasoning-based approaches without test execution remain relatively underexplored compared to the more populated Test-Based Patch Validation category, which contains three papers emphasizing test suite execution.

The taxonomy reveals that most patch validation work clusters around test-based methods or hybrid approaches combining tests with heuristics. The Hybrid Verification Approaches leaf contains two papers, while neighboring branches like Bug Fixing and Automated Repair contain multiple papers relying primarily on test outcomes for validation. The paper's emphasis on reasoning mirrors human code review processes, positioning it closer to formal verification traditions than to the execution-heavy workflows dominant in multi-agent collaborative systems and planning-based architectures. This creates a clear boundary: R4P avoids the fragility and scalability issues of test sandboxes that characterize most validation approaches in adjacent categories.

Among 23 candidates examined across three contributions, none were found to clearly refute the paper's claims. The core R4P verifier contribution examined 10 candidates with zero refutable matches, suggesting limited direct prior work on reasoning-based patch verification at this scale. The group-wise training objective examined only 3 candidates, also with no refutations, indicating this training strategy may be relatively novel within the limited search scope. The Mini-SE agent scaffold examined 10 candidates with no refutations, though this may reflect the specific combination of pure RL training with reasoning-based rewards rather than the broader concept of lightweight agent architectures.

Based on the limited search scope of 23 semantically similar papers, the work appears to occupy a relatively unexplored position combining reasoning-based verification with reinforcement learning for agent training. The sparse population of its taxonomy leaf and absence of refuting candidates suggest novelty, though the analysis cannot rule out relevant work outside the top-K semantic matches examined. The 72.2% accuracy claim on SWE-bench-verified and comparison to OpenAI o3 would benefit from broader literature context to assess whether similar verification performance has been reported elsewhere.

This paper presents **3 main contributions**, each analyzed against relevant prior work:

Contribution 1: R4P: A reasoning-based patch verifier model for scalable supervision

Description: The authors introduce R4P, a model that verifies software patches through reasoning rather than test execution. It uses a group-wise objective to compare multiple patches against each other, providing dense rewards for stable reinforcement learning training without requiring golden tests, developer patches, agent trajectories, or runtime sandboxes.

This contribution was assessed against **10 related papers** from the literature. Papers with potential prior art are analyzed in detail with textual evidence; others receive brief assessments.

---

1. A case study of llm for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback

URL: View paper

Brief Assessment

Vulnerability Repair Study[52] focuses on using reasoning and patch validation feedback for vulnerability repair in security contexts, not on developing a reasoning-based patch verifier model for general software engineering agent supervision. The candidate addresses a different problem domain (security vulnerability repair) with different objectives (fixing CVEs) compared to the original's focus on scalable supervision for software agents through group-wise patch verification.

---

2. Training Software Engineering Agents and Verifiers with SWE-Gym

URL: View paper

Brief Assessment

SWE-Gym[49] focuses on creating training environments and uses trajectory-based verifiers for inference-time scaling, not reasoning-based patch verification models. The candidate's verifiers are trained on agent trajectories, whereas R4P verifies patches through reasoning without requiring trajectories, golden tests, or runtime sandboxes.

---

3. Thinking Longer, Not Larger: Enhancing Software Engineering Agents via Scaling Test-Time Compute

URL: View paper

Brief Assessment

Thinking Longer[1] focuses on test-time compute scaling strategies for software engineering agents, not on developing a reasoning-based patch verifier model. The candidate's approach centers on trajectory synthesis and search strategies during inference, whereas the original paper introduces R4P as a dedicated verifier model trained via reinforcement learning with group-wise objectives for patch verification.

---

4. Demystifying llm-based software engineering agents

URL: View paper

Brief Assessment

Demystifying LLM Agents[46] focuses on a simplistic three-phase approach (localization, repair, patch validation) for issue resolution without autonomous agent complexity, rather than developing a reasoning-based patch verifier model for RL supervision. The candidate does not address reward modeling, group-wise verification objectives, or RL training for patch verification.

---

5. Agentic ai software engineer: Programming with trust

URL: View paper

Brief Assessment

Agentic AI Engineer[51] discusses trust mechanisms for AI software engineers broadly, including testing, formal proofs, and guardrails. It does not present a reasoning-based patch verification model with group-wise objectives for RL training as described in the original paper's R4P contribution.

---

6. AutoPatch: Multi-Agent Framework for Patching Real-World CVE Vulnerabilities

URL: View paper

Brief Assessment

AutoPatch[14] focuses on patching CVE vulnerabilities in LLM-generated code using RAG and multi-agent frameworks, not on training patch verifiers for reinforcement learning supervision. The technical approaches and problem domains differ fundamentally.

---

7. Vul-R2: A Reasoning LLM for Automated Vulnerability Repair

URL: View paper

Brief Assessment

Vul-R2[48] focuses on automated vulnerability repair in security contexts, not on patch verification for software engineering agents. The candidate addresses vulnerability-specific reasoning and repair generation, whereas the original contribution concerns verifying patches from SWE agents without requiring tests or sandboxes.

---

8. Evaluating software development agents: Patch patterns, code quality, and issue complexity in real-world github scenarios

URL: View paper

Brief Assessment

Evaluating Software Agents[11] focuses on evaluating existing agent-generated patches using static analysis tools like SonarQube to assess code quality metrics (reliability, security, maintainability). It does not propose a reasoning-based patch verification model for training or supervising agents, which is the core novelty of R4P.

---

9. Adversarial Reasoning for Repair Based on Inferred Program Intent

URL: View paper

Brief Assessment

Adversarial Reasoning Repair[50] focuses on inferring program intent and generating adversarial tests to guide patch generation for automated program repair. It does not propose a reasoning-based patch verification model for supervising software agents through reinforcement learning, which is R4P's core contribution.

---

10. Swe-debate: Competitive multi-agent debate for software issue resolution

URL: View paper

Brief Assessment

Swe-debate[47] focuses on multi-agent debate for fault localization and issue resolution, not on patch verification or reward modeling for RL training. The candidate does not address reasoning-based patch verification models.

Contribution 2: Group-wise training objective for patch verification

Description: The authors propose a group-wise training approach where R4P assesses each patch by comparing it against others in a candidate set. This formulation provides mutual contextual information to compensate for the absence of tests, reduces reward hacking risk, and offers denser rewards than binary classification for more stable convergence.

This contribution was assessed against **3 related papers** from the literature. Papers with potential prior art are analyzed in detail with textual evidence; others receive brief assessments.

---

1. CCrepairBench: A High-Fidelity Benchmark and Reinforcement Learning Framework for C++ Compilation Repair

URL: View paper

Brief Assessment

CCrepairBench[45] focuses on C++ compilation repair using RL with hybrid reward signals for semantic correctness. It does not employ a group-wise training objective where patches are compared against each other for mutual contextual information, which is the core novelty of the original paper's contribution.

---

2. Keeping authorities" honest or bust" with decentralized witness cosigning

URL: View paper

Brief Assessment

Decentralized Witness Cosigning[44] focuses on collective signing protocols for network authorities (e.g., certificate authorities, timestamping services) using cryptographic multisignatures, not on patch verification or reinforcement learning for software engineering tasks.

---

3. Don't Waste Mistakes: Leveraging Negative RL-Groups via Confidence Reweighting

URL: View paper

Brief Assessment

Negative RL Groups[43] focuses on leveraging negative groups in mathematical reasoning tasks through confidence-weighted penalties on incorrect responses. This is fundamentally different from R4P's group-wise patch verification approach, which compares multiple code patches against each other to provide mutual contextual information for software engineering tasks.

Contribution 3: Mini-SE: A lite execution-free agentic scaffold trained with pure RL

Description: The authors develop Mini-SE, a lightweight software engineering agent with issue-resolving code search and edit capabilities. It is trained using pure reinforcement learning supervised entirely by R4P without test execution during rollout, demonstrating the practical utility of reasoning-based verification for scalable agent training.

This contribution was assessed against **10 related papers** from the literature. Papers with potential prior art are analyzed in detail with textual evidence; others receive brief assessments.

---

1. Knowledge-enhanced software refinement: leveraging reinforcement learning for search-based quality engineering

URL: View paper

Brief Assessment

Knowledge-enhanced Refinement[54] focuses on search-based quality engineering with knowledge enhancement for software refinement, not on developing execution-free agentic scaffolds for issue resolution trained with pure RL.

---

2. VerlTool: Towards Holistic Agentic Reinforcement Learning with Tool Use

URL: View paper

Brief Assessment

VerlTool[56] is a general framework for agentic RL with tool use across multiple domains, not a specific software engineering agent. It does not present an execution-free issue-resolving agent comparable to Mini-SE.

---

3. Agent-RLVR: Training Software Engineering Agents via Guidance and Environment Rewards

URL: View paper

Brief Assessment

Agent-RLVR[59] focuses on training agents with guidance mechanisms and environment rewards in software engineering tasks, but does not claim to be execution-free. The candidate explicitly uses 'unit tests' and 'docker container' environments for verification, which contradicts the execution-free nature of Mini-SE.

---

4. Reinforcement learning for machine learning engineering agents

URL: View paper

Brief Assessment

RL Engineering Agents[55] focuses on reinforcement learning for machine learning engineering tasks (e.g., Kaggle competitions), not general software engineering issue resolution. The candidate does not address execution-free software engineering agents for repository-level bug fixing.

---

5. Towards concept based software engineering for intelligent agents

URL: View paper

Brief Assessment

Concept Based Engineering[61] focuses on hierarchical reinforcement learning and component-based development for AI systems in general, not specifically on execution-free software engineering agents or issue-resolving code generation trained with pure RL.

---

6. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution

URL: View paper

Brief Assessment

Swe-rl[53] focuses on a different scaffold design (Agentless Mini) and applies RL to real-world PR data for issue resolution, whereas the original paper's Mini-SE is specifically designed as an execution-free scaffold with issue-resolving-oriented code search and edit capabilities trained using R4P rewards without test execution during rollout.

---

7. A DQN-based agent for automatic software refactoring

URL: View paper

Brief Assessment

DQN Refactoring Agent[62] focuses on automatic software refactoring using DQN-based reinforcement learning, not on issue-resolving software engineering agents trained without test execution. The candidate addresses a different problem domain (code refactoring) with different technical approaches.

---

8. A Practitioner's Guide to Multi-turn Agentic Reinforcement Learning

URL: View paper

Brief Assessment

Agentic RL Guide[60] does not present an execution-free software engineering agent. The guide focuses on multi-turn RL training recipes across textworld, alfworld, and swe-gym environments, but does not describe developing a lightweight scaffold with issue-resolving capabilities trained purely with RL without test execution.

---

9. From Learning Agents to Agile Software: Reinforcement Learning's Transformative Role in Requirements Engineering

URL: View paper

Brief Assessment

RL Requirements Engineering[57] is a conceptual study on applying reinforcement learning to requirements engineering in software development, not a technical implementation of an execution-free software engineering agent trained with pure RL like Mini-SE.

---

10. Training Long-Context, Multi-Turn Software Engineering Agents with Reinforcement Learning

URL: View paper

Brief Assessment

Long-Context Training[58] focuses on multi-turn interactive agents with environment feedback and test-based rewards, not execution-free training. Their agent uses test execution for rewards ('r(τ)∈{0,1}is the terminal reward' from test suites), contrasting with Mini-SE's pure R4P-based supervision without any test execution during rollout.

Appendix: Text Similarity Detection

No high-similarity text segments were detected across any compared papers.

References

- [0] Scalable Supervising Software Agents with Patch Reasoner View paper
- [1] Thinking Longer, Not Larger: Enhancing Software Engineering Agents via Scaling Test-Time Compute View paper
- [2] SWE-Dev: Building Software Engineering Agents with Training and Inference Scaling View paper
- [3] Faultline: Automated proof-of-vulnerability generation using llm agents View paper
- [4] PatchPilot: A Cost-Efficient Software Engineering Agent with Early Attempts on Formal Verification View paper
- [5] From cve entries to verifiable exploits: An automated multi-agent framework for reproducing cves View paper
- [6] SWE-Synth: Synthesizing Verifiable Bug-Fix Data to Enable Large Language Models in Resolving Real-World Bugs View paper
- [7] PATCH: Empowering Large Language Model with Programmer-Intent Guidance and Collaborative-Behavior Simulation for Automatic Bug Fixing View paper
- [8] How secure is AI-generated code: A large-scale comparison of large language models View paper
- [9] R2e-gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents View paper
- [10] Patchagent: A practical program repair agent mimicking human expertise View paper
- [11] Evaluating software development agents: Patch patterns, code quality, and issue complexity in real-world github scenarios View paper
- [12] PAGENT: Learning to Patch Software Engineering Agents View paper
- [13] SWE-GPT: A Process-Centric Language Model for Automated Software Improvement View paper
- [14] AutoPatch: Multi-Agent Framework for Patching Real-World CVE Vulnerabilities View paper
- [15] Codepori: Large scale model for autonomous software development by using multi-agents View paper
- [16] Eradicating the unseen: Detecting, exploiting, and remediating a path traversal vulnerability across github View paper
- [17] An analysis of patch plausibility and correctness for generate-and-validate patch generation systems View paper
- [18] R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents View paper
- [19] Understanding Software Engineering Agents Through the Lens of Traceability: An Empirical Study View paper
- [20] SEC-bench: Automated Benchmarking of LLM Agents on Real-World Software Security Tasks View paper
- [21] An empirical study on llm-based agents for automated bug fixing View paper
- [22] Marscode agent: Ai-native automated bug fixing View paper
- [23] Dafny as Verification-Aware Intermediate Language for Code Generation View paper
- [24] Self-Healing Autonomous Software Code Development View paper
- [25] Steam: Simulating the interactive behavior of programmers for automatic bug fixing View paper
- [26] Fixing security vulnerabilities with AI in OSS-Fuzz View paper
- [27] Multi-Agent Systems for Dataset Adaptation in Software Engineering: Capabilities, Limitations, and Future Directions View paper
- [28] Intelligent Agents for Software Engineering: A Systematic View paper
- [29] Automating quality assurance for industrial web applications: a hybrid approach using API fuzzing and agentic UI testing View paper
- [30] Agent-based code generation for the Gammapy framework View paper
- [31] From LLMs to Agents: A Comparative Evaluation of LLMs and LLM-based Agents in Security Patch Detection View paper
- [32] Architecture of Multi-agent System for Automatic Code Template Maintenance View paper
- [33] Programming with Pixels: Towards Generalist Software Engineering Agents View paper
- [34] PathFix: Automated Program Repair with Expected Path View paper
- [35] Multi-Agent Code Verification with Compound Vulnerability Detection View paper
- [36] Process-Level Trajectory Evaluation for Environment Configuration in Software Engineering Agents View paper
- [37] Quality Assurance of LLM-generated Code: Addressing Non-Functional Quality Characteristics View paper
- [38] Trusted Automated Programming View paper
- [39] PATCHEVAL: A New Benchmark for Evaluating LLMs on Patching Real-World Vulnerabilities View paper
- [40] Multi-Turn Code Generation Through Single-Step Rewards View paper
- [41] Runtime programming through model-preserving, scalable runtime patches View paper
- [42] Patch Overfitting in Program Repair: A Survey View paper
- [43] Don't Waste Mistakes: Leveraging Negative RL-Groups via Confidence Reweighting View paper
- [44] Keeping authorities" honest or bust" with decentralized witness cosigning View paper
- [45] CCrepairBench: A High-Fidelity Benchmark and Reinforcement Learning Framework for C++ Compilation Repair View paper
- [46] Demystifying llm-based software engineering agents View paper
- [47] Swe-debate: Competitive multi-agent debate for software issue resolution View paper
- [48] Vul-R2: A Reasoning LLM for Automated Vulnerability Repair View paper
- [49] Training Software Engineering Agents and Verifiers with SWE-Gym View paper
- [50] Adversarial Reasoning for Repair Based on Inferred Program Intent View paper
- [51] Agentic ai software engineer: Programming with trust View paper
- [52] A case study of llm for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback View paper
- [53] Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution View paper
- [54] Knowledge-enhanced software refinement: leveraging reinforcement learning for search-based quality engineering View paper
- [55] Reinforcement learning for machine learning engineering agents View paper
- [56] VerlTool: Towards Holistic Agentic Reinforcement Learning with Tool Use View paper
- [57] From Learning Agents to Agile Software: Reinforcement Learning's Transformative Role in Requirements Engineering View paper
- [58] Training Long-Context, Multi-Turn Software Engineering Agents with Reinforcement Learning View paper
- [59] Agent-RLVR: Training Software Engineering Agents via Guidance and Environment Rewards View paper

- [60] A Practitioner's Guide to Multi-turn Agentic Reinforcement Learning View paper
- [61] Towards concept based software engineering for intelligent agents View paper
- [62] A DQN-based agent for automatic software refactoring View paper