SWE-AGILE: A Software Agent Framework for Efficiently Managing Dynamic Reasoning Context

Shuquan Lian[1] **Juncheng Liu**[2] **Yazhe Chen**[1] **Yuhong Chen**[1] **Hui Li**[1] ✉
[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing Ministry of Education of China, Xiamen University
[2] Microsoft
shuquanlian@stu.xmu.edu.cn, hui@xmu.edu.cn

Abstract

Prior representative ReAct-style approaches in autonomous Software Engineering (SWE) typically lack the explicit System-2 reasoning required for deep analysis and handling complex edge cases. While recent reasoning models demonstrate the potential of extended Chain-of-Thought (CoT), applying them to the multi-turn SWE task creates a fundamental dilemma: retaining full reasoning history leads to context explosion and "Lost-in-the-Middle" degradation, while discarding it would force the agent to redundantly re-reason at every step. To address these challenges, we propose SWE-AGILE, a novel software agent framework designed to bridge the gap between reasoning depth, efficiency, and context constraints. SWE-AGILE introduces a Dynamic Reasoning Context strategy, maintaining a "sliding window" of detailed reasoning for immediate continuity to prevent redundant re-analyzing, while compressing historical reasoning content into concise Reasoning Digests. Empirically, SWE-AGILE sets a new standard for 7B-8B models on SWE-Bench-Verified using only 2.2k trajectories and 896 tasks. Code is available at https://github.com/KDEGroup/SWE-AGILE.

1 Introduction

The blossoming of Large Language Models (LLMs) and Code Models (Hui et al., 2024a; Li et al., 2025a) has revolutionized software engineering (SWE), enhancing efficiency in various tasks. Particularly, there is a surge of works on SWE agents for autonomously navigating repositories, localizing bugs, and fixing bugs (Jimenez et al., 2024; Yang et al., 2024b), i.e., the SWE task. The SWE task is difficult because it involves using tools, writing code, and debugging over multiple turns. These activities are closely connected, requiring heterogeneous reasoning capabilities.

Prior representative REACT-style (Yao et al., 2023) approaches like SWE-Dev (Wang et al., 2025a) and SWE-smith (Yang et al., 2025b) train models with limited context length (e.g., Qwen2.5 (Yang et al., 2024a) and Qwen3 (Yang et al., 2025a)) to generate actions alongside shallow thought traces. However, without explicit *System-2 reasoning* (Li et al., 2025b) that is more analytical and deliberate, it is hard for these methods to perform deep analysis and handle edge cases correctly.

Recent advancements in System-2 reasoning models, such as OpenAI o1 (Jaech et al., 2024) and DeepSeek-R1 (DeepSeek-AI, 2025), suggest that extending the Chain-of-Thought (CoT) (Wei et al., 2022) length significantly enhances LLM's problem-solving capabilities. Attempting to harness this potential in automatic SWE, agentic scaffolds relying on powerful reasoning LLMs allow for long CoT during generation but discard these reasoning traces in the historical context, retaining only a concise description and the final action.

One key observation emerges from the recent progress in System-2 reasoning models is that discarding reasoning content upon the arrival of the second round of messages results in significant *token inefficiency*, forcing the model to *redundantly re-reason* through the entire problem for each subsequent tool call, as observed on DeepSeek-V3.2 (DeepSeek-AI et al., 2025). MiniMax M2 (MiniMaxAI, 2025) also claims that Agents require *Interleaved Thinking* and retaining the full session history, including the reasoning content. This approach is made possible by its massive context capacity of 204,800 tokens.

However, indiscriminately applying full-history Interleaved Thinking to the multi-turn SWE task presents fundamental scalability and efficiency challenges. SWE agents typically engage in frequent interactions involving extensive code retrievals and verbose execution logs. When retaining the long CoT from every step, *the context window expands rapidly*. This creates two critical issues. First, even for models capable of processing

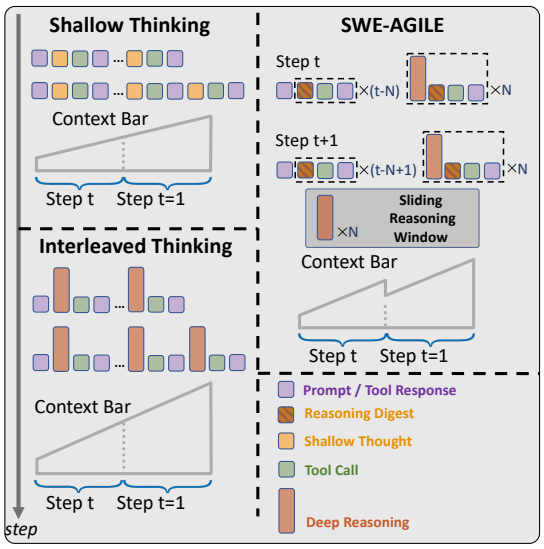

Figure 1: A comparison of context growth patterns in the multi-turn SWE task: (1) Shallow Thinking maintains low context cost but lacks reasoning depth. (2) Interleaved Thinking enables deep System-2 reasoning but suffers from rapid, linear context growth (steep context bar). (3) SWE-AGILE enables a sustainable "Sawtooth" growth pattern. Within the *Sliding Reasoning Window*, the agent engages in deep reasoning (steep slope similar to Interleaved Thinking); however, as steps progress, historical thoughts are compressed into concise *Reasoning Digests*.

long inputs, performance often degrades as context length increases, a phenomenon known as "Lost-in-the-Middle" or attention dilution (Liu et al., 2024), where the model struggles to retrieve relevant information from the middle of a long sequence. Second, processing such massive sequences demands *excessive GPU memory* and drastically reduces *training speed* due to computational overhead.

To address the above issues, we propose a novel framework SWE-AGILE designed to bridge the gap between reasoning depth, efficiency, and context constraints in in the multi-turn SWE tasks. Fig. 1 contrasts our approach with Shallow Thinking and Interleaved Thinking at the level of inference. The main contributions of this work are summarized as follows:

- **Dynamic Reasoning Context:** We propose a hybrid dynamic context management strategy that compresses every historical reasoning content into a concise reasoning digest for long-term retention, while preserving a "Last-N-Steps" sliding reasoning window of long CoT to maintain cognitive continuity in the working context and avoid redundantly re-analyzing the global state at every turn.
- **Trajectory Snapshot Training:** We introduce a snapshot-based training objective to address the misalignment between standard training and dynamic inference. By decomposing trajectories into discrete snapshots with context-aware masking, we force the model to learn under the constraints of dynamic context at runtime.
- **Backfilling Data Synthesis:** We develop a "Hindsight Backfill" pipeline that augments successful trajectories with detailed reasoning content and reasoning digests. This process leverages future ground-truth actions to synthesize high-quality, format-compliant training data tailored to our dynamic context constraints.
- **Compression-Aware Optimization:** We design a trajectory-level Compression Rate Reward within the Reinforcement Learning with Verifiable Rewards (RLVR) process. This mechanism incentivizes the model to generate sufficiently detailed reasoning for problem-solving while maximizing the conciseness of the digests, effectively balancing performance with token efficiency.

Empirically, we validate SWE-AGILE on the SWE-Bench Verified benchmark. Utilizing merely **2.2k** training trajectories, SWE-AGILE achieves a **24.1%** success rate, surpassing all existing 7B/8B baselines. We attribute this success to the effectiveness of our proposed paradigm: by fundamentally resolving the conflict between reasoning depth and context constraints, SWE-AGILE effectively elicits latent System-2 reasoning ability.

2 Our Method

Fig. 2 provides an overview of SWE-AGILE. To enable deep reasoning within a sustainable context window, our framework is realized through three parts: (1) Trajectory Snapshot Training (Sec. 2.1); (2) Backfilling Reasoning and Digest (Sec. 2.2); and (3) RLVR with Trajectory Level Compression (Sec. 2.3).

2.1 Dynamic Reasoning Context and Trajectory Snapshot Training

SWE-AGILE formalizes the agent's interaction as a sequence of steps $t = \{1, 2, 3, \cdots, T\}$. At each step, the generation and the context management operate as follows: for step t, the model is required to generate a composite response y_t, consisting of

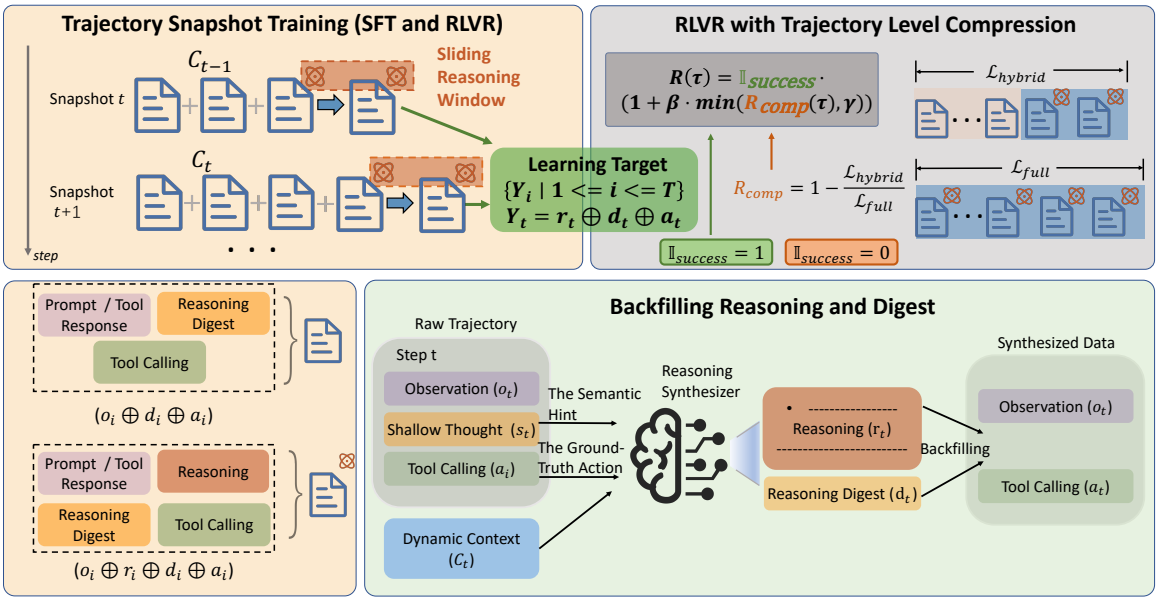

Figure 2: Overview of SWE-AGILE. (1) *Trajectory Snapshot Training*: We decompose long trajectories into discrete snapshots. In each snapshot, historical reasoning traces outside the *Sliding Reasoning Window* are replaced by digests (d_i) and masked from the loss, forcing the model to learn the target Y_t (Reasoning $r_t \rightarrow$ Digest $d_t \rightarrow$ Action a_t) based on a compact context C_t. (2) *Backfilling Reasoning and Digest*: We synthesize high-quality training data by employing a reasoning model to augment raw trajectories. Conditioned on the ground-truth future action, the semantic hint and the dynamic context, the model backfills detailed reasoning (r_t) and concise digest (d_t). (3) *RLVR with Trajectory Level Compression:* We introduce a compression-aware reward function. The model is incentivized to maximize the reduction ratio between the actual hybrid context ($\mathcal{L}_{hybrid}$) and the hypothetical full context ($\mathcal{L}_{full}$), thereby learning to efficiently compress historical information without sacrificing task success ($\mathbb{I}_{success}$).

three strictly ordered components:

$$Y_t = r_t \oplus d_t \oplus a_t \tag{1}$$

where r_t is the detailed reasoning for analyzing the current state, d_t is the reasoning digest (i.e., a concise digest of r_t generated immediately thereafter), and a_t is the executable action.

To generate Y_t, the model conditions on a hybrid context C_t. Let N denote the size of the sliding reasoning window. The context C_t is formulated as the concatenation of long-term history with condensed reasoning and a detailed reasoning sliding window. Specifically, while the full history of environmental interactions (observations and actions) is always retained, the reasoning traces (r) outside the sliding window are replaced by their concise digests (d):

$$C_t = \Big[\bigcup_{i=0}^{t-N-1} (o_i, d_i, a_i)\Big] \oplus \Big[\bigcup_{j=t-N}^{t-1} (o_j, r_j, d_j, a_j)\Big] \oplus o_t \tag{2}$$

where o_t is the observation of the tool calling result.

Our approach differs from standard interaction summarization (e.g., LangChain's ConversationSummaryBufferMemory and MemGPT's working context) (Packer et al., 2024). LangChain performs incremental summarization of the previous dialogue history, resulting in a single, growing summary paragraph. MemGPT utilizes a fixed-size block for unstructured text writable via function calls. In contrast:

- **Targeted Reasoning Compression:** : Our Reasoning Digests specifically target the reasoning traces rather than the general interaction history.
- **Structured Modularity:** Unlike a monolithic summary, our per-step digests remain distinct entities. This structured format not only *better aligns with pre-trained LLM behaviors*, but also mitigates the compounding risk of error propagation—a prevalent challenge in sequential and hierarchical LLM reasoning (Ma et al., 2026).

Standard SFT and RL typically treat a multi-turn trajectory as a single contiguous sequence. However, SWE-AGILE's inference imposes a strict dynamic visibility constraint: *while environmental history (observations and actions) remains fully visible, detailed reasoning traces (r_t) are transient.*

They are retained only within a sliding window (N steps) for cognitive continuity before being permanently replaced by digests (d_t). Training under the standard contiguous sequence assumption violates this dynamic context setting, as it allows the model to attend to all historical reasoning traces ($r_1, r_2, \dots$), which are explicitly hidden during inference. Therefore, applying such a dynamic context strategy brings the misalignment between training and inference. Implementing complex, dynamic attention masks to retroactively hide previous reasoning tokens is engineering-intensive, disrupting standard efficiency optimizations like FlashAttention (Dao et al., 2022), and introducing significant training overhead.

To align the training process with the inference process, we decompose each full trajectory τ of length T into a set of discrete Trajectory Snapshots $\mathcal{S} = \{(C_t, Y_t)\}_{t=1}^{T}$. Each snapshot represents the agent's specific "world view" at step t, optimized via a focused masking strategy:

- **Frozen Snapshot Context (Mask=0):** The input C_t (Eq. 2) simulates the runtime state where historical reasoning outside the sliding window is already compressed. These tokens are visible to attention but excluded from the loss, acting as a fixed prompt.
- **Active Target (Mask=1):** The target Y_t (Eq. 1) comprises the current reasoning, reasoning digest, and action. This is the sole learnable segment for the snapshot.

This decomposition ensures a rolling optimization: every reasoning trace r_t is optimized exactly once in the active target Y_t and subsequently serves as a compressed reasoning digest in future snapshots, effectively resolving the training-inference mismatch.

The necessity of the trajectory snapshots training can be also found in recent studies of context management. AgentFold (Ye et al., 2025) executes a "folding" operation, which manages its historical trajectory at multiple scales: it can perform granular condensations to preserve vital, fine-grained details, or deep consolidations to abstract away entire multi-step subtasks. Context-Folding (Sun et al., 2025) procedurally branches into a sub-trajectory to handle a subtask and then folds it upon completion, collapsing the intermediate steps while retaining a concise summary of the outcome. They both treat a trajectory as multiple training examples due to context modification.

2.2 Backfilling Reasoning and Digest

We further introduce a hindsight backfill pipeline to synthesize reasoning and digests based on existing successful trajectories to support the training of SWE-AGILE, inspired by ActRe (Yang et al., 2024c) and UI-TARS (Qin et al., 2025) on GUI agent that address the lack of explicit reasoning in GUI action traces through annotating intermediate "thoughts". However, unlike recorded GUI traces which are often purely action-driven, existing SWE trajectories typically contain shallow natural language responses. Our pipeline upgrades these sparse signals into explicit System-2 reasoning traces (r_t) and reasoning digests (d_t), bridging the gap between shallow heuristics and deep problem-solving.

The pipeline operates on a raw trajectory $\tau = \{(o_1, s_1, a_1), \dots, (o_T, s_T, a_T)\}$, where s_t is the shallow thought. We employ a strong reasoning model as the reasoning synthesizer to backfill the detailed reasoning traces of the trajectory step-by-step.

For each step t, the synthesizer generates the reasoning trace r_t and reasoning digest d_t conditioned on three critical inputs:

- **The Ground-Truth Action a_t:** Unlike standard inference, the model is provided with the future action, allowing it to backfill a CoT that inevitably leads to the correct decision.
- **The Semantic Hint s_t:** The original shallow thought is provided to preserve the agent's initial intent, ensuring the synthesized reasoning remains grounded in the trajectory's logic.
- **The Dynamic Context C_t:** We strictly simulate the inference-time visibility constraints defined in Eq. (2). It includes the full history of environmental interactions (observations and actions) but applies dynamic compression to reasoning: thoughts prior to the sliding window are replaced by digests d, while detailed thoughts r are retained only for the most recent N steps. Crucially, by exposing the immediate history of detailed thoughts, we enable cognitive continuity: the model builds incrementally upon its recent reasoning process rather than redundantly re-analyzing the global state at every turn.

Ultimately, each step is reformatted into a structured tuple (o_t, r_t, d_t, a_t), where the shallow thought s is replaced with detailed reasoning r and reasoning digest d.

We prioritize this backfilling strategy over direct RLVR or rejection sampling for three reasons:

- **Adaptive Reasoning Efficiency:** In the SWE task, the required depth of reasoning varies widely across steps. Some steps are routine operational actions (e.g., executing a script as previously planned or simple navigation) that require only surface-level intent verification. While other steps, such as analyzing a confusing error message, designing a new function structure or figuring out why a bug occurred, require sustained System-2 reasoning to handle ambiguity. We categorize them into Reflexive Steps and Deliberative Steps. Backfilling reasoning enables more controllable reasoning depth compared to raw trajectory collecting.
- **Format Enforcement:** Pre-trained models' generation often fails to strictly adhere to the "Reasoning $\rightarrow$ Digest $\rightarrow$ Action" format during multi-turn interactions. Backfilling reasoning and digest allow explicitly enforcing this format, creating a stable starting checkpoint for later RLVR.
- **Data Scalability:** Directly collecting new trajectories using our paradigm from scratch is computationally expensive due to the high cost of environment execution and the low success rate. To alleviate this issue, we leverage existing successful trajectories, allowing rapidly synthesizing the data by simply annotating gold trajectories and avoiding the need for extensive exploration.

2.3 Optimization via RLVR with Reasoning Compression

After SFT, we employ RLVR to optimize the policy. The objective is to increase the task success rate while decoupling reasoning depth from context cost: the agent should learn to expand its reasoning (r_t) sufficiently to solve complex problems (Deliberative Steps), while minimizing the permanent context via concise Reasoning Digests (d_t).

Trajectory-Level Compression Rate. We first introduce a metric to quantify the efficiency of context management. Let $|\tau_t|$ denote the token length of the complete interaction tuple (o_t, r_t, d_t, a_t) at step t. $\mathcal{L}_{full} = \sum_t |\tau_t|$ is the hypothetical total context size if the full history (including all reasoning r_t) were retained. $\mathcal{L}_{hybrid}$ is the actual context size under our dynamic policy, where r_t is pruned from the history outside the sliding window. The trajectory-level compression rate is defined as the global reduction ratio:

$$R_{comp} = 1 - \frac{\mathcal{L}_{hybrid}}{\mathcal{L}_{full}}. \tag{3}$$

This metric reflects the percentage of total context memory saved.

Reward Function. The overall reward $R(\tau)$ conditions efficiency on effectiveness:

$$R(\tau) = \mathbb{I}_{\text{success}} \cdot \Big(1 + \beta \cdot \min(R_{\text{comp}}(\tau), \gamma)\Big), \tag{4}$$

where $\mathbb{I}_{\text{success}} \in \{0, 1\}$ denotes task success, β denotes the weight of the compression reward, and γ is a clipping threshold. The clipping mechanism prevents the model from artificially bloating reasoning traces (r_t) merely to inflate the denominator of R_{comp} beyond the saturation point.

The multiplicative gating $\mathbb{I}_{\text{success}}$ ensures that compression rewards are added only on successful trajectories, effectively preventing the model from trading correctness for compression scores.

Global vs. Local Compression Rate: Handling Heterogeneous Step Complexity. A critical design choice is utilizing a global trajectory-level metric rather than an average of step-wise compression ratios (e.g., $\frac{1}{T}\sum(1 - \frac{len(d_t)}{len(r_t)})$). This design choice is driven by the categorization of Reflexive Step and Deliberative Steps of the SWE task described in Sec. 2.2:

- **Robustness to Reflexive Steps:** In Reflexive Steps, the necessary reasoning r_t is naturally brief, often resulting in a reasoning digest d_t of comparable length (i.e., $|r_t| \approx |d_t|$). This yields a near-zero local compression score. A step-wise objective would incentivize the model to "reward hack" by inflating r_t with redundant tokens during these simple steps merely to increase the denominator and improve the local ratio.
- **Incentivizing Depth in Deliberative Steps:** The global compression rate is tolerant of low compression in Reflexive Steps (which contribute minimally to the total sums). Instead, it drives the model to focus its optimization efforts on Deliberative Steps, where $|r_t|$ is large and the potential for substantial context saving ($|r_t| \rightarrow |d_t|$) is high.

Consequently, this optimization process establishes a dynamic balance between reasoning depth and context compression. By decoupling the transient reasoning overhead (generating r_t) from the

permanent context retention (storing d_t), the model learns an adaptive strategy: it behaves as a deep thinker during complex problem-solving moments to ensure $\mathbb{I}_{\text{success}}$, while acting as a concise distiller of its historical thoughts to maximize R_{comp}, thereby achieving competitive performance and token efficiency.

3 Experiment

3.1 Experiment Setup

Dataset. Our training pipeline consists of two stages utilizing distinct data sources. For the Cold-Start SFT phase, we use a high-quality subset of 2.2k trajectories from the SWE-Dev dataset (Wang et al., 2025a) (originally 19.3k). Since the original data lacks explicit System-2 reasoning, we apply our backfilling data synthesis pipeline (Sec. 2.2) to augment these trajectories using Qwen3-235B-A22B-Instruct-2507[1]. We further collect 200 trajectories via rejection sampling using Qwen3-235B-A22B-Thinking-2507[2]. This small batch utilizes the exact same paradigm and scaffold as the subsequent RLVR phase to minimize distribution shift.

We compare all methods on the SWE-Bench-Verified (Jimenez et al., 2024) benchmark, which evaluates AI systems on their ability to solve 500 software issues from 12 real world GitHub repositories.

Base Model. Unlike prior research (e.g., R2E-Gym (Jain et al., 2025) that utilizes the coding-specialized Qwen2.5-Coder (Hui et al., 2024b)), we select Qwen3 (Yang et al., 2025a) as our base model. This choice is driven by our research objective to explore the potential of SWE-AGILE. A base model with strong reasoning capabilities serves as a more suitable subject for this exploration than a model strictly optimized for coding capabilities without thinking abilities.

Implementation Details. Our agentic scaffold is built upon R2E-Gym. We make slight modifications on the prompt to enforce the proposed "Reasoning → Digest → Action" workflow. See more detail in Appendix A. For the RLVR phase, we utilize 896 diverse tasks from the R2E-Gym subset environment. At the SFT stage, detailed hyperparameters are provided in Appendix B. During the rollout phase of RLVR, we set the maximum number of steps to 50, and a strict maximum number of tokens every response generated to 4096 to avoid too verbal reasoning. Any rollout that triggers one of the conditions in max steps, max tokens per response, max context, trajectory timeout or submit will terminate. We use DAPO (Yu et al., 2025b) algorithm to optimize policy, and detailed hyperparameters are provided in Appendix B. We use XML-based tool calling format (see Appendix C for discussion).

Evaluation Settings. We limit the maximum number of steps to 60, the maximum number of context tokens to 65536. We run each evaluation 2 times and report the mean value of metrics.

3.2 Overall Performance

Tab. 1 summarizes the performance of SWE-AGILE compared to state-of-the-art open-source and closed-source models on SWE-Bench-Verified. From the results, we have the following observations.

Our method SWE-AGILE, utilizing the Qwen3-8B model, establishes a new performance standard for models in the 7B-8B parameter class. Starting from a general-purpose Qwen3-8B base, SWE-AGILE (SFT) achieves a success rate of **21.45%**, representing a substantial **35.5% relative improvement** over the base model (15.83%). This verifies the efficacy of SWE-AGILE paradigm in eliciting System-2 reasoning capabilities even in 8B models. Remarkably, SWE-AGILE (SFT) achieves this performance using only **2.2k training trajectories**, a mere **11%** of the 19.3k dataset utilized by SWE-Dev. With the integration of our compression-aware RLVR, SWE-AGILE further elevates the success rate to **24.05%**, outperforming all reported baselines of comparable size. Notably, despite being an 8B model, SWE-AGILE surpasses the Qwen3-based SkyRL-Agent-v0-14B (21.6%). Additionally, we explored the scalability of our method on larger models. While computational constraints limited our ability to perform full RLVR on a 14B model, applying our SFT pipeline to Qwen3-14B yielded a success rate of 30.06% on SWE-Bench-Verified, significantly surpassing existing 14B baselines. We also evaluated SWE-AGILE-8B on SWE-Bench Lite, where it achieved a success rate of 14.77%, outperforming comparable baselines such as SWE-smith-7B (11.7%) and

[1] https://huggingface.co/Qwen/Qwen3-235B-A22B-Instruct-2507

[2] https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507

R2E-Gym (11.0%).

3.3 Analysis of Dynamic Reasoning Context

We utilize the Qwen3-8B model to conduct a controlled analysis of reasoning context management and training paradigms. The results, detailing accuracy, average steps, and token consumption, are presented in Tab. 2.

Lost-in-the-Middle. Comparing Rows (2) and (3) confirms the "Lost-in-the-Middle" phenomenon discussed in the Sec 1. Retaining full reasoning history (Interleaved Thinking) significantly degrades performance (12.42%) compared to discarding it (Current-Step Thinking)(15.83%), despite the richer context. Additionally, we hypothesize that this degradation may also be partially related to the post-training process of Qwen3 models. On the contrary, while the average number of steps is more compared to Current-Step Thinking, SWE-AGILE paradigm ensures that the context length remains manageable. This allows the 8B model to handle long-horizon tasks without suffering from the "Lost-in-the-Middle" phenomenon.

Shallow Reasoning SFT Data Degrades Reasoning Capability. Paradoxically, SFT on original 2.2k SWE-Dev trajectories (Row 4) performs worse (14.83%) than the base model (15.83%). This indicates that training on data restricted to shallow reasoning traces actually constrains the model: it learns to align with the superficial heuristics of the dataset rather than leveraging its full pre-trained potential for deep problem-solving.

We decompose the model response into Reasoning, Textual Summary and Action. For our method, Textual Summary corresponds to the Reasoning Digest. For the Current-Step baseline, represents the standard Thought trace used to justify the action.

Efficiency of SWE-AGILE. To understand how SWE-AGILE achieves efficiency, we decompose models' average per-step response into Active Reasoning (r_t) and Textual Summary (d_t) in Fig. 3.

A key observation is that Current-Step Thinking generates the most verbose active reasoning ($\sim$1,075 tokens/step). Lacking access to enough information of the previous thought, the agent suffers from contextual amnesia and is forced to redundantly reconstruct the state from raw observations at every turn. In contrast, SWE-AGILE (SFT+RL) significantly reduces this load to $\sim$819.6 tokens/step (-28%). This confirms that our Reasoning Digests and Sliding Reasoning Window act as an effective cognitive cache, allowing the agent to perform *incremental reasoning* rather than redundant re-analysis.

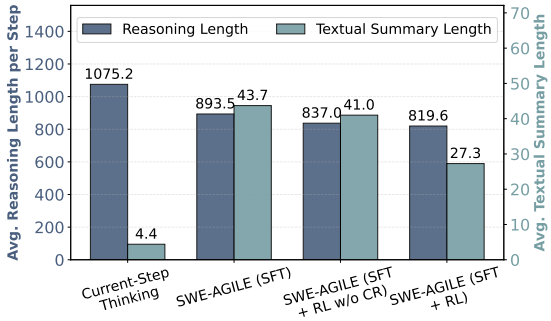

Figure 3: Average Response Length per Step.

While our current paradigm implicitly reduces redundant state reconstruction, a highly promising direction to strictly enforce this efficiency is to quantitatively monitor the reasoning content. By *calculating the embedding similarity between consecutive reasoning steps or employing an LLM-as-a-Judge*, future iterations can explicitly filter out repetitive SFT trajectories or design targeted RLVR penalties, pushing the boundary of cognitive efficiency even further.

Effect of Compression Reward. Comparing *RL w/o Compression Reward* and *RL*, we can see that the Compression Reward reduces the Textual Summary (d_t) by 33.4% ($41 \rightarrow 27.3$ tokens), while maintaining a comparable task success. As tool execution outputs also consume a large portion of the context window in agentic tasks, tool execution is essential to SWE tasks and the CR mechanism can be readily extended to compress verbose tool outputs. Although saving 13.7 tokens per step seems small in isolation, it represents a 33.4% relative reduction in the reasoning digest. When combined with tool output compression, the agent could have significantly more interaction turns before hitting context limits, thereby compressing more tokens in reasoning digests.

Overall, the above experimental results and findings demonstrate that SWE-AGILE successfully decouples reasoning depth from context cost, achieving the Pareto frontier of performance and efficiency.

4 Related Work

Recent advancements in LLMs have spurred the rapid development of autonomous agents across diverse domains, including GUI agents (Lian et al., 2025; Wang et al., 2026; Zhang et al., 2026; Wang

Approach	Base Model	Scaffold	Data	SUCCESS RATE
Closed Weight Models				
OpenHands (Wang et al., 2025d)	GPT-5	OpenHands	-	71.80
OpenHands	Claude 4 Sonnet	OpenHands	-	70.40
SWE-agent (Yang et al., 2024b)	Claude 4 Sonnet	SWE-agent	-	66.6
Agentless-1.5 (Xia et al., 2025a)	GPT-4o	Agentless	-	38.8
SWE-RL (Wei et al., 2025)	Llama-3.3-70B	Agentless Mini	-	41.0
Open Weight Models				
Lager than 32B				
Qwen3-Coder	Qwen3-Coder-480B	OpenHands	-	69.6
Kimi-K2	Kimi-K2-1T	OpenHands	-	65.4
GLM-4.5	GLM-4.5-355B	OpenHands	-	64.2
SWE-fixer (Xie et al., 2025)	Qwen2.5-72B	Pipline	-	32.8
Kimi-Dev (Yang et al., 2025c)	Qwen 2.5-72B	SWE-Agent	150B tokens +	48.6
CodeFuse-CGM (Tao et al., 2025)	Qwen2.5-72B	Graph RAG	200k issue-patch pairs	50.4
32B/30B				
SWE-Gym (Pan et al., 2025)	Qwen2.5-Coder-32B	OpenHands	-	20.6
R2EGym (Jain et al., 2025)	Qwen2.5-Coder-32B	R2EGym	3.3k trajectories	34.4
SWE-Dev (Wang et al., 2025a)	Qwen2.5-Coder-32B	OpenHands	19.3k trajectories	36.6
Skywork-SWE (Zeng et al., 2025)	Qwen2.5-Coder	OpenHands	8k trajectories	38.0
SWE-smith (Yang et al., 2025b)	Qwen2.5-Coder-32B	SWE-agent	5k trajectories	40.2
DeepSWE (Luo et al., 2025b)	Qwen3-32B	R2EGym	4.5k SWE tasks	42.2
ENTROPO-KTO (Yu et al., 2025a)	Qwen3-Coder-30B	R2EGym	-	49.3
14B				
SWE-Gym	Qwen2.5-Coder-14B	OpenHands	-	16.4
SkyRL-Agent-v0 (Cao et al., 2025)	Qwen3-14B	OpenHands	-	21.6
R2EGym	Qwen2.5-Coder-14B	R2EGym	3.3k trajectories	26.8
SWE-AGILE (SFT)	Qwen3-14B	R2EGym	2.2k trajectories	30.06
7B/8B				
SWE-Gym	Qwen2.5-Coder-7B	OpenHands	491 trajectories	10.6
SWE-smith	Qwen2.5-Coder-7B	SWE-agent	5k trajectories	15.2
R2EGym	Qwen2.5-Coder-7B	R2EGym	3.3k trajectories	19.0
SWE-Dev	Qwen2.5-Coder-7B	OpenHands	19.3k trajectories	23.4
Qwen3	Qwen3-8B	R2EGym	-	15.83
SWE-AGILE (SFT)	Qwen3-8B	R2EGym	2.2k trajectories	21.45
SWE-AGILE (SFT+RL)	Qwen3-8B	R2EGym	+ 896 SWE tasks	24.1

Table 1: Performance Comparison on SWE-Bench-Verified benchmark.

Method	**Avg Steps**	**Success Rate (%)**
Inference Baselines (Base Model)		
(1) Disable Thinking	43.95	0.03
(2) Interleavd Thinking	26.08	12.42
(3) Current-Step Thinking	15.77	15.83
SFT		
(4) Standard SFT (w/o Backfilling)	23.85	14.83
(5) **SWE-AGILE (SFT)**	21.00	21.45
RLVR		
(6) Standard SFT+RL	18.89	16.03
(7) **SWE-AGILE (SFT+RL w/o CR)**	22.66	23.45
(8) **SWE-AGILE (SFT+RL)**	24.59	24.05

Table 2: Ablation study on context management and training strategies. We report the Average Steps per trajectory (as a proxy for exploration depth) and the Success Rate on SWE-bench-Verified. w/o CR denotes the ablation setting where the *Compression Reward* is excluded during RLVR.

et al., 2025c; Gan et al., 2026; Chen et al., 2025), search agents (Tang et al., 2025; Team et al., 2026), optimization techniques for agent tasks acceleration (Huang et al., 2026), alongside various other emerging agentic frameworks (Yang et al., 2026; Wang et al., 2025b; Hu et al., 2026b; Zhu et al., 2026). In parallel with these advancements, researchers are increasingly focusing on the highly challenging, multi-turn domain of SWE Agent.

4.1 SWE Agent

Agentic SWE Scaffolds and Pipelines. Current approaches for automated software engineering broadly fall into two categories: agentic scaffolds and pipeline-based frameworks. Agentic scaffolds, such as SWE-agent (Yang et al., 2024b) and OpenHands (Wang et al., 2025d), empower LLMs to autonomously solve tasks by actively navigating repositories, editing files, and executing shell commands. To further enhance decision-making, SWE-Search (Antoniades et al., 2025) integrates Monte Carlo Tree Search (MCTS) into the agentic loop. Conversely, pipeline-based frameworks do use agentic paradigm, but applying structured, multi-stage workflows (Xia et al., 2025a; Yang et al.,

2025c; Tao et al., 2025; Wei et al., 2025). Beyond repository-level bug fixing, agentic frameworks are also effectively deployed for automated adversarial testing to expose code vulnerabilities (Shi et al., 2026).

Environment Curation. Addressing the scarcity of high-quality training trajectories and evaluation tasks, recent research focuses on proposing novel pipelines to create SWE environments (Pan et al., 2025; Jain et al., 2025; Wang et al., 2025a; Yang et al., 2025b; Zeng et al., 2025; Guo et al., 2026).

Training and Inference-Time Scaling Building on these data foundations, works such as SWE-Dev, Kimi-Dev (Yang et al., 2025c), ENTROPO (Yu et al., 2025a) and DeepSWE (Luo et al., 2025b) investigate post-training paradigms like Rejection Sampling Fine-tuning, DPO (Rafailov et al., 2023), and GRPO (Shao et al., 2024) to effectively enhance the capabilities of SWE Agents. Complementary to training, inference-time scaling strategies have proven effective for boosting performance during deployment. Approaches utilized in R2E-Gym, SWE-Dev, DeepSWE, Skywork-SWE and ENTROPO leverage inference-time scaling techniques such as verifier-guided Best-of-N selection to maximize the success rate of tasks.

4.2 CoT Compression

Recent advancements in LLMs, such as OpenAI o1 and DeepSeek-R1, have improved performance in System-2 reasoning domains like mathematics and programming by harnessing supervised fine-tuning and reinforcement learning techniques to enhance the Chain-of-Thought (CoT) reasoning. While appropriate CoT sequences improve performance, overlong CoT sequences may introduce significant computational overhead and attention dilution, and even trigger cognitive side-effects such as "narrative overfitting", where agents synthesize spurious causal relationships to force coherence (Sui et al., 2025; Hu et al., 2026a). (Arora and Zanette, 2025) trains models to produce minimal yet correct CoT by rewarding correctness while penalizing reasoning length to encourage efficient reasoning. TokenSkip (Xia et al., 2025b) enables LLMs to skip redundant tokens within CoTs with controllable compression ratio. O1-Pruner (Luo et al., 2025a) introduces the Length-Harmonizing Reward, combined with a PPO-style loss, to optimize reasoning LLMs by effectively shortening the CoT length. Without relying on a reference model, DAST (Shen et al., 2025) employs SimPO to fine-tune reasoning LLMs using a constructed length preference dataset.

Recent CoT compression techniques like LightThinker (Zhang et al., 2025) and InftyThink (Yan et al., 2025) focus on compressing the internal thinking process within a single turn. Differently, SWE-AGILE addresses the challenge of maintaining cognitive continuity across multi-turn environmental interactions. Hence, SWE-AGILE is orthogonal to those CoT compression techniques and can be deployed in conjunction with them.

5 Conclusion

In this paper, we introduce SWE-AGILE, a framework designed to reconcile reasoning depth with context constraints in the long-horizon SWE task. By integrating a Dynamic Reasoning Context supported by trajectory snapshot training and compression-aware RLVR, SWE-AGILE enables agents to leverage explicit System-2 reasoning while preventing context explosion.

Crucially, by effectively decoupling transient reasoning overhead from permanent context retention, this framework lays a solid groundwork for explicitly identifying and minimizing redundant re-analyzing as discussed in Efficiency of SWE-AGILE in Sec. 3 , thereby opening new avenues for optimizing agent cognitive efficiency in future research.

Limitation

In current version of SWE-AGILE, the size of the sliding reasoning window is set to a random integer between $[2, 5]$ during backfilling, SFT, RLVR and inference. Although this setting reveals the robustness of SWE-AGILE, more analysis of the size of the sliding reasoning window remains unexplored.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (No. 2025YFE0113500), the National Science Fund for Distinguished Young Scholars (No. 62525605), and the National Natural Science Foundation of China (No. 62572410, No. 62272401, and No. U22B2051).

References

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Yang Wang. 2025. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *ICLR*. https://openreview.net/forum?id=G7sIFXugTX.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *arXiv Preprint*. https://arxiv.org/abs/2502.04463.

Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, and et al. 2025. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning.

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, and et al. 2025. Gui-shepherd: Reliable process reward and verification for long-sequence gui tasks. *Preprint*, arXiv:2509.23738.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv Preprint*. https://arxiv.org/abs/2501.12948.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, and et al. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv Preprint*. https://arxiv.org/abs/2512.02556.

Guo Gan, Yuxuan Ding, Cong Chen, Yuwei Ren, Yin Huang, and Hong Zhou. 2026. Android coach: Improve online agentic training efficiency with single state multiple actions. *Preprint*, arXiv:2604.07277.

Lianghong Guo, Yanlin Wang, Caihua Li, Wei Tao, Pengyu Yang, Jiachi Chen, Haoyu Song, Duyu Tang, and Zibin Zheng. 2026. Swe-factory: Your automated factory for issue resolution training data and evaluation benchmarks. *Preprint*, arXiv:2506.10954.

Jinwei Hu, Xinmiao Huang, Youcheng Sun, Yi Dong, and Xiaowei Huang. 2026a. Lying with truths: Open-channel multi-agent collusion for belief manipulation via generative montage. *Preprint*, arXiv:2601.01685.

Junan Hu, Shudan Guo, Wenqi Liu, and et al. 2026b. Context-agent: Dynamic discourse trees for non-linear dialogue. *Preprint*, arXiv:2604.05552.

Haoyu Huang, Jinfa Huang, Zhongwei Wan, and et al. 2026. Speceyes: Accelerating agentic multimodal llms via speculative perception and planning. *Preprint*, arXiv:2603.23483.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, and et al. 2024a. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, and et al. 2024b. Qwen2.5-coder technical report. *arXiv Preprint*. https://arxiv.org/abs/2409.12186.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, and et al. 2024. Openai o1 system card. *arXiv Preprint*. https://arxiv.org/abs/2412.16720.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. 2025. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *arXiv Preprint*. https://arxiv.org/abs/2504.07164.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? https://openreview.net/forum?id=VTF8yNQM66.

Yuqi Li, Zijie Zhou, Zhiyuan Peng, Junhao Dong, Haochen You, Renye Yan, Shiping Wen, Yingli Tian, and Tingwen Huang. 2025a. A preference-driven methodology for efficient code generation. *IEEE Transactions on Artificial Intelligence*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, and et al. 2025b. From system 1 to system 2: A survey of reasoning large language models. *arXiv Preprint*. https://arxiv.org/abs/2502.17419.

Shuquan Lian, Yuhang Wu, Jia Ma, Yifan Ding, Zihan Song, Bingqi Chen, Xiawu Zheng, and Hui Li. 2025. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding. *Preprint*, arXiv:2507.22025.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv Preprint*. https://arxiv.org/abs/2501.12570.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, and et al. 2025b. Deepswe: Training a state-of-the-art coding agent from scratch by scaling rl. Notion Blog.

Guoqi Ma, Liang Zhang, Hongyao Tu, and et al. 2026. Hcre: Llm-based hierarchical classification for cross-document relation extraction with a prediction-then-verification strategy. *Preprint*, arXiv:2604.07937.

MiniMaxAI. 2025. Minmax m2. https://huggingface.co/MiniMaxAI/MiniMax-M2.

Charles Packer, Sarah Wooders, Kevin Lin, and el al. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. 2025. Training software engineering agents and verifiers with swe-gym. In *ICML*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, and et al. 2025. UI-TARS: pioneering automated GUI interaction with native agents. *arXiv Preprint*. https://arxiv.org/abs/2501.12326.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, and et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv Preprint*. https://arxiv.org/abs/2402.03300.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. DAST: difficulty-adaptive slow-thinking for large reasoning models. https://arxiv.org/abs/2503.04472.

Jingwei Shi, Xinxiang Yin, Jing Huang, Jinman Zhao, and Shengyu Tao. 2026. Codehacker: Automated test case generation for detecting vulnerabilities in competitive programming solutions. *Preprint*, arXiv:2602.20213.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Trans. Mach. Learn. Res.*, 2025.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. 2025. Scaling long-horizon LLM agent via context-folding. *arXiv Preprint*. https://arxiv.org/abs/2510.11967.

Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, and et al. 2025. Beyond turn limits: Training deep search agents with dynamic context window. *Preprint*, arXiv:2510.08276.

Hongyuan Tao, Ying Zhang, Zhenhao Tang, Hongen Peng, Xukun Zhu, Bingchang Liu, Yingguang Yang, Ziyin Zhang, Zhaogui Xu, Haipeng Zhang, Linchao Zhu, Rui Wang, Hang Yu, Jianguo Li, and Peng Di. 2025. Code graph model (CGM): A graph-integrated large language model for repository-level software engineering tasks. *arXiv Preprint*. https://arxiv.org/abs/2505.16901.

MiroMind Team, S. Bai, L. Bing, L. Lei, R. Li, X. Li, X. Lin, and et al. 2026. Mirothinker-1.7 & h1: Towards heavy-duty research agents via verification. *Preprint*, arXiv:2603.15726.

Haoran Wang, Zhenyu Hou, Yao Wei, Jie Tang, and Yuxiao Dong. 2025a. Swe-dev: Building software engineering agents with training and inference scaling. In *ACL (Findings)*, pages 3742–3761.

Wenhao Wang, Peizhi Niu, Zhao Xu, Zhaoyu Chen, and et al. 2025b. Mcp-flow: Facilitating llm agents to master real-world, diverse and scaling mcp tools. *Preprint*, arXiv:2510.24284.

Wenhao Wang, Zijie Yu, Rui Ye, and et al. 2025c. Fedmabench: Benchmarking mobile agents on decentralized heterogeneous user data. *Preprint*, arXiv:2503.05143.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, and et al. 2025d. Openhands: An open platform for AI software developers as generalist agents. *ICLR*. https://openreview.net/forum?id=OJd3ayDDoF.

Zezhou Wang, Ziyun Zhang, Xiaoyi Zhang, Zhuzhong Qian, and Yan Lu. 2026. From off-policy to on-policy: Enhancing gui agents via bi-level expert-to-policy assimilation. *Preprint*, arXiv:2601.05787.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. 2025. SWE-RL: advancing LLM reasoning via reinforcement learning on open software evolution. *arXiv Preprint*. https://arxiv.org/abs/2502.18449.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2025a. Demystifying llm-based software engineering agents. *Proc. ACM Softw. Eng.*, 2(FSE):801–824.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025b. TokenSkip: Controllable chain-of-thought compression in LLMs. In *EMNLP*, pages 3351–3363.

Chengxing Xie, Bowen Li, Chang Gao, He Du, Wai Lam, Difan Zou, and Kai Chen. 2025. Swe-fixer: Training open-source llms for effective and efficient github issue resolution. In *ACL (Findings)*, pages 1123–1139.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. 2025. Inftythink: Breaking the length limits of long-context reasoning in large language models. *arXiv Preprint*. https://arxiv.org/abs/2503.06692.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, and et al. 2025a. Qwen3 technical report. *arXiv Preprint*. https://arxiv.org/abs/2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, and et al. 2024a. Qwen2.5 technical report. *arXiv Preprint*. https://arxiv.org/abs/2412.15115.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024b. Swe-agent: Agent-computer interfaces enable automated software engineering. In *NeurIPS*.

John Yang, Kilian Leret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. 2025b. Swe-smith: Scaling data for software engineering agents. https://arxiv.org/abs/2504.21798.

Shuo Yang, Caren Han, Yihao Ding, Shuhe Wang, and Eduard Hovy. 2026. Tooltree: Efficient LLM tool planning via dual-feedback monte carlo tree search and bidirectional pruning. In *The Fourteenth International Conference on Learning Representations*.

Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. 2024c. React meets actre: Autonomous annotation of agent trajectories for contrastive self-training. https://openreview.net/forum?id=0VLBwQGWpA.

Zonghan Yang, Shengjie Wang, Kelin Fu, Wenyang He, Weimin Xiong, Yibo Liu, and et al. 2025c. Kimi-dev: Agentless training as skill prior for swe-agents. *arXiv Preprint*. https://arxiv.org/abs/2509.23045.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *ICLR*. https://openreview.net/forum?id=JvkuZZ04O7.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, and et al. 2025. Agentfold: Long-horizon web agents with proactive context management. *arXiv Preprint*. https://arxiv.org/abs/2510.24699.

Jiahao Yu, Zelei Cheng, Xian Wu, and Xinyu Xing. 2025a. Building coding agents via entropy-enhanced multi-turn preference optimization. *arXiv Preprint*. https://arxiv.org/abs/2509.12434.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, and et al. 2025b. DAPO: an open-source LLM reinforcement learning system at scale. *arXiv Preprint*. https://arxiv.org/abs/2503.14476.

Liang Zeng, Yongcong Li, Yuzhen Xiao, Changshi Li, and et al. 2025. Skywork-swe: Unveiling data scaling laws for software engineering in llms. *arXiv Preprint*. https://arxiv.org/abs/2506.19290.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. 2025. LightThinker: Thinking step-by-step compression. In *EMNLP*, pages 13318–13339.

Hyperparameter	**Value**
Learning Rate	1×10^{-6}
Global Batch Size	16
Mini-Batch Size	8
Generations per Prompt (G)	8
Max Prompt Length	28,582
Max Response Length	4096
Clip Ratio (Low / High)	0.2 / 0.28
KL Coefficient	0.0
Compression Reward Weight β	0.2
Compression Clipping threshold γ	0.55
Temperature	1.0
Repetition Penalty	1.15
Total Epochs	1

Table 3: Hyperparameters for RLVR (DAPO) Training

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, and et al. 2026. Infiniteweb: Scalable web environment synthesis for gui agent training. *Preprint*, arXiv:2601.04126.

Wei Zhu, Zhiwen Tang, and Kun Yue. 2026. Symphony: Synergistic multi-agent planning with heterogeneous language model assembly. *Preprint*, arXiv:2601.22623.

A Scaffold

Following R2E-Gym, we use four tools to enable the agent to perform diverse SWE tasks; 1) file editor: for viewing and editing files, 2) search tool: for searching a relevant term in a given file or folder, 3) execute bash: allowing execution of non-interactive bash commands (e.g., for running test scripts), 4) submit: for ending the current trajectory while returning expected outputs.

SWE-Dev trajectories use only three tools, which are basically the same R2E-Gym but lack 2) search tool, although using the bash tool can basically achieve the same effect of a search tool. Therefore, we further collect about 200 trajectories on tasks from R2E-Gym using the four tools to supplement SFT data.

The detailed prompts are provided in Fig. 4 and Fig. 5.

B Detailed Hyperparameters

Tabs. 3 and 4 provide more detailed hyperparameters.

C Tool Calling Format

Standard JSON-based tool calling poses significant robustness challenges in the SWE task. Since tool

Hyperparameter	Value
Learning Rate	1×10^{-5}
Batch Size	32
Max Sequence Length	26000 tokens
Total Epochs	4
Sliding Window Size	random in [2, 5]

Table 4: Hyperparameters for Snapshots Training

arguments often include code snippets containing strings and special characters, encapsulating this content within a JSON structure requires complex, multi-level escaping. This complexity frequently leads to syntax errors, particularly when model capabilities are limited. To mitigate these nesting and escaping issues, we adopt an XML-based tool calling format, which allows for more robust parsing of raw code content.

System Prompt for SWE-AGILE

You are a programming agent who is provided a github issue and repository bash environment and is tasked to solve certain tasks (e.g., file localization, testcase generation, code repair and editing etc) to resolve the issue.

We have access to the following functions:

—— BEGIN FUNCTION #1: file_editor ——
Description:
Custom editing tool for viewing, creating and editing files

[Skip details. See the full prompt in our GitHub repository.]

—— END FUNCTION #4 ——

Your every response MUST follow a precise three-part structure:

1. **reasoning (`<think>`):** Use this space to analyze observations, debate potential causes, and plan the next step. **Note:** The length and depth of this section should adjust dynamically based on the task complexity (see "Adaptive Reasoning Depth" below).
2. **reasoning_digest:** A compressed summary of your current thought and intent inside `<reasoning_digest>...</reasoning_digest>` tags.
3. **action:** The tool call using the specified XML format.

[Skip details. See the full prompt in our GitHub repository.]

<IMPORTANT>
1. Context
* **Transient `<think>`:** Only some recent `<think>` blocks is visible to you in the next turn. Old thoughts vanish.
* **Persistent `<reasoning_digest>`:** All `<reasoning_digest>` blocks remain in history forever.

2. Adaptive Reasoning Depth
*** **Complex Reasoning:** Use **deep, detailed, and exploratory** reasoning when the step involves uncertainty, diagnosis, or design.**
*** *Examples:* Analyzing a confusing error message, designing a new function structure, figuring out why a bug occurred, or deciding a complex test strategy.**
*** *Instruction:* Break down the logic step-by-step. But do **not** make too many assumptions and do **not** be too verbose.**
*** **Routine Execution:** Use **concise** reasoning when the step is deterministic, mechanical, or part of an already-made plan.**
*** *Examples:* Executing a script you just decided to run or simple navigation.**
*** *Instruction:* Do not over-analyze. State your intent directly (e.g., "Executing the test script as planned") and verify the action.**

3. Continuity
*** **Bridge the Gap:** Start your `<think>` by explicitly connecting to the previous reasoning and the execution result of the last step, building a logical bridge to your next action.**
*** **No Redundancy:** Do not re-state the overall project goal or background info if not necessary. Do not re-analyze content that was already covered in previous reasoning steps unless an error or unexpected result necessitates re-reanalyzing.**

[Skip details. See the full prompt in our GitHub repository.]

Figure 4: The System Prompt for SWE-AGILE. Coherent with Backfilling Prompt, it helps reducing redundant re-analyzing.

System Prompt for SWE-AGILE

You are a programming agent who is provided a github issue and repository bash environment and is tasked to solve certain tasks (e.g., file localization, testcase generation, code repair and editing etc) to resolve the issue.

We have access to the following functions:

—- BEGIN FUNCTION #1: file_editor —-
Description:
Custom editing tool for viewing, creating and editing files

[Skip details. See the full prompt in our GitHub repository.]

—- END FUNCTION #4 —-

Your every response MUST follow a precise three-part structure:

1. **reasoning (`<think>`):** Use this space to analyze observations, debate potential causes, and plan the next step. **Note:** The length and depth of this section should adjust dynamically based on the task complexity (see "Adaptive Reasoning Depth" below).
2. **reasoning_digest:** A compressed summary of your current thought and intent inside `<reasoning_digest>...</reasoning_digest>` tags.
3. **action:** The tool call using the specified XML format.

[Skip details. See the full prompt in our GitHub repository.]

<IMPORTANT>
1. Context
* **Transient `<think>`:** Only some recent `<think>` blocks is visible to you in the next turn. Old thoughts vanish.
* **Persistent `<reasoning_digest>`:** All `<reasoning_digest>` blocks remain in history forever.

2. Adaptive Reasoning Depth
*** **Complex Reasoning:** Use **deep, detailed, and exploratory** reasoning when the step involves uncertainty, diagnosis, or design.**
*** *Examples:* Analyzing a confusing error message, designing a new function structure, figuring out why a bug occurred, or deciding a complex test strategy.**
*** *Instruction:* Break down the logic step-by-step. But do **not** make too many assumptions and do **not** be too verbose.**
*** **Routine Execution:** Use **concise** reasoning when the step is deterministic, mechanical, or part of an already-made plan.**
*** *Examples:* Executing a script you just decided to run or simple navigation.**
*** *Instruction:* Do not over-analyze. State your intent directly (e.g., "Executing the test script as planned") and verify the action.**

3. Continuity
*** **Bridge the Gap:** Start your `<think>` by explicitly connecting to the previous reasoning and the execution result of the last step, building a logical bridge to your next action.**
*** **No Redundancy:** Do not re-state the overall project goal or background info if not necessary. Do not re-analyze content that was already covered in previous reasoning steps unless an error or unexpected result necessitates re-reanalyzing.**

[Skip details. See the full prompt in our GitHub repository.]

Figure 5: The Prompt for Backfilling Reasoning and Digest. Coherent with Agent System Prompt, it helps reducing redundant re-analyzing.