

DISCOVERING HIERARCHICAL SOFTWARE ENGINEERING AGENTS VIA BANDIT OPTIMIZATION

000
001
002
003
004
005 **Anonymous authors**
006 Paper under double-blind review
007
008
009
010

ABSTRACT

011 Large language models (LLMs) are increasingly applied to software engineering
012 (SWE), but they struggle on real-world tasks that are long-horizon and often out
013 of distribution. Current systems typically adopt monolithic designs where a sin-
014 gle model attempts to interpret ambiguous issues, navigate large codebases, and
015 implement fixes in one extended reasoning chain. This design makes it difficult
016 to generalize beyond training data. Inspired by how human engineers decompose
017 problems into sub-tasks, we argue that SWE agents should be structured as or-
018 chestrators coordinating specialized sub-agents, each responsible for a specific
019 sub-task such as bug reproduction, fault localization, code modification, or valida-
020 tion. The central challenge is how to design these hierarchies effectively. Manual
021 decompositions follow human workflows but often mismatch LLM capabilities,
022 while automated search methods such as evolutionary strategies require evaluating
023 a very large number of candidates, making them prohibitively expensive for SWE.
024 We show that formulating hierarchy discovery as a multi-armed bandit problem
025 enables efficient exploration of sub-agent designs under limited budgets. On SWE-
026 bench-Verified, this approach outperforms single-agent systems and manually
027 designed multi-agent systems. On SWE-bench-Live, which features recent and
028 out-of-distribution issues, our system ranks 2nd on the leaderboard with a 36B
029 model, surpassing larger systems such as GPT-4 and Claude. This provides the
030 first evidence that hierarchical multi-agent systems improves generalization on
031 challenging long-horizon SWE tasks.

1 INTRODUCTION

032 Large language models (LLMs) have achieved remarkable progress in natural language processing
033 [41] and reasoning [18], and are increasingly adopted in solving complex coding problems [61].
034 Yet solving real-world software engineering (SWE) problems remains challenging [24] for LLMs,
035 particularly for issues that fall outside the training distribution [58]. Despite strong results on SWE-
036 bench-Verified [24], state-of-the-art systems struggle on more recent and out-of-distribution issues in
037 SWE-bench-Live [58].

038 One possible cause is the long-horizon nature of SWE tasks: Current LLM agents typically rely on
039 a single model to interpret underspecified problem statements, navigate large and interdependent
040 codebases, and carry out all sub-tasks—reproducing the bug, localizing the fault, editing the code,
041 and validating the fix—within *one* extended reasoning chain. This monolithic design hinders gen-
042 eralization: long contexts dilute attention and reduce retrieval accuracy [38; 39; 21], while jointly
043 solving all sub-tasks prevents modularity and hinders robustness [59; 60].

044 Inspired by how human engineers approach complex problems, we posit that explicit hierarchy can
045 help LLM-based agents manage long workflows. Cognitive science shows that people reduce mental
046 effort by decomposing tasks into smaller sub-tasks [31; 40; 33]; in software engineering, this typically
047 involves bug reproduction, fault localization, code modification, and validation [49; 23; 29]. This
048 same idea appears as temporal abstraction in hierarchical reinforcement learning (HRL) [10]: instead
049 of solving everything step by step, problems are handled by delegating to reusable sub-agents, each
050 defined as a policy for a specific sub-task [44]. An orchestrator coordinates these sub-agents by
051 choosing which one to activate, and each runs until it finishes its sub-task [44; 16; 9]. By planning
052 with sub-agents rather than individual steps, the orchestrator shortens the reasoning horizon, which

054 reduces distraction from irrelevant details and isolates reusable patterns. This not only makes complex
 055 problems more manageable but also improves generalization, since effective sub-agents can be reused
 056 across different tasks and contexts.

057 Approaches to hierarchical multi-agent system design range from manual to automated. At one
 058 end, engineers manually design decompositions, specifying sub-tasks and sub-agents that follow
 059 human workflows [8; 35; 11]. These designs require substantial effort, and workflows that are natural
 060 for humans do not necessarily translate into effective performance on LLM agents as shown in
 061 our experiments (Section 5). At the other end, automated methods such as evolutionary search
 062 generate designs automatically [56; 25; 22], but require evaluating large numbers of candidates. This
 063 is practical in domains with cheap evaluation, such as multi-hop question answering [53; 30], but
 064 infeasible for software engineering (SWE). In SWE, evaluations are *expensive* and *long-horizon*:
 065 validating a single design can take up to an hour, requiring multi-step sandboxed runs and full
 066 integration tests. The difficulty of *credit assignment* [42; 36] makes this even worse: determining
 067 which sub-agents actually contributed to success would typically require extensive sampling, which
 068 is prohibitively costly.

069 To address these challenges, we draw inspiration from multi-armed bandit (MAB) [14; 1; 17; 54] and
 070 formulate the design of hierarchical multi-agent systems as a sequential decision-making process. We
 071 term our method *Bandit Optimization for Agent Design (BOAD)*. In MAB problems, a learner must
 072 identify the best arm from a pool with limited performance queries, where outcomes are stochastic.
 073 The learner balances exploration (testing new or uncertain arms) with exploitation (selecting the
 074 arm with the highest observed reward so far). In our setting, each arm corresponds to a sub-agent
 075 design (i.e., an agent prompt). We first optimize sub-agents and then fix them before deriving an
 076 orchestrator design using LLM prompting, since jointly optimizing both would be a costly bi-level
 077 optimization problem [13]. Typical evolutionary algorithms [20] generate candidate designs, evaluate
 078 them, discard them, and propose new ones by mutating the top performers. This process is wasteful:
 079 useful sub-agents may be discarded and rarely rediscovered because evolution is stochastic. Instead,
 080 we archive all generated designs, and adaptively choose promising combinations to evaluate next.
 081 This ensures efficient reuse of past designs and increases the likelihood of retaining strong sub-agents.
 082 To tackle the credit assignment problem, we go beyond binary success signals of entire sub-agent
 083 sets. We use LLM-as-a-judge [27] to assess whether individual sub-agents contributed meaningfully
 084 within a trajectory and use these “helpfulness” scores as rewards for the model selection algorithm.

085 We evaluate our BOAD on SWE-bench-Verified [24], a benchmark for software engineering tasks
 086 grounded in real GitHub issues. On SWE-bench-Verified, our method consistently outperforms single-
 087 agent systems and manually designed multi-agent systems. On SWE-bench-Live [28], which includes
 088 more recently collected issues and presents out-of-distribution challenges, our system achieves **2nd**
 089 **place** on the leaderboard using a 36B model—outperforming larger-scale systems based on Claude
 090 and GPT-4. To our best knowledge, our work is the first to show generalization improvements using
 091 automatically discovered hierarchical multi-agent systems on challenging long-horizon interactive
 092 tasks like SWE-bench.

093 2 RELATED WORKS

094 **Meta-agent design.** Recent research has explored automatically designing agent organizations
 095 to reduce reliance on human intuition. Zhang et al. [56]; Hu et al. [22] propose frameworks for
 096 workflow generation and system-level automation, while Kim et al. [25] use evolutionary strategies
 097 for self-referential prompt refinement. Chen et al. [12] dynamically constructs role-based agents and
 098 coordinates them per task, though these roles are ephemeral and not reusable across settings. Other
 099 approaches, such as Misaki et al. [32], improve inference-time compute allocation by adaptively
 100 deciding whether to explore new candidates or refine existing ones. However, a common limitation
 101 of these meta-agent design methods is their reliance on frequent evaluations to guide search. They
 102 perform well when feedback is cheap and abundant (e.g., reasoning benchmarks or simple QA
 103 tasks), but become impractical in software engineering settings, where each candidate often requires
 104 sandboxed execution or full unit testing. In contrast, our method evolves reusable sub-agents and an
 105 orchestrator, reusing effective components to reduce redundant evaluations and maintain efficiency
 106 even under expensive feedback conditions.

108 **Evolutionary strategies for LLMs.** Evolutionary strategies has been applied broadly to improve
 109 LLM-based systems, such as algorithm discovery via code mutation [34] and self-rewriting agents
 110 that iteratively mutate their own source code [55]. Evolutionary strategies have also been widely
 111 explored for prompt optimization: Kim et al. [25] use co-evolutionary refinement, Guo et al. [19]
 112 apply genetic search to discrete prompt tokens, and Agrawal et al. [2] leverage reflection with Pareto-
 113 based selection for sample-efficient instruction tuning. These methods highlight how evolution can
 114 reduce human effort and uncover novel strategies, but they typically rely on abundant, inexpensive
 115 evaluations. In contrast, our method targets costly, long-horizon SWE tasks by maintaining a reusable
 116 archive of sub-agents and casting sub-agent selection as a multi-armed bandit problem, improving
 117 sample efficiency while preserving the benefits of hierarchical organization.

118 **Multi-agent systems for SWE.** Prior work has explored multi-agent designs for software engineering
 119 tasks. Arora et al. [8]; Phan et al. [35]; Chen et al. [11] adopt modular or graph-based pipelines,
 120 where subtasks and agent roles are manually defined. While structured, such pipelines require
 121 heavy engineering and often fail to generalize. Other works exploit agent diversity through fixed
 122 coordination schemes. Li et al. [28] engages specialized agents in a multi-round debate over candidate
 123 bug-fix plans before producing the final repair, while Zhang et al. [57] introduce a meta-policy that
 124 aggregates the code patch from multiple agents and identifies the most promising solution through
 125 re-ranking. Although effective, these methods rely on predetermined pipelines or coordination rules,
 126 which limit flexibility and adaptability. In contrast, our method learns hierarchical multi-agent
 127 system automatically by jointly optimizing sub-agents and an orchestrator, avoiding manual task
 128 decomposition and fixed ensemble strategies.

129 3 PRELIMINARIES

131 **Software engineering agents** We study the problem of using LLMs to resolve real-world GitHub
 132 issues, where each issue consists of a textual description and a corresponding code repository. Since
 133 issues are not self-contained, solving them requires identifying and modifying relevant parts of the
 134 codebase. In this work, we focus exclusively on agentic methods [52], where an LLM interacts with
 135 a runtime environment through tool use. Such agents can browse files, execute shell commands, run
 136 tests, and edit code directly, giving them the flexibility to tackle long-horizon tasks end-to-end.

138 **Markov Decision Process (MDP)** We model agent–environment interaction as a finite-horizon
 139 Markov decision process (MDP) [37], $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, H)$. At each step t , the agent observes
 140 a state $s_t \in \mathcal{S}$, consisting of the issue description x and the history of prior tool interactions
 141 $h_{t-1} = (a_1, o_1, \dots, a_{t-1}, o_{t-1})$. The agent samples an action $a_t \in \mathcal{A}$ from its policy $\pi(a_t | s_t)$,
 142 where \mathcal{A} includes all available tools and commands. Executing a_t yields an observation $o_t \in \mathcal{O}$ (e.g.,
 143 logs, diffs, or test results), updating the state to s_{t+1} . A trajectory $\tau = (s_0, a_0, \dots, s_T, y)$ terminates
 144 at $T \leq H$ when the agent submits a patch y (forced at $T = H$ if none is submitted earlier). Rewards
 145 are sparse:

$$146 \quad r(s_t, a_t) = 0 \text{ for } t < T, \quad r(s_T, a_T) = \begin{cases} 1 & \text{if } y \text{ passes all tests,} \\ 0 & \text{otherwise.} \end{cases}$$

148 The agent’s goal is to maximize the expected rewards $J(\pi) = \mathbb{E}_{\tau \sim \pi}[r(s_T, a_T)]$. With sparse rewards
 149 and long horizons, discovering successful trajectories is challenging.

151 **Temporal Abstraction via Semi-MDP (SMDP)** Semi-Markov Decision Process (SMDP) frame-
 152 work [43] is widely used to mitigate long-horizon sparse reward problems. Instead of issuing primitive
 153 actions $a_t \in \mathcal{A}$, the orchestrator selects a temporally extended action (option) $\omega_t \in \Omega$. Each option
 154 corresponds to a sub-agent that executes a sequence of actions (a_t, \dots, a_{t+m-1}) until termination,
 155 after which control returns at s_{t+m} , where m denotes the duration of an sub-agent. This reduces
 156 decision frequency and simplifies planning.

157 **Multi-Armed Bandit (MAB)** Multi-armed bandit (MAB) [26] is a special case of an MDP with a
 158 single state and no transitions. At each round t , the learner selects an arm $a_t \in \mathcal{A}$, receives a stochastic
 159 reward $r_t \in [0, 1]$ drawn from an unknown distribution, and seeks to maximize the cumulative reward
 160 $\sum_{t=1}^B r_t$ over a fixed interaction budget B . The MAB framework captures decision-making under
 161 uncertainty when only a limited number of trials are available.

162 4 METHOD: BANDIT OPTIMIZATION FOR AGENT DESIGN (BOAD)
163164 Our goal is to automatically discover a set of K sub-agents $\Omega = \{\omega_1, \dots, \omega_K\}$ and an orchestrator
165 π that maximizes the expected reward of solving issues. A naive approach is to use evolutionary
166 search over (π, Ω) :
167

168
$$\max_{\pi, \Omega} \mathbb{E}_{x \sim \mathcal{D}_{\text{design}}, \tau \sim \pi} [r(s_T, a_T)], \quad (1)$$

169

170 where $\mathcal{D}_{\text{design}}$ is a *design set* consisting of example problems. In this paper, both the sub-agent and
171 the orchestrator agent are parameterized by their prompt. However, this requires generating many
172 full trajectories τ and repeatedly querying the reward function r , which is prohibitively expensive.
173174 **Agent design as multi-armed bandit:** We formulate the discovery of orchestrators and sub-agents as
175 a multi-armed bandit (MAB) problem [26], where each arm corresponds to a particular sub-agent and
176 at every round t , K sub-agents are chosen. This framing directs more evaluations toward promising
177 designs while continuing to explore new ones, reducing wasted trajectories on poor candidates.
178 Consequently, it makes automatic discovery of multi-agent systems tractable despite the high cost of
179 evaluations in SWE. A detailed formulation is given in Section 4.1. However, this direct approach
faces two challenges.
180181 1. The space of possible orchestrators and sub-agent sets is extremely large and initially unknown,
182 making it impractical to enumerate arms in advance.
183 2. Even if we evaluate a sub-agent along with an orchestrator and the other sub-agents, credit
184 assignment is ambiguous: some sub-agents may succeed only by “free-riding” on others, so the
185 observed reward does not necessarily reflect their individual contribution.
186187 Next, we illustrate how we tackle these challenges in the following sections.
188189 4.1 AGENT DESIGN AS A MULTI-ARMED BANDIT PROBLEM
190191 The space of orchestrator–subagent pairs (π, Ω) is vast and infeasible to enumerate. To make the
192 search tractable, we maintain an archive Γ of candidate sub-agents. Instead of treating each subset
193 of sub-agents Ω as an arm, we treat each sub-agent $\omega \in \Gamma$ as an arm, enabling information sharing
194 across different sub-agents (will be explained below). At each round t , the algorithm selects a subset
195 $\Omega_t \subseteq \Gamma$ by choosing K arms, instantiates an orchestrator π_t for Ω_t , evaluates (π_t, Ω_t) on example
196 problems from a *design* set, and propagates feedback to all participating sub-agents. Feedback
197 is calculated for each sub-agent independently to tackle the credit assignment problem (details in
198 4.2). Because sub-agents appear in multiple subsets, each evaluation, even if unsuccessful, provides
199 signal for multiple subsets at the same time. This formulation supports efficient credit assignment,
200 reduces redundant exploration, and progressively refines estimates u_ω for each $\omega \in \Gamma$. Algorithm 1
summarizes the procedure, with further details provided below.
201202 **Bootstrapping a sub-agent archive** We begin with an initial archive Γ_0 of candidate sub-agents.
203 This archive is generated by prompting an LLM with the template in Appendix A.1.2. However,
204 simply generating sub-agents is insufficient because the orchestrator may not know how to invoke
205 them. We present sub-agents as tools to the orchestrator and adopt the standard tool-calling protocol
206 from SWE-agent [52]. To call a sub-agent, the orchestrator must parse its docstring to understand
207 the functionality and supply the required inputs. For example, an issue-localizer sub-agent requires
208 the issue summary as input; without it, the sub-agent cannot operate. To ensure this, we introduce
209 a warm-up stage that rewrites each generated sub-agent’s docstring into a precise specification of
210 its inputs and outputs, enabling the orchestrator to integrate it correctly. Details are provided in
211 Appendix A.1.1.
212213 **Sub-agent evaluation** At each round t , we select a set of K sub-agents $\Omega_t = \{\omega_1, \dots, \omega_K\} \subseteq$
214 Γ_{t-1} . Given this set, an orchestrator π_t is instantiated by prompting an LLM (see Appendix A.1.3),
215 and the system (π_t, Ω_t) is evaluated on a subset of example problems from a design set. This
216 evaluation yields a performance score $u_\omega \in [0, 1]$ for each sub-agent $\omega \in \Omega_t$. A straightforward
217 choice for u_ω is the success rate of the system on the design set, but as discussed in Section 4.2, this
218 metric is suboptimal and we propose a more effective alternative.
219

216 **Balancing exploration and exploitation on sub-agent selection** After bootstrapping the archive,
 217 the next challenge is deciding which sub-agents to evaluate in each round. To balance exploration
 218 with exploitation, we adopt the Upper Confidence Bound (UCB) [26] strategy. For each sub-agent
 219 $\omega \in \Gamma_{t-1}$, we track its empirical mean $\hat{\mu}_\omega(t)$ of the performance score of u_ω and selection count
 220 $n_\omega(t)$ up to round t . The UCB score of a sub-agent ω at round t is defined as

$$222 \quad \text{UCB}_\omega(t) = \hat{\mu}_\omega(t) + \sqrt{\frac{2 \ln t}{n_\omega(t)}}.$$

225 The first term favors sub-agents with high observed performance, while the second term gives an
 226 optimism bonus to under-sampled sub-agents (i.e., exploration). At each round t , we select the
 227 top- K sub-agents based on their UCB scores, ensuring that evaluations increasingly focus on strong
 228 candidates while still allocating time to uncertain ones.

229 **Expanding the archive** A fixed archive risks stagnation: once UCB identifies a few strong sub-
 230 agents, it will repeatedly exploit them, leaving little opportunity to discover new behaviors. To
 231 address this, we expand the archive dynamically using a Chinese Restaurant Process (CRP) [3; 48].
 232 At each round t , we prompt the LLM to generate a new sub-agent distinct from those in the current
 233 archive Γ_{t-1} (see Appendix A.1.2). The probability of introducing a new sub-agent is

$$235 \quad \text{Pr(new at } t) = \frac{\theta}{\theta + |\Gamma_{t-1}|},$$

236 where $\theta > 0$ is a concentration parameter. This mechanism ensures diversity: when the archive is
 237 small, new sub-agents are frequently added; as the archive grows, the probability decreases, shifting
 238 the emphasis toward reuse of existing ones. Over time, the expected number of distinct sub-agents
 239 after T rounds grows as $O(\theta \log T)$, providing unbounded but controlled expansion. We also run the
 240 warmup stage (Appendix A.1.1) to ensure the sub-agent is usable by the orchestrator.

Algorithm 1 Bandit Optimization for Agent Design (BOAD)

245 **Require:** budget B , number of sub-agents to select K , concentration θ
 246 1: Initialize archive $\Gamma_0 \leftarrow \text{BOOTSTRAP}.$
 247 2: **for** $t = 1, 2, \dots, B$ **do**
 248 3: With probability $\frac{\theta}{\theta + |\Gamma_{t-1}|}$, create a new sub-agent ω_{new} and set $\Gamma_t \leftarrow \Gamma_{t-1} \cup \{\omega_{\text{new}}\}$;
 249 otherwise set $\Gamma_t \leftarrow \Gamma_{t-1}$.
 250 4: **for** each $\omega \in \Gamma_t$ **do**
 251 5: **if** $n_\omega(t-1) = 0$ **then**
 252 6: $\text{UCB}_\omega(t) \leftarrow +\infty$ ▷ force initial exploration
 253 7: **else**
 254 8: $\text{UCB}_\omega(t) \leftarrow \hat{\mu}_\omega(t-1) + \sqrt{\frac{2 \ln t}{n_\omega(t-1)}}$
 255 9: **end if**
 256 10: **end for**
 257 11: Select top- K sub-agents based on UCB scores as a set of sub-agents Ω_t .
 258 12: Instantiate orchestrator π_t conditioned on Ω_t .
 259 13: Evaluate (π_t, Ω_t) on a subset of training problems; observe performance score $u_\omega \in [0, 1]$
 (Sec. 4.2).
 260 14: Update $\hat{\mu}_\omega(t)$ and $n_\omega(t)$ for each $\omega \in \Omega_t$.
 261 15: **end for**

 262
 263 **4.2 HINDSIGHT CREDIT ASSIGNMENT**

264 A central challenge in our framework is defining the performance score u_ω of individual sub-agents
 265 ω . A simple approach is to set the score of a sub-agent to the success rate of all trajectories that
 266 include it:

$$267 \quad u_\omega = \frac{1}{|\mathcal{T}_\omega^t|} \sum_{\tau \in \mathcal{T}_\omega^t} \mathbb{1}\{\tau \text{ is successful}\},$$

270 where \mathcal{T}_ω^t is the set of trajectories at round t in which ω is used by the orchestrator π . However,
 271 this suffers from a “free-rider” problem: a sub-agent may appear effective simply because it often
 272 co-occurs with strong sub-agents, even if it contributes little itself.
 273

274 To overcome this, we adopt a hindsight-based credit assignment strategy. The idea is to reward
 275 a sub-agent whenever its actions help the orchestrator make progress toward solving the problem,
 276 even if the orchestrator ultimately fails. Thus, sub-agents that provide useful intermediate steps
 277 are credited, while those that do not are penalized, regardless of the outcomes. Concretely, let
 278 $\tau = (a_1, o_1, \dots, a_T, o_T)$ denote the trajectory of actions and observations produced during problem
 279 solving. For each sub-agent ω that appears in τ , we query an LLM judge (Appendix A.1.4) with the
 280 trajectory and obtain a binary label $\ell_\omega(\tau) \in \{0, 1\}$, where $\ell_\omega(\tau) = 1$ indicates that the LLM judge
 281 deems ω ’s contribution in the trajectory as helpful. The performance score of sub-agent ω is then
 282 defined as the empirical average over all evaluated trajectories:
 283

$$u_\omega = \frac{1}{|\mathcal{T}_\omega^t|} \sum_{\tau \in \mathcal{T}_\omega^t} \ell_\omega(\tau).$$

285 This hindsight-based score $u_\omega \in [0, 1]$ provides a more reliable estimate of the utility of ω than
 286 success rates. By directly linking credit to judged contributions, it avoids free-riding effects.
 287

288 5 EXPERIMENTS

290 Our experiments address the central question: *Can properly designed hierarchical multi-agent*
 291 *systems improve the generalization performance of SWE agents?* We further analyze how the systems
 292 discovered by our algorithm differ from human-designed ones and examine the contribution of each
 293 design choice to the overall performance gains.
 294

295 5.1 SETUP

297 **Task format and datasets.** We evaluate on the SWE-BENCH benchmarks: SWE-BENCH VERIFIED
 298 (500 instances) [24] and SWE-BENCH LIVE (300 instances) [58]. VERIFIED is a curated, frozen
 299 set of real GitHub issues, while LIVE continuously adds newly collected, human-verified issues
 300 from active repositories, making it more resistant to data contamination [50] and better suited for
 301 testing generalization to out-of-distribution problems. Each instance includes a GitHub issue, a
 302 repository-specific container image, and an executable test harness. The agent must interact with the
 303 repository (files and, when available, history) and produce a patch that resolves the issue by passing
 304 *all* tests (pass-to-pass and fail-to-pass).

305 To avoid overfitting and limit design-time compute, we construct a small *design set* by sampling
 306 one random issue per repository (12 total) from VERIFIED, ensuring diversity while keeping the set
 307 small. The design set is disjoint from all issues in LIVE. All results in Tables 2 and 3 are reported on
 308 VERIFIED and LIVE (lite) splits. We also report the result of BOAD on VERIFIED (HELD OUT) that
 309 exclude the 12 issues used in the design set.

310 **Optimization Details** During BOAD optimization, we run $B = 20$ rounds of the bandit loop
 311 (Algorithm 1) (testing on up to 100 rounds shows that sub-agents created after around 20 rounds
 312 are generally worse, and 20 iterations is enough to converge to a state where helpful rate and UCB
 313 rankings align). Each time a new sub-agent is created, the sub-agent first goes through a warm-
 314 up process, which uses randomly sampled instances from the design set to iteratively refine the
 315 documentation/instance prompt of the sub-agent, ensuring that the sub-agent is usable. We use
 316 $W = 4$ rounds for the warm-up process. Each round samples $K = 3$ sub-agents and evaluates
 317 them on the design set. **The optimization took around 12 hours on a machine with 56 CPU cores**
 318 **and 440GB RAM.** For running SWE-agent (orchestrator and sub-agent LLM inference), we deploy
 319 Seed-OSS-36B-Instruct on one H100 GPU node. Note that by the tenth iteration, the top-2 most
 320 helpful subagents are the same as the ones converged on later. Running $B = 10$ iterations took less
 321 than 7 hours.

322 **Implementation.** All experiments use the SWE-AGENT scaffold with a set of default tools from
 323 SWE-agent [52]: `edit_anthroptic` (file viewing/editing), `bash` (restricted shell commands),

and submit. The orchestrator calls sub-agents through the same API, passing information via a *context* parameter; sub-agents return outputs through this channel without access to the orchestrator’s history. We use Claude-4 to generate candidate designs (Section 4.1) with temperature 0.0 and evaluate sub-agent helpfulness (Section 4.2). For execution, both orchestrator and sub-agents use Seed-OSS-36B-Instruct with temperature 0.0, unless specified, a strong instruction-following model that is not heavily tuned on SWE tasks. This choice ensures improvements reflect the benefit of orchestration rather than fine-tuning on SWE-task specific data. Each sub-agent is equipped with prompts discovered by BOAD, defined with docstrings and argument specs, and invoked via XML-based tool calling.

Finally, for evaluation, we use the two subagents with the highest helpfulness score (the hindsight-based score) found during the subagent discovery (Appendix A.2.1). We also ablate by varying k in the top- k selection and by using success rate instead of helpfulness as the ranking metric (Section 5.3).

5.2 MAIN RESULTS

Table 1: Success rate on SWE-BENCH VERIFIED and SWE-BENCH LIVE.

Scale	Model	Scaffold	Verified Resolved (%)	Live Resolved (%)
Large	GPT-4o [52]	SWE-agent	23.0	10.0
	GPT-4o [51]	Agentless	38.8	11.7
	Claude 3.5 Sonnet [5]	Agentless	50.8	–
	Claude 3.5 Sonnet [5]	OpenHands	53.0	–
	Claude 3.7 Sonnet [4]	SWE-agent	62.4	13.7 ¹
	Claude 4.0 Sonnet [6]	SWE-agent	66.8	–
	Claude 4.0 Sonnet [6]	OpenHands	70.4	–
	DeepSeek-R1 [18]	Agentless	49.2	–
	DeepSeek-V3 [15]	Agentless	42.0	13.3
Small	GLM-4.5-Air [45]	OpenHands	57.6	–
	GLM-4.5-Air [45]	SWE-Agent	–	17.7
	Qwen3-Coder 480B/A35B Instruct [47]	OpenHands	69.6	24.7
	Qwen3-Coder-30B-A3B-Instruct [47]	SWE-agent	–	17.0
	Qwen3-Coder-30B-A3B-Instruct [47]	OpenHands	51.6	–
	Devstral-Small-2505 [7]	OpenHands	46.8	–
	Seed-OSS-36B-Instruct [46]	SWE-agent (baseline)	49.8	12.3
Medium	Seed-OSS-36B-Instruct [46]	SWE-agent + Manual Sub-agent	47.4	14.0
	Seed-OSS-36B-Instruct [46]	SWE-agent + Evolutionary Search	46.0	17.0
Large	Seed-OSS-36B-Instruct [46]	SWE-agent + BOAD	53.2 ²	20.0

Success Rate Table 1 shows that with Seed-OSS-36B-Instruct, BOAD resolves **20.0%** of issues on LIVE, ranking second on the leaderboard and outperforming larger models in popular scaffolds (e.g., GPT-4o, DeepSeek-V3, GLM-4.5-Air, Claude 3.7 Sonnet). This is a **63%** improvement over the same model with default SWE-agent tools. On VERIFIED, BOAD achieves **53.12%**, surpassing many larger models (e.g., GPT-4o, Claude 3.5 Sonnet OpenHands, DeepSeek-R1, DeepSeek-V3) and setting a new state of the art among smaller models (e.g., Qwen3-Coder-30B-A3B-Instruct, Devstral-Small-2505), with a **13.4%** gain over the default SWE-agent. Interestingly, adding manually designed sub-agents (Appendix A.2.4) from prior work [8; 35; 11] lowers performance, indicating that human-crafted roles can be misaligned with LLM behavior. Overall, these results demonstrate that BOAD automatically discovers orchestrator–sub-agent structures that not only boost in-distribution performance but also generalize more effectively to out-of-distribution tasks.

Comparison with Evolutionary Baseline We also implement an evolutionary search for a multi-agent system adapted from Automated Design of Agentic Systems [22], as a baseline to compare BOAD against. Implementation details for the evolutionary search are provided in Appendix A.4.1. When using the same number of evaluation instances (i.e number of SWE-Bench patches generated), we find that the sub-agents generated by the evolutionary search achieve worse performance (17.0% vs 20.0% on SWE-Bench-Live) than BOAD. Additionally, for the same number of iterations, the

¹The SWE-bench-live leaderboard score was 17.7, based on an earlier issue set from April 2025.

²53.1 on the SWE-bench-verified set excluding the 12 issues used in the design set.

378 cost of Claude API calls is more than double than that of BOAD due to the need of generating many
 379 sub-agents at each iteration, whereas BOAD reuses subagents across iterations.
 380

381 **Token Analysis** In addition to success rate, we analyze the test-time token usage of the hierar-
 382 chical multi-agent system discovered by BOAD in comparison to the default single-agent system.
 383 Hierarchical multi-agent systems introduce communication overhead, since agents must exchange
 384 information, but they can also reduce context length: sub-agents focus on specialized sub-tasks while
 385 the orchestrator handles high-level coordination without low-level details. Table 2 compares token
 386 usage between SWE-agent and BOAD. Total tokens refer to the average sum of input and output
 387 tokens per issue, while max input tokens capture the average maximum input length per instance.
 388 Surprisingly, the total token count is comparable—and even lower on SWE-bench-live—than in
 389 the original SWE-agent. Moreover, BOAD consistently reduces input tokens, confirming that task
 390 decomposition shortens context length.
 391

392 Table 2: **Token usage.** BOAD lowers input token counts, thus shortening the model’s input context length.

Metric	Setting	Verified	Live
Total tokens (M)	SWE-agent	0.92	1.49
	SWE-agent + BOAD	0.93 (+0.7%)	1.13 (-23.8%)
Max input tokens	SWE-agent	34.6k	49.0k
	SWE-agent+BOAD	30.5k (-11.6%)	36.7k (-25.0%)

400 5.3 ABLATION STUDIES AND ANALYSIS

401 Table 3: **Ablation studies and analysis.** Each row corresponds to one research question. Results are reported on
 402 SWE-bench Live using Seed-OSS-36B-Instruct unless otherwise specified.
 403

Research Question	Configuration	SWE-Bench Live (%)
Does prompt optimization explain the gains?	w/o Sub-agent	16.3
	w Sub-agent	20.0
Do more sub-agents improve performance?	Top-5 sub-agents	13.7
	Top-4 sub-agents	16.7
	Top-3 sub-agents	16.3
	Top-2 sub-agents	20.0
	Top-1 sub-agent	16.3
Do we need to customize the orchestrator?	w/o customization	16.7
	w customization	20.0
Is expanding the sub-agent archive needed?	w/o expansion	17.0
	w expansion	20.0
Is hindsight credit assignment necessary?	Top-3 subagents (success rate)	11.3
	Top-3 subagents (helpfulness)	16.3
	Top-2 subagents (success rate)	15.3
	Top-2 subagents (helpfulness)	20.0
Are discovered sub-agents transferable to other models?	Claude 3.7 Sonnet + Top-2 sub-agents (helpfulness)	13.7 16.3

424 **Does prompt optimization explain the gains?** One possible explanation for BOAD’s performance
 425 improvement is that it simply arises from better prompt optimization of SWE-agent. To test this, we
 426 introduce a baseline that optimizes the SWE-agent prompt without adding sub-agents (w/o Sub-agent).
 427 We run 10 iterations: in each, a new SWE-agent prompt is generated by prompting Claude-4 with
 428 the template shown in A.1.5, evaluated on 12 issues (the same setting as BOAD). The first iteration
 429 is initialized without history, and from the second onward, prompt generation is conditioned on the
 430 top five prompts from previous rounds, ranked by performance. Results in Table 3 show that prompt
 431 optimization alone does not reach the performance of BOAD, indicating that the gains are not solely
 432 due to prompt tuning but from the discovery of effective sub-agents and orchestration.

432 **Do more sub-agents improve performance?** One might expect performance to improve as more
 433 sub-agents are added, since each can specialize. To test this, we vary the number of top- K sub-agents
 434 from 1 to 5 based on the helpfulness score (Section 4.2) and evaluate on LIVE. Surprisingly, Table
 435 3 shows that performance peaks with exactly two sub-agents, achieving 60/300 (20.0%). A single
 436 sub-agent (49/300) fails to leverage specialization, while larger teams of three (49/300), four (50/300),
 437 or five (41/300) reduce performance due to communication and coordination overhead. These results
 438 suggest that small, focused teams strike the best balance, outperforming both minimal and overly
 439 large teams of sub-agents.

440 **Do we need to customize the orchestrator?** We next ask whether gains come solely from sub-agent
 441 discovery or if the orchestrator must also adapt to its team. We compare two prompting strategies:
 442 (i) a generic prompt encouraging sub-agent calls (Appendix A.1.1), and (ii) a customized prompt
 443 generated by Claude-4 that explicitly references the top two sub-agents (selected by helpfulness
 444 scores) and outlines a plan for using them. Both settings use the same sub-agent set, but only the
 445 customized prompt allows the orchestrator to reason about and plan calls to specific sub-agents.
 446 Results in Table 3 show the customized orchestrator (w customization) achieves 60/300 (20.0%),
 447 versus 50/300 (16.7%) (w/o customization) for the generic one. This indicates that while sub-agents
 448 provide new capabilities, the orchestrator must also be specialized to effectively coordinate them.
 449

450 **Is expanding the sub-agent archive needed?** As discussed in Section 4.1, the initial archive may be
 451 limited, and adding new sub-agents during the design process could be necessary to discover stronger
 452 ones. To test this, we compare orchestrator performance using (i) sub-agents from the initial archive
 453 (w/o expansion) and (ii) sub-agents selected at the end of the design process (w expansion). Both
 454 settings use two sub-agents, consistent with our best configuration in Section 5.2, and the orchestrator
 455 is generated as described in Section 4.1. Results in Table 3 show that final sub-agents outperform
 456 those from the initial archive, highlighting the importance of expanding the archive over time.
 457

458 **Is hindsight credit assignment necessary?** To address free-riding issues in sub-agent selection
 459 (Section 4.1), we use a helpfulness score to measure each sub-agent’s contribution. To test its
 460 importance, we compare orchestrator performance when sub-agents are selected by (i) individual
 461 success rate versus (ii) helpfulness score. As shown in Table 3, helpfulness-based selection consis-
 462 tently outperforms success-rate selection, indicating that hindsight credit assignment (Section 4.2) is
 463 essential for identifying useful sub-agents.
 464

465 **Are discovered sub-agents transferable to other models?** Since BOAD optimizes sub-agents
 466 for a specific model, we ask whether the best sub-agents differ across models and whether effective
 467 sub-agents can transfer. To test this, we apply the sub-agents from Section 5.2 to SWE-agent+Claude-
 468 3.7-Sonnet. As shown in Table 3, the discovered sub-agents do transfer to some extent, though the
 469 gains are smaller than those achieved with Seed-OSS-36B-Instruct, the model used for sub-agent
 470 optimization.
 471

472 5.4 QUALITATIVE ANALYSIS OF SINGLE- VS MULTI-AGENT OUTCOMES

473 We manually inspected trajectories in which the single- and multi-agent systems produced different
 474 outcomes. Three recurring patterns emerged:
 475

- 476 **1. Over-editing (multi-agent advantage).** Single agents frequently produced extremely long
 477 patches, including attempts to create new tests and edits outside the scope of the bug. Such patches
 478 inflate apply time and increase the chance of failing pass-to-pass, even if the agent is able
 479 to address the primary fault. In contrast, the multi-agent system tended to emit short, localized
 480 patches, highlighting the advantage of separating phases like localization and editing/testing.
 481
- 482 **2. Multi-site fixes and coverage (multi-agent advantage).** When the fix required edits at multiple
 483 call sites or modules, the single agent often either over-edited unrelated regions or missed one
 484 or more necessary locations. The hierarchical system mitigated both omission and extraneous
 485 edits by delegating to a sub-agent for localization, which did a thorough analysis of the repository
 486 before making any targeted modifications.
 487
- 488 **3. Error propagation from unvalidated sub-agent outputs (multi-agent failure mode).** In a
 489 minority of cases, the multi-agent system failed while the single agent succeeded. Inspecting these
 490 outputs, we found that erroneous sub-agent outputs (e.g., incomplete span identification, misinter-
 491 pretation of the issue) were accepted as ground truth by the orchestrator, leading subsequent steps
 492 to failure.
 493

486 astray. Because there is no intermediate validation or self-checking, the orchestrator has limited
 487 ability to recover from such upstream.
 488

489 These observations align with our hypothesis: hierarchical delegation constrains edit scope and
 490 improves coverage of multi-site fixes, but introduces a new dependency on the *quality of sub-agent*
 491 *handoffs*. Incorporating lightweight verification (e.g., span cross-checks, invariant tests, or dual-read
 492 localization) is a promising mitigation for the third failure mode.

494 6 DISCUSSION & CONCLUSION

495 We present BOAD, a framework that formulates hierarchical multi-agent design as a sequential,
 496 online decision making problem to automatically discover multi-agent systems for long-horizon
 497 software engineering tasks. Our experiments show that automatically discovered sub-agents, when
 498 combined with a customized orchestrator, outperform single-agent and manually designed multi-agent
 499 systems on both SWE-BENCH VERIFIED and SWE-BENCH LIVE.

500 **Limitations and Future Work:** We find that discovered sub-agents transfer across models only
 501 partially and failure cases highlight error propagation when orchestrators unconditionally accept
 502 sub-agent outputs. Future work should explore evolution on large models, adaptive team sizing,
 503 verification, as well as extending the framework to domains beyond software engineering.

504 ETHICS STATEMENT

505 Coding agents hold strong promise for automating code generation and bug fixing, but they also
 506 carry risks of unintended or harmful outputs. For instance, an LLM-based agent may produce
 507 commands that could compromise a system (e.g., downloading unauthorized packages or deleting
 508 user files with `rm -rf`). To mitigate these risks in our study, all experiments were conducted within
 509 Docker containers, providing isolated and sandboxed environments that prevent harmful commands
 510 from impacting real user devices and substantially reducing the potential for actual harm. Our
 511 implementation also builds upon the SWE-Agent framework, which has been previously published
 512 and reviewed under established ethical standards. We carefully follow its curated protocols and
 513 licensing requirements.

514 Nevertheless, as with any AI-based coding agent framework, there remains the risk of deliberate
 515 misuse, for example, a malicious user prompting the system to generate harmful or hacking code.
 516 While our contribution is centered on advancing the technical design of hierarchical coding agents,
 517 we emphasize that real-world deployment should be coupled with responsible auditing and oversight,
 518 so that potential misuse and unintended consequences can be effectively mitigated.

523 524 REPRODUCIBILITY STATEMENT

525 For all open-source LLMs (e.g., Seed-OSS-36B-Instruct, Qwen3-Coder-30B-A3B-Instruct), we rely
 526 on their official releases. Commercial LLMs and LLM-based tools are accessed through their official
 527 APIs and reference implementations. We provide detailed implementation notes of BOAD, including
 528 the prompt design for meta-agents, sub-agents, and LLM judges, in Section 5.1. To further support
 529 reproducibility and foster future research, we will also release all of our code, used data, and prompts.

531 532 REFERENCES

533 [1] Alekh Agarwal, Haipeng Luo, Behnam Neyshabur, and Robert E Schapire. Corralling a band
 534 of bandit algorithms. In *Conference on Learning Theory*, pp. 12–38. PMLR, 2017.

535 [2] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-
 536 Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts,
 537 Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khatab.
 538 Gepa: Reflective prompt evolution can outperform reinforcement learning, 2025. URL <https://arxiv.org/abs/2507.19457>.

540 [3] David J Aldous. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-*
 541 *Flour XIII—1983*, pp. 1–198. Springer, 2006.

542

543 [4] Anthropic. Claude 3.7 sonnet and claude code, 2025. URL <https://www.anthropic.com/news/clause-3-7-sonnet>.

544

545 [5] Anthropic. Claude 3.5 sonnet, 2025. URL <https://www.anthropic.com/news/clause-3-5-sonnet>.

546

548 [6] Anthropic. Introducing claude 4, 2025. URL <https://www.anthropic.com/news/clause-4>.

549

550 [7] Anthropic. Devstral, 2025. URL <https://mistral.ai/news/devstral>.

551

552 [8] Daman Arora, Atharv Sonwane, Nalin Wadhwa, Abhav Mehrotra, Saiteja Utpala, Ramakrishna
 553 Bairi, Aditya Kanade, and Nagarajan Natarajan. Masai: Modular architecture for software-
 554 engineering ai agents. *arXiv preprint arXiv:2406.11638*, 2024.

555

556 [9] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement
 557 learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

558

559 [10] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement
 560 learning. *Discrete event dynamic systems*, 13(4):341–379, 2003.

561

562 [11] Dong Chen, Shaoxin Lin, Muhan Zeng, Daoguang Zan, Jian-Gang Wang, Anton Cheshkov, Jun
 563 Sun, Hao Yu, Guoliang Dong, Artem Aliev, et al. Coder: Issue resolving with multi-agent and
 564 task graphs. *arXiv preprint arXiv:2406.01304*, 2024.

565

566 [12] Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje Karlsson, Jie Fu, and
 567 Yemin Shi. Autoagents: A framework for automatic agent generation. In *Proceedings of the
 568 Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South
 569 Korea, August 3-9, 2024*, pp. 22–30. ijcai.org, 2024. URL <https://www.ijcai.org/proceedings/2024/3>.

570

571 [13] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization.
 572 *Annals of operations research*, 153(1):235–256, 2007.

573

574 [14] Chris Dann, Claudio Gentile, and Aldo Pacchiano. Data-driven online model selection with
 575 regret guarantees. In *International Conference on Artificial Intelligence and Statistics*, pp.
 576 1531–1539. PMLR, 2024.

577

578 [15] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu,
 579 Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian
 580 Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
 581 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang,
 582 Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo,
 583 Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong
 584 Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean
 585 Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li,
 586 Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian,
 587 Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du,
 588 R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu
 589 Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu,
 590 Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng
 591 Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng,
 592 Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang,
 593 X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen,
 594 Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang,
 595 Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi
 596 Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei,
 597 Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng
 598 Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying

594 He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo,
 595 Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha,
 596 Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou,
 597 Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang,
 598 Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong
 599 Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu,
 600 Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report,
 601 2025. URL <https://arxiv.org/abs/2412.19437>.

602 [16] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function
 603 decomposition. In *Proceedings of the 17th International Conference on Machine Learning*
 604 (*ICML*), pp. 118–126. Morgan Kaufmann, 2000.

605 [17] Dylan J Foster, Akshay Krishnamurthy, and Haipeng Luo. Model selection for contextual
 606 bandits. *Advances in Neural Information Processing Systems*, 32, 2019.

607 [18] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
 608 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in
 609 llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

610 [19] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian,
 611 and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful
 612 prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.
 613 URL <https://openreview.net/forum?id=ZG3RaNIs08>.

614 [20] Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new*
 615 *evolutionary computation*, pp. 75–102. Springer, 2006.

616 [21] Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent:
 617 Hierarchical working memory management for solving long-horizon agent tasks with large
 618 language model. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher
 619 Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational*
 620 *Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, Vienna, Austria, July 2025. Association
 621 for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1575.
 622 URL <https://aclanthology.org/2025.acl-long.1575/>.

623 [22] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint*
 624 *arXiv:2408.08435*, 2024.

625 [23] Yu Jiang, Ke Wang, Xin Wang, Yanyan Zhao, and Zhiqiang Zhang. Extracting concise bug-
 626 fixing patches from human-written patches in version control systems. In *Proceedings of the*
 627 *43rd International Conference on Software Engineering (ICSE)*, pp. 1351–1362. IEEE/ACM,
 628 2021.

629 [24] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik
 630 Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint*
 631 *arXiv:2310.06770*, 2023.

632 [25] Jaehun Kim, Shunyu Yao, Howard Chen, Karthik Narasimhan, et al. Promptbreeder: Self-
 633 referential self-improvement via prompt evolution. In *International Conference on Learning*
 634 *Representations (ICLR)*, 2024.

635 [26] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. 2020.

636 [27] Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun
 637 Liu. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint*
 638 *arXiv:2412.05579*, 2024.

639 [28] Han Li, Yuling Shi, Shaoxin Lin, Xiaodong Gu, Heng Lian, Xin Wang, Yantao Jia, Tao Huang,
 640 and Qianxiang Wang. Swe-debate: Competitive multi-agent debate for software issue resolution.
 641 *arXiv preprint arXiv:2507.23348*, 2025.

648 [29] Fernanda Madeiral, Thomas Durieux, Matias Martinez, and Martin Monperrus. Bears: An
649 extensible java bug benchmark for automatic program repair studies. In *Proceedings of the 26th*
650 *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*,
651 pp. 468–478. IEEE, 2019.

652 [30] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia:
653 a benchmark for general ai assistants. In *The Twelfth International Conference on Learning*
654 *Representations*, 2023.

655 [31] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity
656 for processing information. *Psychological review*, 63(2):81, 1956.

657 [32] Kou Misaki, Yuichi Inoue, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba.
658 Wider or deeper? scaling LLM inference-time compute with adaptive branching tree search. In
659 *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025. URL <https://openreview.net/forum?id=3HF6yogDEM>.

660 [33] Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and
661 search. *Communications of the ACM*, 19(3):113–126, 1976.

662 [34] Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt
663 Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian,
664 M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian
665 Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and
666 algorithmic discovery, 2025. URL <https://arxiv.org/abs/2506.13131>.

667 [35] Huy Nhat Phan, Tien N Nguyen, Phong X Nguyen, and Nghi DQ Bui. Hyperagent: Generalist
668 software engineering agents to solve coding tasks at scale. *arXiv preprint arXiv:2409.16299*,
669 2024.

670 [36] Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier
671 Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement
672 learning. *arXiv preprint arXiv:2312.01072*, 2023.

673 [37] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*.
674 John Wiley & Sons, 2014.

675 [38] Jielin Qiu, Zuxin Liu, Zhiwei Liu, Rithesh Murthy, Jianguo Zhang, Haolin Chen, Shiyu Wang,
676 Ming Zhu, Liangwei Yang, Juntao Tan, et al. Locobench: A benchmark for long-context large
677 language models in complex software engineering. *arXiv preprint arXiv:2509.09614*, 2025.

678 [39] Stefano Rando, Luca Romani, Alessio Sampieri, Luca Franco, John Yang, Yuta Kyuragi, Fabio
679 Galasso, and Tatsunori Hashimoto. Longcodebench: Evaluating coding llms at 1m context
680 windows. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025. URL
681 <https://openreview.net/forum?id=GFPoM8Ylp8>. COLM 2025.

682 [40] Herbert A Simon. The structure of ill structured problems. *Artificial intelligence*, 4(3-4):
683 181–201, 1973.

684 [41] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec
685 Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback.
686 *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

687 [42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

688 [43] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A
689 framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):
690 181–211, 1999.

691 [44] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A
692 framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):
693 181–211, 1999.

702 [45] 5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang
 703 Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu,
 704 Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu,
 705 Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen,
 706 Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi
 707 Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng
 708 Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan,
 709 Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai,
 710 Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan
 711 Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi
 712 Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan,
 713 Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang,
 714 Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan
 715 Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu,
 716 Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu
 717 Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao
 718 Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan
 719 Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong
 720 Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng
 721 Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing
 722 Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao,
 723 Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu,
 724 Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang,
 725 Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5: Agentic, reasoning, and
 726 coding (arc) foundation models, 2025. URL <https://arxiv.org/abs/2508.06471>.
 727

726 [46] ByteDance Seed Team. Seed-oss open-source models. <https://github.com/ByteDance-Seed/seed-oss>, 2025.

728 [47] Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.

729 [48] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet
 730 processes. *Journal of the american statistical association*, 101(476):1566–1581, 2006.

731 [49] Dong Wang, Yu Li, Ming Jiang, Lu Zhang, and Zhiqiang Chen. A systematic mapping study of
 732 bug reproduction and fault localization. *Information and Software Technology*, 167:107316,
 733 2024.

734 [50] Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, Senjie Jin, Xiaoran Fan,
 735 Yuhao Zhou, Huijie Lv, Ming Zhang, et al. Reasoning or memorization? unreliable results of
 736 reinforcement learning due to data contamination. *arXiv preprint arXiv:2507.10532*, 2025.

737 [51] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying
 738 llm-based software engineering agents, 2024. URL <https://arxiv.org/abs/2407.01489>.

739 [52] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik
 740 Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software
 741 engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

742 [53] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhut-
 743 dinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop
 744 question answering. *arXiv preprint arXiv:1809.09600*, 2018.

745 [54] Chen Bo Calvin Zhang, Zhang-Wei Hong, Aldo Pacchiano, and Pulkit Agrawal. Orso: Ac-
 746 celerating reward design via online reward selection and policy optimization. *arXiv preprint
 747 arXiv:2410.13837*, 2024.

748 [55] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine:
 749 Open-ended evolution of self-improving agents, 2025. URL <https://arxiv.org/abs/2505.22954>.

756 [56] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen
 757 Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow
 758 generation. *arXiv preprint arXiv:2410.10762*, 2024.

759 [57] Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh Murthy, Tian Lan, Lei
 760 Li, Renze Lou, Jiacheng Xu, et al. Diversity empowers intelligence: Integrating expertise of
 761 software engineering agents. *arXiv preprint arXiv:2408.07060*, 2024.

762 [58] Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao
 763 Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, et al. Swe-bench goes live! *arXiv preprint*
 764 *arXiv:2505.23419*, 2025.

765 [59] Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arik. Chain of
 766 agents: Large language models collaborating on long-context tasks. In Amir Globersons, Lester
 767 Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang
 768 (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural*
 769 *Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December*
 770 *10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ee71a4b14ec26710b39ee6be113d7750-Abstract-Conference.html.

771 [60] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang.
 772 Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan
 773 Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett,
 774 and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine*
 775 *Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR,
 776 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/zhou24r.html>.

777 [61] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari,
 778 Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Bench-
 779 marking code generation with diverse function calls and complex instructions. *arXiv preprint*
 780 *arXiv:2406.15877*, 2024.

781
 782
 783
 784
 785
 786
 787
 788
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809

810

811

812

A EXPERIMENT DETAILS

813

A.1 PROMPT TEMPLATES

814

A.1.1 PROMPT TEMPLATE FOR REFINING SUBAGENT DURING WARMUP STAGE

815

816

817

818

Prompt template for refining subagent during warmup stage

819

820

You are improving a subagent's prompts/config for a software engineering (SWE) automation system, based
 ↪ on recent run trajectories. The subagent is used by an AI main agent to address code issues.

821

CONTEXT

- You will receive trajectory summaries below, starting with the main agent's trajectory, followed by
 ↪ any subagent trajectories in call order.
- Each summary shows what the agent did, what was observed, and how far it progressed.

822

GOAL

Analyze the subagent's performance and suggest improvements to make it:

1. More discoverable by the main agent (when appropriate)
2. More reliable in its behavior
3. More useful in its output

823

ANALYSIS FRAMEWORK

Consider these questions:

- Did the main agent discover and use the subagent when it should have?
- Did the subagent behave as expected and return useful information?
- Were there missed opportunities or inefficient behaviors?

824

IMPROVEMENT TYPES

Focus on one or more of these areas:

1. ****docstring**:** Make the subagent easier for the main agent to discover and choose appropriately.
 ↪ CRITICAL: Make sure to include "[subagent]" at the beginning of the docstring.
2. ****context_description**:** Improve the description of the 'context' argument (the only argument) to be
 ↪ clearer and more helpful
3. ****instance_template**:** Better incorporation of context, clearer framing for each problem instance

825

Note that the docstring and context_description are visible only to the main agent, while the
 ↪ instance_template is visible only to the subagent.

826

Thus, if the subagent was not called, you should not edit instance_template. Similarly, if the only
 ↪ issue is the subagent's trajectory, not how it was called, do not edit docstring or
 ↪ context_description.

827

PRINCIPLES

- Make surgical, targeted improvements rather than broad rewrites
- Preserve existing style and capabilities. Only edit components that need improvement.
- Focus on clarity, discoverability, and reliability
- Ensure generality. Avoid repo- or issue-specific assumptions.
- CRITICAL: DO NOT WRITE ANYTHING SPECIFIC TO THE PARTICULAR CODEBASE, PROJECT, OR DOMAIN.

828

OUTPUT FORMAT

First, explain your reasoning about what issues you noticed with the provided trajectory and what
 ↪ improvements you're making. Then, output the YAML in a code block.

829

****IMPORTANT**:** Only suggest edits when you identify a clear, specific problem. If the entire subagent
 ↪ is working well, use an empty updates dictionary.

830

Sample outputs:

****When improvements are needed:****

Explain your reasoning here.

```yaml

updates:

docstring: "<improved docstring if needed>"
 context_description: "<improved context argument description if needed>"
 instance_template: "<improved instance template if needed>"

```

831

RULES

- Only include keys that you intend to change.
- Start with 'updates:' as the top-level key. If there are no updates to make, the value for 'updates'
 ↪ should be an empty dictionary.
- You may update any combination of the three fields (docstring, context_description,
 ↪ instance_template), but only include a field if it needs improvement.
- No explanations or extra content in the YAML
- Keep each field concise but complete

832

HEURISTICS

- ****Discovery issues**:** Strengthen docstring with clear use cases and when to invoke
- ****Insufficient context passed to subagent**:** Improve context_description with clearer argument
 ↪ explanation
- ****Subagent behavior and output (Incorrect subagent trajectory or return information)**:** Improve
 ↪ instance_template with better instructions and output specifications

833

```{{TRAJECTORIES}}

864 A.1.2 META AGENT PROMPTS
865866 Prompt for generating a new subagent configuration
867

```

868 You are an expert at designing custom tools for SWE-agent, an autonomous agent that can resolve code
869    ↳ issues in large repositories.

870 YOUR TASK
871 Invent a subagent tool for SWE-agent.
872 - The subagent should enable the main agent to better perform its task of automatically resolving code
873    ↳ issues in large static repositories.
874 - Design for broad applicability across the full workflow. Create broad subagents that are can solve an
875    ↳ entire step of the pipeline, such as:
876      - code localization
877      - reproducing issues and running scripts/tests
878      - code editing/patching
879      - code testing
880 - The subagents created should ONLY be focused on correctness of the final patch (e.g. style,
881    ↳ complexity of code does not matter)
882 - PRIORITY TOKEN EFFICIENCY: Design concise, focused subagents that use minimal tokens. Avoid verbose
883    ↳ explanations or redundant information that wastes tokens.
884 - If you see a subagent that looks good but has bad token efficiency, you may generate a similar
885    ↳ subagent with the same function but better implementation.
886 - The subagent takes a SINGLE argument that is a string, called "context".
887 - BE NOVEL! Think carefully about how to help the main agent perform one of its subtasks.
888   - Example subagents include localize, patch_editor, or code_tester.
889 - Do not create a subagent that overlaps with previous subagents (other than the token efficiency
890    ↳ situation).
891 - In your reasoning, you must explicitly list which steps the subagent supports (examples: explore,
892    ↳ read/search, edit, run, validate) and the expected outputs for each supported step.
893   - CRITICAL: Output exactly ONE YAML document with the tool under a single key, which is the name of
894    ↳ the tool.
895   - The name of the tool should be simple and descriptive.
896   - The docstring for each tool should be comprehensive and describe what the output contains, as well
897    ↳ as the state of the repository after the subagent is finished (if files will be edited or not,
898    ↳ etc.).
```

899 The structure must be exactly as follows:
900 Add reasoning here about WHY you design this subagent...

```

901   ...yaml
902     tool_name:
903       signature: "tool_name <context>"
904       docstring: docstring: "comprehensive description of this subagent, the output, and the state of
905         ↳ repository on completion. Starts with '[subagent]:'"
906       arguments:
907         - name: context
908           type: string
909           description: "detailed description of what the context parameter should contain"
910           required: true
911       subagent: true
912     ...
913
914     Sample output:
915     It may be useful to have a patch editor subagent. This would go well with previous subagents and help
916     ↳ the main agent more efficiently patch the issue.
917     ...yaml
918     patch_editor:
919       signature: "patch_editor <context>"
920       docstring: "[subagent] Fixes a specific part of code that has errors. Outputs the changes made with
921         ↳ reasoning. After calling, the correct changes are already implemented in the repository."
922       arguments:
923         - name: context
924           type: string
925           description: "A string containing the specific file path to make edits in, the lines where edits
926             ↳ need to be made, a comprehensive description of the issue with the code (do not assume the
927             ↳ subagent has any information about the repository or problem statement), and what to edit."
928           required: true
929       subagent: true
930     ...
931     {{PREVIOUS_ITERATION_FEEDBACK}}
```

908 Prompt for generating subagent templates
909

```

910 You are an expert at creating SWE-agent subagent configuration files for automating code-patching tasks
911    ↳ in large GitHub repositories.
912 Given a description of the subagent, you need to generate the system_template and instance_template
913    ↳ parts that will be used in the subagent configuration.

914 IMPORTANT FORMATTING RULES:
915 - First output your reasoning that details your thinking process for creating the templates. Then,
916    ↳ output a yaml block with both templates.
917 - Use MINIMAL spacing - avoid excessive blank lines
918 - Use only SINGLE blank lines between sections (never double or triple spacing)
919 - Keep templates compact and readable without unnecessary whitespace
920 - CRITICAL: Use YAML literal block syntax with | and |- (see example below)
921 - Do NOT use quoted strings - use literal blocks to avoid quotes in output
```

```

918
919     - Replace ONLY text in [] with text specific to the subagent. Do NOT MODIFY any other parts.
920     - Copy EXACTLY the parts other than [], including how to call the functions (e.g.
921     ↪ "<function=example_function_here>")

922     Output format:
923     [Reasoning here...]
924     ````yaml
925     system_template: |
926         You are a helpful [role] assistant that can interact with a computer to [main task].
927         <IMPORTANT>
928             * If user provides a path, you should NOT assume it's relative to the current working directory.
929             ↪ Instead, you should explore the file system to find the file before working on it.
930         </IMPORTANT>

931         You have access to the following functions:
932         {{command_docs}}

933         If you choose to call a function, you must ONLY reply in the following format with NO suffix:
934         Provide any reasoning for the function call here.
935         <function=example_function_name>
936             <parameter=example_parameter_1>value_1</parameter>
937             <parameter=example_parameter_2>
938                 This is the value for the second parameter
939                 that can span
940                 multiple lines
941                 </parameter>
942             </function>
943             (You must use the exact text function=" and "parameter=" for each function and argument, respectively,
944             ↪ e.g. <parameter=command>value</parameter>)

945             <IMPORTANT>
946             Reminder:
947                 - Function calls MUST follow the specified format, start with <function= and end with </function>
948                 - Required parameters MUST be specified
949                 - CRITICAL: Only call ONE function at a time
950                 - Always provide reasoning for your function call in natural language BEFORE the function call (not
951                 ↪ after)
952             </IMPORTANT>

953             <pr_description>
954                 {{problem_statement}}
955             </pr_description>

956             CRITICAL: Use the submit_subagent function to provide the results when you are finished with your
957             ↪ task.
958             You are ONLY responsible for your specific assigned task. Do NOT attempt to resolve entire
959             ↪ pr_description, only your task.
960             Your goal is to complete your task in the MINIMAL NUMBER of steps. Resolve the issue fast and call
961             ↪ submit_subagent as soon as possible.

962             instance_template: |-
963                 Your task:
964                 [Provide detailed, step-by-step instructions for your assigned subagent task, tailored to your
965                 ↪ specific role. The instructions must ONLY reference this subagent's function.]
966                 [If a context argument is provided, you MUST include its contents by inserting "{{context}}" here and
967                 ↪ explaining what the parameter is.]
968
969                 **CRITICAL: STAY IN YOUR LANE**
970                 - You are ONLY responsible for your specific assigned task
971                 - You are NOT responsible for solving the entire issue
972                 - You are NOT responsible for other subagent tasks
973                 - Focus EXCLUSIVELY on your assigned task and nothing else
974                 - CRITICAL: Call EXACTLY one function in your output!
975                 - CRITICAL: When you are finished, immediately call submit_subagent. Do not call any other tools or
976                 ↪ produce additional output.

977                 Focus exclusively on your assigned task and strictly follow these instructions. Do not attempt to
978                 ↪ address unrelated parts of the PR or perform work outside your specific subagent role.
979                 Use the submit_subagent tool after you are finished with your specific task to provide a clear and
980                 ↪ complete summary of your findings or changes.
981                 Your thinking should be thorough and so it's fine if it's very long.
982
983             Rules for generating templates:
984             1. The system_template should clearly define the subagent's role and capabilities based on the
985                 ↪ available tools
986             2. The instance_template should provide clear instructions for each task
987             3. Both templates should maintain consistent formatting with the base template
988             4. Ensure the templates encourage thorough analysis and clear documentation
989             5. MUST use literal block syntax: system_template: | and instance_template: |-
990             6. Never use quoted strings for templates
991             7. You should copy the given system template exactly other than the first sentence.
992             8. Modify the system template in the spots with [].

993             {{PREVIOUS_ITERATION_FEEDBACK}}

```

972 A.1.3 CUSTOM ORCHESTRATOR PLAN PROMPT
973974 Prompt template for generating a custom orchestrator plan given a set of subagents
975

```

976 You are a master workflow architect for automated software engineering. Your job is to design
977 → innovative, strategic workflows that maximize the effectiveness of available tools.
978
979 CONTEXT: You're designing workflows for an AI assistant that solves coding problems in software
980 → repositories. The assistant receives a problem description (like a bug report, feature request, or
981 → code issue) and needs to systematically work through the codebase to understand, fix, and validate
982 → the solution. The assistant has access to specialized "subagents" - each designed for specific
983 → aspects of the coding workflow.
984
985 You will be given a toolkit of specialized "subagents" - each with unique capabilities. Your challenge
986 → is to:
987 1. **Design** a comprehensive problem-solving plan that addresses the coding issue systematically
988 2. **Integrate** subagents strategically where they add the most value to your workflow
989 3. **Optimize** the sequence and wording of the plan to minimize the number of steps that the AI
990 → assistant takes while remaining effective
991
992 Think like a senior engineer designing a solution strategy. Consider:
993 - What are the key phases needed to solve this type of coding problem?
994 - Which subagents would be most valuable for specific phases?
995 - How can you combine subagent work with direct problem-solving phases?
996 - What's the most logical progression to integrate subagent input and output to solve the issue?
997 - How can you utilize subagents to minimize language model token usage and number of steps?
998
999 INPUT
1000 - The available subagents will be provided inline between the following tags:
1001 <available_subagents>
1002 {{subagents_overview}}
1003 </available_subagents>
1004 The content in <available_subagents> lists each subagent with its name and short docs
1005 → (summary/description). Treat it as the authoritative source for tool names and purposes.
1006
1007 WHAT TO OUTPUT
1008 - Output ONLY your strategic plan as plain text (no YAML, no code fences, no headers).
1009 - Each phase MUST start with a number and a period, e.g. "1. ...".
1010 - For subagent phases, use the exact form: "Use the <name> subagent to ..."
1011 - For direct phases, describe the action clearly without mentioning subagents, ensuring that it can be
1012 → applied to any problem.
1013 - Be creative and strategic - design workflows that combine different approaches effectively
1014 - Keep 3 to 7 steps total, but make each step purposeful and well-reasoned
1015 - Make sure the last steps are:
1016   - After you have solved the issue, delete any test files or temporary files you created.
1017   - Use the submit tool to submit the changes to the repository.
1018   - Do not mention any function-call formats or system details
1019
1020 EXAMPLE (illustrative only; adapt to the given input)
1021 <available_subagents>
1022 - name: issue_localizer | Identify files and code regions relevant to the issue.
1023 - name: error_reproducer | Reproduce the failing behavior and capture commands/outputs.
1024 - name: code_tester | Run tests/commands to verify the fix and regressions.
1025 </available_subagents>
1026
1027 Expected output EXAMPLE (plain text only):
1028 1. Use the issue_localizer subagent to map the problem space and identify all potentially affected
1029 → files and code regions.
1030 2. Analyze the problem description and examine the identified files to understand the root cause and
1031 → requirements.
1032 3. Use the error_reproducer subagent to create a reproducible test case and capture the exact failure
1033 → conditions.
1034 4. Design and implement the fix based on the analysis, focusing on the specific files and code areas
1035 → identified.
1036 5. Use the code_tester subagent to validate the fix against the original failure case and run
1037 → regression tests.
1038 6. After you have solved the issue, delete any test files or temporary files you created.
1039 7. Use the submit tool to submit the changes to the repository.
1040
1041 Now, based on the provided subagents, produce ONLY the numbered plan as plain text:
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3
```

```

1026
1027
1028 TOOL TO ANALYZE: {{TOOL_NAME}}
1029
1030 Your task is to determine if the subagent "{{TOOL_NAME}}" was helpful in this set of
1031 ↪ trajectories.
1032 A tool is considered helpful if:
1033 1. It was called/invoked by the main agent in the main agent trajectory
1034 2. It provided useful information, analysis, or insights that contributed to solving the
1035 ↪ problem
1036 3. The main agent made progress after using this tool (e.g., identified the issue, made
1037 ↪ code changes, validated results, etc.)
1038 4. It completed its task as intended (followed proper analysis process, not just got
1039 ↪ lucky results)
1040
1041 Look for positive evidence such as:
1042 - The subagent being called with appropriate parameters
1043 - The subagent providing insights that led to code changes or problem understanding
1044 - The main agent referencing or building upon the subagent's output
1045 - The subagent's output being used in subsequent reasoning or actions
1046
1047 Look for negative evidence such as:
1048 - The subagent not being called by the main agent, or called incorrectly
1049 - The subagent providing irrelevant or incorrect information that was not later used
1050 - The subagent's response was valid but did not move the main agent closer to resolving
1051 ↪ the problem.
1052 - The subagent failed to execute properly or had many errors during its run.
1053 - The subagent's output appeared correct, but its trajectory did not actually achieve
1054 ↪ those results (e.g., claimed to test code but just reported all tests passed).
1055 - The main agent had to call the subagent over and over again to get the proper results.
1056 - The subagent trajectory was unnecessarily long or verbose, taking many steps to
1057 ↪ complete its task
1058 - The main agent trajectory became inefficient due to excessive subagent calls or overly
1059 ↪ verbose subagent responses
1060 - The subagent's results did not actually help the main agent make progress in resolve
1061 ↪ the issue. If a subagent did not contribute to producing the correct patch, e.g. only
1062 ↪ improved performance, style, or documentation, this is NOT helpful.
1063
1064 Respond with YAML format (exactly):
1065 ---yaml
1066 helpful: true/false
1067 reasoning: |
1068   Brief explanation of why the tool was or wasn't helpful, including specific evidence
1069   ↪ from the trajectories
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

```

A.1.5 ORCHESTRATOR-ONLY PROMPT

```

1065 Prompt template for generating an orchestrator prompt under orchestrator-only settings
1066
1067 You are a master workflow architect for automated software engineering. Your job is to design effective
1068 ↪ workflows that solve coding problems efficiently.
1069
1070 CONTEXT: You're designing workflows for an AI assistant that solves coding problems in software
1071 ↪ repositories. The assistant receives a problem description (like a bug report, feature request, or
1072 ↪ code issue) and needs to work through the codebase to understand, implement changes, and validate
1073 ↪ the solution.
1074
1075 Your goal is to create workflows that are both effective at solving problems and efficient in their
1076 ↪ execution. Focus on designing strategic approaches that lead to successful problem resolution.
1077
1078 CONTEXT FOR PLAN USAGE:
1079 Your generated plan will be inserted into this agent template context:
1080
1081 I've uploaded a python code repository in the directory {{working_dir}}. Consider the following PR
1082 ↪ description:
1083
1084 <pr_description>
1085 {{problem_statement}}
1086 </pr_description>
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
20100
20101
20102
20103
20104
20105
20106
20107
20108
20109
20110
20111
20112
20113
20114
20115
20116
20117
20118
20119
20120
20121
20122
20123
20124
20125
20126
20127
20128
20129
20130
20131
20132
20133
20134
20135
20136
20137
20138
20139
20140
20141
20142
20143
20144
20145
20146
20147
20148
20149
20150
20151
20152
20153
20154
20155
20156
20157
20158
20159
20160
20161
20162
20163
20164
20165
20166
20167
20168
20169
20170
20171
20172
20173
20174
20175
20176
20177
20178
20179
20180
20181
20182
20183
20184
20185
20186
20187
20188
20189
20190
20191
20192
20193
20194
20195
20196
20197
20198
20199
20200
20201
20202
20203
20204
20205
20206
20207
20208
20209
20210
20211
20212
20213
20214
20215
20216
20217
20218
20219
20220
20221
20222
20223
20224
20225
20226
20227
20228
20229
20230
20231
20232
20233
20234
20235
20236
20237
20238
20239
20240
20241
20242
20243
20244
20245
20246
20247
20248
20249
20250
20251
20252
20253
20254
20255
20256
20257
20258
20259
20260
20261
20262
20263
20264
20265
20266
20267
20268
20269
20270
20271
20272
20273
20274
20275
20276
20277
20278
20279
20280
20281
20282
20283
20284
20285
20286
20287
20288
20289
20290
20291
20292
20293
20294
20295
20296
20297
20298
20299
20300
20301
20302
20303
20304
20305
20306
20307
20308
20309
20310
20311
20312
20313
20314
20315
20316
20317
20318
20319
20320
20321
20322
20323
20324
20325
20326
20327
20328
20329
20330
20331
20332
20333
20334
20335
20336
20337
20338
20339
20340
20341
20342
20343
20344
20345
20346
20347
20348
20349
20350
20351
20352
20353
20354
20355
20356
20357
20358
20359
20360
20361
20362
20363
20364
20365
20366
20367
20368
20369
20370
20371
20372
20373
20374
20375
20376
20377
20378
20379
20380
20381
20382
20383
20384
20385
20386
20387
20388
20389
20390
20391
20392
20393
20394
20395
20396
20397
20398
20399
20400
20401
20402
20403
20404
20405
20406
20407
20408
20409
20410
20411
20412
20413
20414
20415
20416
20417
20418
20419
20420
20421
20422
20423
20424
20425
20426
20427
20428
20429
20430
20431
20432
20433
20434
20435
20436
20437
20438
20439
20440
20441
20442
20443
20444
20445
20446
20447
20448
20449
20450
20451
20452
20453
20454
20455
20456
20457
20458
20459
20460
20461
20462
20463
20464
20465
20466
20467
20468
20469
20470
20471
20472
20473
20474
20475
20476
20477
20478
20479
20480
20481
20482
20483
20484
20485
20486
20487
20488
20489
20490
20491
20492
20493
20494
20495
20496
20497
20498
20499
20500
20501
20502
20503
20504
20505
20506
20507
20508
20509
20510
20511
20512
20513
20514
20515
20516
20517
20518
20519
20520
20521
20522
20523
20524
20525
20526
20527
20528
20529
20530
20531
20532
20533
20534
20535
20536
20537
20538
20539
20540
20541
20542
20543
20544
20545
20546
20547
20548
20549
20550
20551
20552
20553
20554
20555
20556
20557
20558
20559
20560
20561
20562
20563
20564
20565
20566
20567
20568
20569
20570
20571
20572
20573
20574
20575
20576
20577
20578
20579
20580
20581
20582
20583
20584
20585
20586
20587
20588
20589
20590
20591
20592
20593
20594
20595
20596
20597
20598
20599
20600
20601
20602
20603
20604
20605
20606
20607
20608
20609
20610
20611
20612
20613
20614
20615
20616
20617
20618
20619
20620
20621
20622
20623
20624
20625
20626
20627
20628
20629
20630
20631
20632
20633
20634
20635
20636
20637
20638
20639
20640
20641
20642
20643
20644
20645
20646
20647
20648
20649
20650
20651
20652
20653
20654
20655
20656
20657
20658
20659
20660
20661
20662
20663
20664
20665
20666
20667
20668
20669
20670
20671
20672
20673
20674
20675
20676
20677
20678
20679
20680
20681
20682
20683
20684
20685
20686
20687
20688
20689
20690
20691
20692
20693
20694
20695
20696
20697
20698
20699
20700
20701
20702
20703
20704
20705
20706
20707
20708
20709
20710
20711
20712
20713
20714
20715
20716
20717
20718
20719
20720
20721
20722
20723
20724
20725
20726
20727
20728
20729
20730
20731
20732
20733
20734
20735
20736
20737
20738
20739
20740
20741
20742
20743
20744
20745
20746
20747
20748
20749
20750
20751
20752
20753
20754
20755
20756
20757
20758
20759
20760
20761
20762
20763
20764
20765
20766
20767
20768
20769
20770
20771
20772
20773
20774
20775
20776
20777
20778
20779
20780
20781
20782
20783
20784
20785
20786
20787
20788
20789
20790
20791
20792
20793
20794
20795
20796
20797
20798
20799
20800
20801
20802
20803
20804
20805
20806
20807
20808
20809
20810
20811
20812
20813
20814
20815
20816
20817
20818
20819
20820
20821
20822
20823
20824
20825
20826
20827
20828
20829
20830
20831
20832
20833
20834
20835
20836
20837
20838
20839
20840
20841
20842
20843
20844
20845
20846
20847
20848
20849
20850
20851
20852
20853
20854
20855
20856
20857
20858
20859
20860
20861
20862
20863
20864
20865
20866
20867
20868
20869
20870
20871
20872
20873
20874
20875
20876
20877
20878
20879
20880
20881
20882
20883
20884
20885
20886
20887
20888
20889
20890
20891
20892
20893
20894
20895
20896
20897
20898
20899
20900
20901
20902
20903
20904
20905
20906
20907
20908
20909
20910
20911
20912
20913
20914
20915
20916
20917
20918
20919
20920
20921
20922
20923
20924
20925
20926
20927
20928
20929
20930
20931
20932
20933
20934
20935
20936
20937
20938
20939
20940
20941
20942
20943
20944
20945
20946
20947
20948
20949
20950
20951
20952
20953
20954
20955
20956
20957
20958
20959
20960
20
```

```

1080
1081 Can you help me implement the necessary changes to the repository so that the requirements specified in
1082 → the <pr_description> are met? I've already taken care of all changes to any of the test files
1083 → described in the <pr_description>. This means you DON'T have to modify the testing logic or any of
1084 → the tests in any way! Your task is to make the minimal changes to non-test files in the
1085 → {{working_dir}} directory to ensure the <pr_description> is satisfied. When solving the task,
1086 → **first create a plan by breaking the problem into subtasks**. Think systematically about the steps
1087 → needed to understand the problem, locate relevant code, implement changes, and verify the solution.
1088 → Follow this process:
1089 {{(plan)}} <-- YOUR PLAN GOES HERE
1090 You MUST follow the plan exactly.
1091 ...
1092
1093 AVAILABLE TOOLS:
1094 - bash: Execute shell commands for searching, testing, running scripts, exploring codebase structure
1095 - str_replace_editor: View, create, and edit files with precise string replacement capabilities
1096 - submit: Submit the final solution
1097
1098 LEARNING FROM HISTORY:
1099 <sampled_templates>
1100 {{sampled_templates_summary}}
1101 </sampled_templates>
1102
1103 If historical templates are provided above, identify what made the highest-scoring approaches
1104 → successful and what caused failures. Look for patterns in tool usage, step efficiency, and
1105 → problem-solving strategies. If no history exists, design breakthrough approaches that challenge
1106 → conventional software engineering workflows.
1107
1108 WHAT TO OUTPUT:
1109 - Create a step-by-step plan that an AI agent can execute systematically
1110 - Each step must clearly specify tool usage ("Use bash to..." or "Use str_replace_editor to...") and
1111 → expected outcomes
1112 - Design for Python repositories and PR-based problem solving
1113 - Focus on minimal, targeted changes rather than broad exploration
1114 - Structure as numbered steps (1., 2., 3., etc.) with logical flow
1115 - Final step must use submit tool to deliver the solution
1116 - Make each step actionable and specific enough for precise execution
1117 - Output ONLY the numbered plan as plain text (no formatting, headers, or explanations)
1118
1119
1120
1121 A.2 SUBAGENTS
1122
1123 A.2.1 BOAD TOP 2 DISCOVERED SUBAGENT CONFIGURATIONS FOR
1124 SEED-OSS-36B-INSTRUCT
1125
1126
1127 issue_analyzer configuration
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3
```

1134
 1135 CRITICAL: When you are finished, immediately call `submit_subagent`. Do not call any other tools or
 → produce additional output. Focus exclusively on your assigned task and strictly follow these
 → instructions. Do not attempt to address unrelated parts of the PR or perform work outside your
 → specific subagent role.
 1137 Use the `submit_subagent` tool after you are finished with your specific task to provide a clear and
 → complete summary of your findings or changes.
 1138 Your thinking should be thorough and so it's fine if it's very long.
 1139

1140

1141 **code_navigator configuration**

1142 **docstring:**
 1143 [subagent] Explores and maps relevant code structure in large repositories. Outputs structured
 → information about key files, functions, classes, and their relationships. Repository state
 → unchanged - only reads and analyzes code without modifications.
 1144
 1145 **argument:**
 1146 context (string) [required] : A string containing the issue description, error messages, stack traces,
 → or specific code elements to investigate. Should include any relevant file paths, function names,
 → class names, or keywords that might help locate the problematic code.
 1147
 1148 **instance template:**
 1149 Your task: Explore and map the relevant code structure in the repository based on the provided context:
 → {{context}}
 1150 Follow these steps:
 1. Parse the provided context to identify key elements to investigate (file paths, function names,
 → class names, error messages, etc.)
 2. Navigate through the repository structure to locate relevant files and directories.
 3. Examine and analyze the identified code elements including: - Key files and their purposes -
 → Important functions and their signatures - Classes and their methods/attributes - Module
 → dependencies and imports - Code relationships and call hierarchies
 4. Map the structure and relationships between different code components
 5. Provide structured information including: - File locations and their roles in the codebase -
 → Function/class definitions and their responsibilities - Dependencies between modules/components -
 → Code patterns and architectural insights - Relevant code snippets that relate to the context
 **CRITICAL: When you finish your analysis, immediately call `submit_subagent` with a comprehensive
 → summary of your findings.**
 CRITICAL: STAY IN YOUR LANE - You are ONLY responsible for your specific assigned task - You are
 → NOT responsible for solving the entire issue - You are NOT responsible for other subagent tasks -
 → Focus EXCLUSIVELY on your assigned task and nothing else - Do not attempt to address unrelated
 → parts of the PR or perform work outside your specific subagent role
 Use `submit_subagent` to provide a clear and complete summary of your findings when finished.

1161

1162

1163 **A.2.2 ALL BOAD DISCOVERED SUBAGENTS**

1164 Here we report all sub-agents discovered over 100 iterations of BOAD. For each sub-agent, we report
 1165 the iteration number where it was first proposed (Generated Iteration), the number of times it was
 1166 selected during optimization (n), and its final hindsight helpfulness rate.
 1167

Subagent	Generated Iteration	n	Helpfulness
code_navigator	1	1140	0.933
test_runner	1	120	0.625
code_fixer	1	36	0.361
fix_validator	2	36	0.333
issue_reproducer	3	564	0.768
dependency_resolver	5	24	0.125
test_analyzer	6	24	0.083
issue_analyzer	7	1116	0.982
precision_editor	8	120	0.642
multi_file_coordinator	10	24	0.042
code_detective	11	60	0.500
config_manager	14	12	0.000
git_resolver	26	12	0.000
error_debugger	32	12	0.000
api_analyzer	36	24	0.208
performance_analyzer	46	36	0.250
compatibility_checker	48	36	0.333
spec_analyzer	53	36	0.361
data_flow_analyzer	60	96	0.562
refactor_architect	96	24	0.042

1186

1187

Table 4: Subagent statistics sorted by `exp_num` after 100 iterations of BOAD

1188
1189

A.2.3 ALL EVOLUTION DISCOVERED SUBAGENTS

1190
1191
1192
1193
1194

Here we report all bundles discovered over 20 iterations of BOAD (one for each iteration). For each sub-agent, we report its iteration number, the generated subagents, and the bundle’s success rate. Note that sub-agents with the same name across different iterations may have different system/instance templates because sub-agents are not reused. On evaluation, we choose the bundle with the highest success rate (latest bundle if tied).

1195
1196
1197
1198
1199

Sub-agents focused on **problem analysis or file localization**—such as `issue_analyzer` (0.968 average helpfulness), `code_navigator` (0.917), and `issue_reproducer` (0.817)—consistently appear more useful. Our observation is that these agents provide value *independently* of how later stages unfold: even if code editing or testing fails, identifying the correct files and clarifying the underlying issue is almost always beneficial.

1200
1201
1202
1203

In contrast, sub-agents whose usefulness depends on earlier steps tend to show lower average helpfulness. For example, `test_analyzer` (0.167) is only useful *after* the system has already identified the faulty files and produced a candidate patch; if either prerequisite fails, it cannot contribute, which naturally lowers its average helpfulness.

1204
1205
1206

Specialized sub-agents such as `dependency_resolver` (0.250) and `config_manager` (0.000) also have low average helpfulness, largely because dependency or configuration issues appear in only a small fraction of tasks.

1207
1208
1209
1210

Finally, once core analysis is completed and the faulty files are correctly located, the orchestrator can often complete the remaining code-editing steps on its own. This explains why code-editing sub-agents like `code_fixer` do not exhibit high average helpfulness.

1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229

Iteration	Subagents	Success Rate
1	<code>issue_localizer</code> , <code>issue_reproducer</code> , <code>patch_editor</code>	0.333
2	<code>fix_validator</code> , <code>fix_planner</code> , <code>code_explorer</code>	0.500
3	<code>issue_localizer</code> , <code>test_runner</code> , <code>code_patcher</code>	0.417
4	<code>repo_mapper</code> , <code>code_searcher</code> , <code>fix_validator</code>	0.500
5	<code>code_editor</code> , <code>issue_reproducer</code> , <code>issue_localizer</code>	0.182
6	<code>fix_validator</code> , <code>code_locator</code> , <code>problem_analyzer</code>	0.583
7	<code>test_runner</code> , <code>code_editor</code> , <code>code_analyzer</code>	0.583
8	<code>code_locator</code> , <code>fix_validator</code> , <code>issue_reproducer</code>	0.417
9	<code>code_editor</code> , <code>repo_analyzer</code> , <code>test_runner</code>	0.250
10	<code>code_locator</code> , <code>issue_reproducer</code> , <code>solution_planner</code>	0.417
11	<code>code_patcher</code> , <code>test_validator</code> , <code>repo_explorer</code>	0.417
12	<code>issue_reproducer</code> , <code>code_locator</code> , <code>fix_implementer</code>	0.364
13	<code>fix_validator</code> , <code>code_analyzer</code> , <code>code_editor</code>	0.500
14	<code>issue_reproducer</code> , <code>code_locator</code> , <code>bug_analyzer</code>	0.417
15	<code>test_validator</code> , <code>patch_editor</code> , <code>repo_explorer</code>	0.500
16	<code>code_locator</code> , <code>issue_reproducer</code> , <code>bug_analyzer</code>	0.455
17	<code>test_validator</code>, <code>patch_editor</code>, <code>codebase_explorer</code>	0.583
18	<code>issue_reproducer</code> , <code>bug_localizer</code> , <code>bug_analyzer</code>	0.250
19	<code>code_patcher</code> , <code>fix_validator</code> , <code>issue_analyzer</code>	0.417
20	<code>code_locator</code> , <code>reproducer</code> , <code>code_editor</code>	0.417

1230
1231
1232

Table 5: Subagents and success rates per iteration for the evolutionary baseline.

1233
1234
1235
1236
1237
1238
1239
1240
1241

A.2.4 MANUALLY DESIGNED SUBAGENT CONFIGURATIONS

issue_localizer configuration

1236
1237
1238
1239
1240
1241

```

docstring: A subagent that localizes the issue in the repository. Takes a context string specifying the
→ brief description of the issue. Outputs a brief report about which files and lines are relevant to
→ the issue.

argument: context (string) [required] { A string containing the brief description of the issue.

instance template:
Issue description:
{{context}}

```

```

1242
1243     Please identify which files and specific lines or functions are most relevant to this issue. Output a
1244     ↪ short, clear report that mentions:
1245     - File paths
1246     - Line numbers or function names
1247     - A one-sentence explanation for why each location is relevant
1248
1249
1250
1251     Keep the report concise and focused on helping later agents work on the correct parts of the
1252     ↪ repository.

```

1251 error_reproducer configuration

```

1252
1253     docstring: A subagent that creates and executes test scripts to verify reported errors. Outputs the
1254     ↪ result of the tests and locations of test files created.
1255
1256     argument: context (string) [required] { A string containing error details, file paths and line numbers
1257     ↪ of code relevant to the error, and expected vs actual behavior.
1258
1259     instance template:
1260     Error context:
1261     {{context}}
1262
1263     Please create and execute a temporary reproduction script to verify this error. You should:
1264     - Create temporary files (prefixed with 'tmp_')
1265     - Include only what's needed to reproduce the error
1266     - Report whether the error reproduces exactly as described
1267     - Note any deviations from expected behavior
1268
1269     Output a short, clear report that mentions:
1270     - Result of the tests
1271     - Locations of test files created

```

1265

1266

1267

1268 code_editor configuration

```

1269
1270     docstring: A subagent that implements specified code changes in the repository. Outputs the specific
1271     ↪ code changes made, file paths/line numbers edited, and what should be tested to verify the fix.
1272
1273     argument: context (string) [required] { A string containing the code snippet(s) to modify and file
1274     ↪ path(s), and the changes to be applied.
1275     instance template:
1276     Context for code changes:
1277     {{context}}
1278
1279     Please implement the specified code changes in the repository. You should:
1280
1281     - Identify the relevant files and code sections
1282     - Make precise edits according to the specification
1283     - Maintain code quality and consistency
1284     - Output a short, clear report that mentions:
1285     - File paths/line numbers edited
1286     - What should be tested to verify the fix

```

1280

1281

1282

1283 code_tester configuration

```

1284
1285     docstring: A subagent that tests code after edits have been made to verify the fix works correctly.
1286     ↪ Outputs the result of the tests.
1287
1288     argument: context (string) [required] { A string containing the specific code changes made, file paths,
1289     ↪ and original error.
1290
1291     instance template:
1292     Code changes made:
1293     {{context}}
1294
1295     Please test the code after the edits to verify the fix works correctly. You should:
1296
1297     Use existing test files if available (prefixed with 'tmp_'), or create new ones as needed
1298
1299     - Test the specific functionality that was changed
1300     - Determine whether or not the original error is fixed.
1301     - Output a short, clear report that mentions:
1302     - List of tests run and results of the tests
1303     - Whether the original error is fixed, and if any new errors were introduced

```

1296 A.3 ADDITIONAL EXPERIMENTAL RESULTS
12971298 A.3.1 EFFECT OF DESIGN-SET SIZE
12991300 We evaluated whether the size of the design set affects the performance of the discovered sub-agents
1301 and found no meaningful impact. We sample 6 unique problems from SWE-Bench-Verified and
1302 sample one problem from each to make a small design set of 6 problems, and sample two problems
1303 from each to make a large design set of 24 problems. Across different design-set sizes, the resulting
1304 sub-agents achieve similar performance as shown in Table 6.
1305

Design Set Size	Resolution Rate
6	21%
12	20%
24	19%

1310 Table 6: Performance of discovered sub-agents across different design-set sizes.
13111312
1313 A.4 ADDITIONAL EXPERIMENTAL SETUP DETAILS
13141315 A.4.1 IMPLEMENTATION DETAILS FOR EVOLUTIONARY BASELINE
13161317 For comparison with BOAD, we implemented an evolutionary multi-agent design baseline where the
1318 orchestrator and three sub-agent prompts are bundled together and treated as a single evolutionary
1319 individual. We use three sub-agents paired with an orchestrator because BOAD chooses three sub-
1320 agents when optimizing the sub-agents in the experiments (Section 5.2). At each iteration of the
1321 evolution, the system generates three new subagents sequentially, given the previous round subagent
1322 configurations and their measured helpfulness and success rates (using the same prompts as BOAD,
1323 provided in A.1.2). Next, both the sub-agent warm-up and the orchestrator construction follow the
1324 same procedures used in BOAD. The generated orchestrator and subagents are then run on the same
1325 design set to get the helpfulness and success rate for the next iteration.
1326

A.4.2 API COST COMPARISON WITH BOAD

1327 The evolutionary baseline also requires Claude calls—both to propose new orchestrator/sub-agent
1328 prompts and to perform LLM-as-judge scoring. Under the same setup, each evolution iteration costs
1329 \$2.33, whereas each BOAD iteration costs \$0.96, indicating that evolution is substantially more
1330 expensive to run. Evolution incurs higher cost because it mutate the bundle of orchestrator and
1331 sub-agents at each iteration.
1332

B THE USAGE OF LARGE LANGUAGE MODELS (LLMs)

1333 In our work, we used large language models (LLMs) for two purposes: (1) as a general writing
1334 assistant to check the grammar of the manuscript for readability, and (2) as the model component of
1335 our proposed system BOAD, where LLMs function as the evolution engine, meta-/sub-agents, and
1336 evaluation judges in experiments with mainstream coding agents. All research ideas, methodological
1337 contributions, and conclusions are solely those of the authors, who take full responsibility for the
1338 content of this work.
1339