

Multi-Round Audit Convergence (MRAC): Search-Space Coverage Assessment and Second-Order Observation for Complex AI Engineering Tasks

Author: Liu Ming

Date: August 14, 2026

Version: V2.1.9

Status: Final Manuscript

Abstract

Large language models and their agents can already handle some real-world software engineering tasks, yet a single successful execution is still insufficient to demonstrate that a specific result has reached a level at which it can be delivered and relied upon. Prior work proposed the SCR relationship (Search Space–Coverage Capability–Delivery Reliability), arguing that delivery reliability depends on how well the coverage capability of the complete AI development system matches the task search space. This matching relationship, however, cannot be observed directly. Engineering systems therefore need to infer it from evidence produced during execution and verification.

This paper first analyzes the relationship among execution, independent audit, and repair. After execution produces a candidate result, an independent audit can search the concrete engineering object for omissions, conflicts, and verification gaps while operating within a problem scope that is more constrained than the original generation task. Once identified issues are repaired, the next audit operates on an updated engineering object. Execution, audit, and repair are therefore not simple repetitions of the same problem. Instead, they allow limited coverage capability to act on different problems in stages, making it possible for the complete system to handle some tasks beyond the range that a single execution can cover reliably.

On this basis, this paper proposes **Multi-Round Audit Convergence (MRAC)**. Each audit round produces findings on the current engineering object; successive audit, repair, and re-audit rounds further form a trajectory describing changes in the number, severity, and types of newly discovered issues. MRAC performs **second-order observation** on this multi-round issue trajectory: when newly discovered significant issues decrease overall and the trajectory gradually stabilizes, it exhibits convergence; when new significant issues continue to emerge despite repeated repairs, the trajectory exhibits non-convergence. The significance of multiple rounds is not simply to increase the number of audits, but to create the cross-round object required for this second-order observation.

What MRAC observes directly is the issue trajectory. In conjunction with SCR, when task boundaries, audit protocols, and verification conditions remain comparable, and later audits retain an effective ability to discover remaining significant issues, convergence can serve as positive process evidence that the current system has sufficient coverage capability to handle the task consistently. It cannot prove that the result is absolutely correct, nor can it rule out shared blind spots. Persistent non-convergence more directly indicates that the current system has not demonstrated that it can handle the task consistently. The two forms of evidence are therefore asymmetric: non-convergence is relatively strong negative evidence, whereas convergence depends on the additional condition that auditing remains effective.

As of the data cutoff for this paper, the current project's MRAC observation records cover 35 tasks or subtasks resulting from decomposition. Of these, 33 have reached the project's adopted convergence condition,

completed verification, and been merged into the main branch; 2 remain in progress. In addition, 3 parent tasks triggered decomposition after persistent non-convergence within their original boundaries, producing 10 subtasks in total. All of these subtasks reached the convergence condition and completed verification, after which the 3 parent tasks were closed. No case has yet been observed in the current records in which, after decomposition, persistent non-convergence of a subtask prevented closure of the overall task. This small sample cannot prove that any engineering task can always be made to converge through recursive decomposition, nor can it by itself prove that true delivery reliability improves correspondingly. It does, however, show that the process of “non-convergence → decomposition → renewed convergence” has repeatedly occurred in multiple real engineering tasks.

This paper therefore positions MRAC as a mechanism for search-space coverage assessment and task adjustment built on top of the SCR relationship. The evidentiary requirements and permissible scope of extrapolation for claims at different levels are left to subsequent MRAC L1–L5 research.

Keywords: Multi-Round Audit Convergence; SCR; second-order observation; search space; coverage capability; delivery reliability; independent audit; task decomposition; software engineering agents

1. Introduction

Prior work proposed the SCR relationship (Search Space–Coverage Capability–Delivery Reliability): the delivery reliability of an AI task depends on how well the coverage capability of the complete AI development system matches the task search space [1]. Existing research has shown that language-model agents can autonomously edit files, run tests, and solve some software engineering tasks in real repositories [3]. However, a locally successful execution or partial passage of known validations is still insufficient to establish that the current task has been adequately covered.

This basic relationship leaves two direct engineering questions.

First, the match between coverage capability and search space cannot be observed directly. Engineering systems usually cannot enumerate all critical conditions in advance, nor can they determine whether models, tools, and verification feedback have already covered the decisive risks to delivery.

Second, if a single execution is insufficient to handle the current task consistently, why might independent audit and repair improve the situation? And how can successive audit–repair cycles provide information beyond single-round issue findings for judging whether the system has become capable of handling the task consistently?

This paper first analyzes why execution, independent audit, and repair face different problems, and then introduces the core observation mechanism of MRAC: each audit round produces first-order issue findings; multi-round results form an issue trajectory; and MRAC performs second-order observation on changes across rounds. The paper then explains under what observability conditions, based on SCR, convergence and non-convergence can be interpreted as process evidence about the match between coverage capability and search space, and discusses capability enhancement and task-boundary adjustment after persistent non-convergence.

2. Theoretical Premise: Execution, Audit, and Repair Face Different Problems

2.1 Search Space, Coverage Capability, and Delivery Reliability

This paper adopts the SCR relationship (Search Space–Coverage Capability–Delivery Reliability) proposed in prior work [1]. That study treats the complete AI development system as the object of analysis, including the foundation model, development agent, retrieval and editing tools, codebase and execution environment, tests and other verification feedback, and the multi-round correction process available within a given budget.

Its three basic concepts are:

- **Search Space (S):** the relevant range of possibilities surrounding requirement interpretation, engineering state, implementation paths, and verification paths when completing a specific task;
- **Coverage Capability (C):** the ability of the complete AI development system, under a given task type, execution budget, and feedback conditions, to consistently identify critical conditions, distinguish candidate paths, eliminate decisive errors, and complete sufficient verification;
- **Delivery Reliability (R):** the degree to which a specific result satisfies critical requirements, has its relevant impacts sufficiently verified, remains valid under expected conditions, and can therefore be trusted as a completed delivery.

Their basic relationship is:

The delivery reliability (R) of an AI task depends on the degree of match between system coverage capability (C) and task search space (S).

Accordingly, there are two fundamental ways to improve delivery reliability: expand coverage capability, or govern the search space so that the task enters a range the current system can handle consistently [1].

This paper does not re-argue SCR. Instead, it further examines why execution, audit, and repair can change the problems the system must subsequently handle, and how this continuous process can provide process evidence for judging whether coverage capability and search space are sufficiently matched.

2.2 The Observability Problem of Coverage

The match between coverage capability and search space cannot be inferred directly from the fact that code has been generated, that some tests have passed, or that one audit round has completed. Engineering systems usually cannot enumerate all critical conditions in advance, nor can they confirm that current verification methods cover every decisive delivery risk.

In practice, judgments can only rely on proxy evidence such as test results, runtime feedback, static checks, audit findings, and issue-repair trajectories.

One important property of the audit–repair process is that it can produce multi-round records with version continuity. The system can therefore move from observing “what was found in this round” to observing “how newly discovered issues change after repeated repair.” To understand the meaning of this trajectory, execution and audit must first be distinguished as different problems.

2.3 Execution and Independent Audit Face Different Problems

A single execution faces a relatively open-ended generation problem.

To complete an engineering task, the system may need to handle simultaneously:

- interpretation of requirements and intent;
- identification of the current state of the codebase;
- identification of relevant modules, dependencies, and areas of impact;
- selection among multiple implementation paths;
- boundary conditions and exceptional paths;
- verification methods and verification scope.

Before a candidate result has been produced, these possibilities collectively constitute the search space that must be handled during execution.

Once execution has produced a concrete engineering object, the problem faced by subsequent audit changes. An audit usually does not need to search again for all possible implementations. Instead, it can inspect the existing result for:

- violations of known requirements;
- omitted necessary conditions;
- impacts of changes on existing behavior;
- obvious gaps in the current tests;
- conflicts with existing architecture, interfaces, or data contracts;
- areas that still lack sufficient evidence to support reliable delivery.

The possibilities in the original task have not disappeared, but the candidate result provides a concrete object that further constrains what the audit must actively search. The open-ended problem of “finding a solution” is partly transformed into the more constrained problem of “finding evidence of failure” in the existing result.

In this paper, **independent audit** mainly refers to an operational separation between the audit objective and the execution path: the audit is not merely a continuation of the original execution process, but reframes the search around the task requirements, the current engineering object, and the verification evidence, with the explicit goal of discovering failures, omissions, and boundary gaps.

This independence is not statistical independence, nor does it require the use of a different model. Different sessions, reloading task conditions, different audit prompts, different audit perspectives, or different models can all increase the degree of separation between execution and audit; their actual effects still require experimental determination.

Therefore:

Execution searches for a candidate solution; given that candidate solution, independent audit searches for evidence that it cannot yet be trusted.

This allows the system to concentrate limited coverage capability on a problem that is more explicit and more constrained than the original generation task.

2.4 Why Each Audit Round Must Be Followed by Repair

Discovering issues through audit does not itself change the engineering object.

If multiple rounds audit the same result without repair, the main outcome is repeated checking of a fixed object by the auditor. This may increase the chance of discovering issues, but it cannot reflect how the engineering object changes after continued repair.

Therefore, to form an interpretable MRAC issue trajectory, issues discovered in one audit round need to receive the necessary treatment before the next round begins:

Current object → audit findings → repair → updated object

The next audit operates on the updated version and continues checking unresolved or newly exposed issues. The process can continue as:

Execution → independent audit → repair → re-audit → repair again → re-audit

Intermediate repairs ensure that cross-round changes reflect not only repeated sampling by the auditor, but also continued changes in the engineering object itself. Yet issue counts need not decrease monotonically. A repair can introduce new defects, affect previously correct behavior, expose deeper issues that were hidden by earlier problems, or gradually move the result away from the original objective.

Accordingly, the audit–repair cycle first provides an engineering process capable of producing a multi-round issue trajectory; it does not presuppose convergence.

2.5 Extending Beyond the Coverage Range of a Single Execution

The complete system does not require a single execution to cover all critical conditions in the original task simultaneously.

Execution first produces a candidate result; audit searches that concrete object for evidence of failure; repair then produces a new engineering state. Limited coverage capability can therefore act on different problems in stages.

As a result, an execution–audit process may be able to handle some tasks beyond the range that a single execution can cover reliably. This does not mean that the foundation model itself has acquired new intrinsic capability. The applicable range and actual magnitude of improvement still require empirical study.

3. Multi-Round Audit Convergence

3.1 Definition: First-Order Audit Results and Second-Order Observation

Multi-Round Audit Convergence (MRAC): under the same task boundary, repeatedly auditing, repairing, and re-auditing an engineering object across successive versions, and determining whether the issue trajectory gradually stabilizes by observing changes in the number, severity, and types of newly discovered valid issues across rounds.

Here, a “valid issue” refers to an issue that, after necessary deduplication and confirmation, affects requirement satisfaction, engineering boundaries, implementation correctness, change impacts, or verification sufficiency. Wording differences, duplicate reports, and style suggestions that do not affect delivery judgment should not be mechanically counted as new issues.

MRAC involves two levels. Each audit round first produces issue findings on the current engineering object, such as requirement omissions, implementation errors, or verification gaps. These are **first-order audit results**. When the first-order results from multiple rounds are arranged according to the audit–repair sequence, they form a new object of observation: the **multi-round issue trajectory**.

MRAC then observes changes across these first-order results, such as whether the number of new issues is decreasing, whether high-severity issues continue to emerge, and whether new issue types keep appearing. This paper calls such cross-round observation **second-order observation**. “Second-order” here means only a

further observation of the relationships among first-order audit results; it does not denote a second-order quantity in statistics, calculus, or any other formal system.

Thus:

Each audit round produces a first-order result; multi-round first-order results form an issue trajectory; MRAC performs second-order observation on that trajectory.

The significance of “multiple rounds” lies not in simply increasing the number of audits, but in forming this object of observation.

3.2 Multi-Round Issue Trajectories

MRAC tracks at least the following changes in an issue trajectory:

- whether newly discovered valid issues decrease overall;
- whether high-severity issues continue to appear;
- whether new issue types and boundary conflicts continue to emerge;
- whether repairs continue to expose new significant issues;
- whether issues gradually concentrate into a small number of low-impact details;
- whether the overall trajectory tends toward stability.

These signals arise from relationships among rounds rather than from any single audit result. A round with no new findings, the reappearance of a high-severity issue in one round, or a sharp drop in issue count in a particular round cannot, in isolation, determine the MRAC state apart from the complete trajectory.

Therefore, the additional information in MRAC comes from the **cross-round relationships themselves**, not from simply accumulating more single-round results.

3.3 Convergence and Non-Convergence: Two States of the Issue Trajectory

If, after successive audits and repairs:

- newly discovered significant issues decrease overall;
- high-severity issues gradually decrease;
- new issue types no longer continue to appear;
- multi-round results eventually exhibit a stable trend;

then the issue trajectory exhibits **convergence**.

MRAC does not require issue counts to decrease strictly monotonically. In real engineering work, repairs can expose previously invisible issues, and one round may have no new findings while a later round discovers a significant issue. Convergence describes the overall multi-round trend and eventual stabilization, not mandatory improvement between every pair of adjacent rounds.

Conversely, if, after continued audits and repairs:

- new significant issues continue to appear;
- high-severity issues persist for a long time;
- new issue types, boundary conflicts, or omissions keep emerging;
- no stable trend appears within the predefined observation range;

then the issue trajectory exhibits **non-convergence**.

Convergence and non-convergence are descriptions of multi-round issue trajectories, not classifications of individual audit rounds.

3.4 From Issue Trajectory to Coverage Assessment

What MRAC directly yields is a multi-round issue trajectory. Interpreting what that trajectory means for coverage status requires returning to the relationship among search space, coverage capability, and delivery reliability described by SCR.

A key **observability premise** is required: if significant conditions remain in the task that the current system has not yet handled, later audits that remain effective should retain a non-negligible chance of exposing them as newly discovered valid issues. This premise does not require audits to find every remaining error; it only requires that auditing has not degenerated into a repetitive process that can no longer expose significant issues.

When task boundaries and audit conditions remain comparable, audit rounds are valid, and the above premise holds, then if newly discovered significant issues decrease overall and tend toward stability after continued repair, such convergence can be treated as conditional positive process evidence supporting the claim that:

The current system has sufficient coverage capability to handle this task consistently.

This interpretation remains conditioned on the specific model, tools, context, audit direction, and verification conditions. Convergence cannot prove that the result is absolutely correct, nor can it rule out a blind spot shared by multiple rounds. If later audits have lost the ability to discover remaining significant issues, apparent convergence cannot be reliably interpreted as evidence of sufficient coverage.

Non-convergence has a different evidential character. If the engineering object continues to be repaired while new significant issues keep appearing, the system is still exposing important conditions that had not previously been handled. When issue findings are valid, the task boundary is stable, and the repair process is functioning normally, persistent non-convergence provides relatively strong support for the claim that:

The current system still cannot consistently handle this task.

The two forms of evidence are therefore not fully symmetric: **persistent non-convergence** provides relatively strong grounds for rejecting the claim that “the current system already has stable coverage of this task,” whereas **convergence** can serve as positive process evidence for sufficient coverage only when the audit continues to retain effective discovery capability.

MRAC observes the issue trajectory; coverage status is a further interpretation of that trajectory based on SCR. The two should be kept distinct.

3.5 Conditions for Interpreting Second-Order Observation as Coverage Evidence

For MRAC's second-order observation to support a consistent interpretation, the following basic conditions are required.

First, the task boundary should remain relatively stable.

If goals and scope keep changing across rounds, issues may arise from changes to the task itself and cannot be interpreted directly as convergence or non-convergence on the same task.

Second, issues found in each round should receive the necessary treatment before the next round.

If the engineering object does not change in response to audit findings, multi-round results mainly reflect repeated inspection of the same object rather than a trajectory after continued repair.

Third, the core audit protocol should remain comparable.

Audit scope, issue criteria, and basic verification conditions should remain stable or change according to predefined rules. If the model, prompt, context-loading method, or audit perspective is intentionally changed, those changes should be recorded so that different rounds remain interpretable and comparable.

“Comparable” does not require every round to use exactly the same model or prompt. Predefined multi-perspective, cross-model, or strategy variations can also form a valid trajectory, provided the changes are explicitly recorded and can be interpreted.

Fourth, the audit rounds themselves must be valid.

Rounds affected by tool failure, missing context, abnormal output, or incomplete execution of the intended audit scope cannot be interpreted as normal issue-discovery results.

Fifth, later audits should retain an ability to discover remaining significant issues.

This is the critical observability condition for interpreting issue trajectories as evidence about coverage status. If multiple rounds repeatedly follow the same path and share the same blind spot, then even if the trajectory appears stable, its positive interpretation is substantially weakened.

MRAC therefore does not mechanically equate “trajectory stabilization” with “task correctness.” It requires the task boundary, audit protocol, repair process, and continuing discovery capability to be constrained and recorded together.

3.6 Operationalizing MRAC Decision Rules

The theoretical definition of MRAC does not require all tasks to use a single universal threshold. Different task types, error severities, and audit methods may use different engineering decision rules.

A concrete implementation should specify in advance at least:

- which issues count as valid issues;
- how issue severity is classified;
- how many rounds or what observation window is used to judge the trend;
- the allowable threshold for newly discovered issues;
- how many consecutive stable rounds are required;
- which verification conditions must also be satisfied.

These parameters may vary across tasks, but they should be determined before observation or changed according to predefined rules, while preserving an interpretable basis for comparison across rounds.

MRAC is therefore a parameterized framework for second-order observation, not a fixed stopping number universally applicable to all tasks.

4. Task Adjustment After Non-Convergence

4.1 From Non-Convergence to Search-Space Governance

If MRAC persistently exhibits non-convergence, then according to the interpretation in Section 3, the current system is not yet able to handle the task consistently.

Under SCR, there are two basic directions:

1. **Enhance coverage capability:** replace or enhance the model, tools, context, verification methods, and related components;
2. **Govern the search space:** change task boundaries so that the task enters a range that the current system can handle more consistently with its existing coverage capability.

This paper focuses on task decomposition as one form of the second approach.

A larger engineering task may simultaneously contain multiple groups of requirement interpretations, engineering states, dependencies, implementation choices, and verification paths, making the overall task difficult for the current system to handle consistently. If the original boundary remains non-convergent, continuing to add local patches may not resolve the mismatch and may instead introduce new constraints and issues. At that point, work under the original task boundary can be stopped and the higher-level objective revisited for decomposition.

Decomposition is not simply a mechanical division of work into smaller text or code fragments. Its goal is to reduce, as far as possible, the state, dependencies, constraints, decisions, and scope of impact that each subtask must handle simultaneously.

Each subtask is then executed again within its new boundary and subjected to MRAC. If it converges, the process can continue; if it remains non-convergent, the task can be decomposed further or the system can turn to coverage-capability enhancement. After multiple subtasks are completed, their combined result still requires the necessary audit and verification.

This process can be summarized as:

Persistent non-convergence → adjust task boundaries → re-execute and run MRAC within the new boundaries → make the next decision based on the new trajectories.

When a task can continue to be decomposed meaningfully, this process can occur recursively.

MRAC does not require a fixed form of audit object. It can be applied to task plans, requirement specifications, designs, code changes, or other engineering objects that can maintain relatively stable boundaries while supporting continued repair. The Specs and PRs in the cases reported here are only two practical forms.

4.2 How to Determine Whether Decomposition Is Effective

Decomposition does not by itself guarantee that a task becomes easier to handle. Poor boundaries can increase cross-task dependencies, redundant judgments, integration conflicts, state-synchronization costs, and verification burden.

The effectiveness of decomposition therefore cannot be judged only by the number or apparent size of subtasks. It should also be evaluated by whether MRAC trajectories improve under the new boundaries. If the original task remains persistently non-convergent while the decomposed subtasks each exhibit convergence,

this provides positive process evidence that the new task boundaries have improved the match between coverage capability and search space.

If the decomposed tasks remain persistently non-convergent, the system should continue adjusting the decomposition, decompose further, or turn to stronger models, tools, and verification capabilities. If further decomposition no longer reduces the amount of state and dependency that must be handled simultaneously, or the task can no longer be decomposed meaningfully, the system should adjust system capabilities or objectives, hand the task to humans, or terminate it rather than recurse indefinitely.

A further distinction is necessary: improvement in MRAC trajectories after decomposition directly shows that the new task boundaries are easier for the current system to handle consistently. Whether this also improves final delivery reliability still requires independent outcome metrics.

5. Preliminary Engineering Evidence

5.1 Data Characteristics, Overall Sample, and Decision Rule

The following records come from MRAC practice in the author's real software project. The currently retained data include:

- the complete audit prompts used for each task;
- audit results from each valid audit round;
- corresponding issue-repair records;
- metadata linking parent tasks and decomposed subtasks;
- each task's final status and closure reason.

These records contain not only final task states, but also the process from auditing and non-convergence judgment through task decomposition, renewed auditing of subtasks, and closure.

As of the data cutoff for this paper, MRAC observation records cover **35 tasks or subtasks resulting from decomposition**, of which **33** have reached the convergence condition adopted by the project, completed verification, and been merged into the main branch; **2** remain in progress. In-progress tasks are included in the observation denominator to avoid reporting only completed successful samples.

The current project uses:

Two consecutive valid full-scope audits with no newly discovered substantive issues

as a practical stopping condition. This rule is used only for state determination in the engineering cases reported in this paper; it is not part of the theoretical definition of MRAC. Nor does it carry a predefined statistical confidence level: even when undiscovered issues remain, repeated audits may occasionally yield two consecutive rounds with no new findings. Its actual reliability therefore depends on the audit protocol's continuing ability to discover significant issues. The number of stable rounds required under different models, tasks, and audit configurations remains to be experimentally calibrated.

These data come from a single internal software engineering environment. The complete repository and full audit trajectories have not yet been made public and therefore cannot be treated as externally replicated evidence. This paper reports only the representative trajectories and aggregate results needed to support the basic mechanism. The theoretical boundaries, evidence requirements, and research path for claims at different levels are further developed in the MRAC L1–L5 research program [2].

5.2 Example of a Persistently Non-Convergent Trajectory

Task: n306-14 — **Closing the Native-Unit Loop for Gate and Retiring the False Conversion Axis**

This task underwent 20 consecutive valid audit rounds within its original task boundary. For presentation, this paper rennumbers them R1–R20 in the order of valid audits; Round 2 in the original audit records corresponds to R1 here. Issues discovered in each round were repaired accordingly before the next audit round began.

Round	P1	P2	Round	P1	P2
R1	5	3	R11	3	1
R2	4	2	R12	6	0
R3	3	1	R13	4	1
R4	5	1	R14	4	1
R5	4	1	R15	3	1
R6	3	2	R16	2	2
R7	4	0	R17	2	1
R8	3	2	R18	3	1
R9	3	2	R19	2	1
R10	3	2	R20	4	0

Across the 20 rounds, 70 P1 findings and 25 P2 findings were recorded. These counts represent accepted issue findings in each round. This paper does not claim that findings across different rounds are statistically independent, nor does it interpret them as 95 mutually unrelated independent defects.

Viewed across rounds, issue counts fluctuated and sometimes declined locally, but **P1 findings appeared in all 20 valid rounds, with 2–6 P1 findings per round; even in the final round, 4 new P1 findings were still discovered.**

Thus, despite continued repair of the engineering object, new high-severity issues never ceased to appear, and the overall trajectory did not show signs of approaching stability within 20 valid rounds. Under the definition in this paper, the task exhibited clear non-convergence within its original boundary.

The project then stopped continuing work under the original task boundary and decomposed it into four subtasks: n306-32, n306-33, n306-34, and n306-35. All four subsequently reached the project's MRAC convergence condition and completed verification. These results are reported together with two other completed decomposition cycles in the next section.

5.3 Three Completed Decomposition Cycles

The current records contain 3 completed decomposition cycles whose parent tasks have been closed. All three parent tasks were persistently non-convergent within their original boundaries and were decomposed into 10 subtasks in total.

Parent task	Number of subtasks	Subtask result	Parent-task status
n306-4	3	3/3 converged and verified	Closed
n306-14	4	4/4 converged and verified	Closed
n306-29	3	3/3 converged and verified	Closed

Parent task	Number of subtasks	Subtask result	Parent-task status
Total	10	10/10 converged and verified	3/3 Closed

The closure reason recorded for all three parent tasks was:

Split child tasks completed and verified.

The parent–child task relationships are:

- n306-4 → n306-26, n306-27, n306-28
- n306-14 → n306-32, n306-33, n306-34, n306-35
- n306-29 → n306-30, n306-31, n306-36

All related subtasks have been closed, and the parent–child relationships and decomposition metadata have been fully retained. Audit prompts and round-by-round records also remain available for tracing the actual MRAC trajectories, rather than inferring convergence only from the closed state in the task-management system.

As of the data cutoff for this paper, no case has been observed in which an originally non-convergent task was decomposed, yet persistent non-convergence of the subtasks prevented the overall task from being closed. This observation describes only the current record and does not constitute a general guarantee for future tasks or other engineering environments.

5.4 What the Current Evidence Can Support

The current records contain multiple completed decomposition cycles; the overall observation denominator is 35 tasks or subtasks resulting from decomposition, including 2 tasks that remain in progress.

After three persistently non-convergent parent tasks were decomposed, all 10 resulting subtasks reached the convergence condition adopted by the project and completed verification, after which the parent tasks were closed. The current records therefore support a limited but clear observation:

The process “persistent non-convergence → decomposition → rerun MRAC within new boundaries → subtask convergence” has repeatedly occurred in multiple real engineering tasks.

What these records directly support is that, after task boundaries were redefined, the MRAC trajectories improved and the tasks became more tractable for the current system. They do not by themselves prove that final delivery reliability improved; that still requires independent outcome indicators such as post-merge defects, rework, rollback, and behavior in production.

The current data also cannot prove that:

- any engineering task can always be made to converge through recursive decomposition;
- decomposition was the only cause of eventual convergence in these tasks;
- the results can be directly generalized to other projects, models, or engineering environments;
- the 10 subtasks can be treated as 10 statistically independent samples;
- the current sample is sufficient to determine an optimal decomposition strategy or recursion depth;
- MRAC convergence can replace independent evaluation of real delivery outcomes.

Nor does this paper define decomposition adequacy retrospectively by “eventual convergence.” The purpose of decomposition remains to reduce the state, constraints, dependencies, and impact scope that a single task must handle simultaneously; subsequent MRAC is used to determine whether the new boundary actually improves task tractability.

6. Relationship to Prior Work

6.1 Iterative Self-Feedback and Revision

Self-Refine allows the same large language model to successively take on the roles of generation, feedback, and revision, and demonstrates across multiple task types that iterative self-feedback can improve initial outputs [5]. At the same time, Huang et al. show that, on reasoning tasks and without external feedback, intrinsic self-correction by models is unstable and can sometimes worsen results [6]. Together, these studies indicate that repeated generation and revision do not automatically imply reliable improvement; the source of feedback and the verification conditions affect the outcome of iteration.

MRAC shares the iterative form of “feedback → revision → re-checking” with this line of work, but its object of study is different. Self-Refine mainly asks whether outputs improve after feedback; this paper focuses on the relationships among issue findings from different rounds, and on whether multi-round issue trajectories can serve as the object of observation for coverage assessment.

MRAC is therefore not a new method for optimizing a single output.

6.2 Research on Iterative Audit Convergence

Calboreanu conducted a nine-round audit case study on the prompt specification of the AEGIS multi-agent orchestration system, consisting of eight documents and approximately 7,152 lines, and reported 51 consistency defects. The number of newly discovered defects by round was 15, 8, 12, 2, 8, 1, 4, 1, and 0, showing a non-monotonic decline and reaching one full-scope audit with no new findings in the ninth round [4]. The audit scope and protocol expanded during the process, so the author explicitly interprets the curve as an observation jointly produced by issue repair and improvement of the audit method, rather than as independent repeated measurement under a fixed audit function.

The published version also carried out partial cross-model replication and consistency checks on a public synthetic specification, but it remained a single case involving a production system and did not include a one-pass full-scope audit control [4]. The study notes that a clean round can establish only internal consistency within the scope of the adopted protocol, not real-world correctness, and it does not measure whether the repairs improved system performance, reliability, or operational outcomes. The paper recommends that future replication use the stricter rule of “two consecutive full-scope audits with no new findings.”

That study provides a public case in which multi-round auditing exhibits non-monotonic convergence. This paper goes further by using SCR to explain why execution, audit, and repair can change the problems that need to be handled subsequently, and by treating the multi-round trajectory itself as a second-order observation object for coverage assessment.

6.3 Specification Revision and Traditional Defect-Trajectory Research

The open-source project *Automated Plan Reviser Pro* (APR) performs multi-round AI revision of complex Specs while retaining round history. Its convergence analysis combines trends in output size, rate of modification, and similarity between adjacent rounds to estimate document stability [7].

APR shows that multi-round specification revision and stability analysis have entered public engineering practice. However, shorter outputs, slower changes, and higher textual similarity mainly indicate that a document is changing less; they do not directly show that significant issues are decreasing, critical constraints are covered, or a specific result can be delivered reliably.

Software engineering research has also used defect-discovery trajectories, reliability-growth models, and capture–recapture methods to estimate remaining defects and discuss stopping conditions for testing or review [8,9].

MRAC shares with these studies the basic idea of extracting additional information from issue records across repeated observations, but it focuses specifically on **execution–independent audit–repair** processes involving LLMs: each audit round forms a first-order record, multi-round records form an issue trajectory, and that trajectory is further used to interpret whether the current AI system can already handle the task consistently.

6.4 Position of This Paper

This paper does not claim to be the first to discover the general phenomenon that repeated auditing or revision may tend toward stability. Its main contributions are in three areas:

1. **Basic mechanism:** explaining why, once a candidate result exists, independent audit and repair can gradually change the problems that subsequent rounds need to handle;
2. **Second-order observation:** distinguishing single-round audit results from multi-round issue trajectories, and treating cross-round changes as MRAC's direct object of observation;
3. **Coverage interpretation and task adjustment:** using SCR, under an explicit observability premise, to interpret issue trajectories as process evidence about the match between coverage capability and search space, and using persistent non-convergence to trigger capability enhancement or search-space governance.

The key to MRAC is therefore not merely “auditing more times” or “issue counts eventually declining.” Rather:

In a continuously repaired AI engineering process, MRAC treats the cross-round issue trajectory as a new object of observation and, while auditing retains effective discovery capability, uses it to support judgment of the otherwise unobservable coverage relationship.

The engineering records in this paper are used to show that this mechanism has already appeared in multiple real tasks; they are not intended here to establish broader claims by extrapolation.

7. Limitations and Research Agenda

7.1 Effectiveness of Independent Audit and Shared Blind Spots

Even when auditing is operationally separated from execution, multiple audit rounds using the same model, similar prompts, and similar contexts may still follow the same paths and share blind spots. Nor can the use of different models be assumed to make their error modes truly independent.

Trajectory stabilization may therefore mean that few significant issues remain, or it may simply mean that the current audit method can no longer discover them. Future research needs to compare independent sessions using the same model, different audit perspectives, cross-model auditing, tool-based verification, and human

spot checks; identify which error types different configurations can detect; and test whether greater operational separation between execution and audit yields more valid findings.

7.2 A Candidate Result Does Not Necessarily Reduce Audit Difficulty Substantially

Once execution produces a candidate result, auditing can usually focus on a concrete object, but not every task becomes substantially easier as a result.

If the candidate result is seriously misaligned with the objective, the task requirements themselves are unclear, the audit still needs to reconstruct almost the entire engineering state, or critical failures can only be found through open-ended exploration, the audit may still have to handle a large problem space.

This paper provides a mechanism-level explanation but has not yet directly quantified how much candidate results reduce the difficulty of subsequent audits.

7.3 Protocol Changes, Repair Perturbations, and Goal Drift

If audit prompts, context scope, issue criteria, or focus areas change without being recorded, the multi-round issue trajectory may be affected by changes in the audit protocol itself.

New findings may reflect previously missed issues, but they may also have been introduced by the previous repair. Repeated repairs can also gradually move the engineering object away from the original goal.

Future experiments should therefore keep audit protocols comparable where possible and retain the original task description, initial engineering object, audit results from each round, protocol changes, and repair diffs, so that different sources of change can be analyzed separately.

7.4 Strength of the Current Evidence

The records in Section 5 already include multiple completed decomposition cycles and unfinished tasks, but all data still come from a single software project and engineering workflow, the sample remains limited, and there is no external independent replication.

The current evidence is therefore sufficient to show that the mechanism has repeatedly appeared in real engineering work and is operationally feasible, but not sufficient to support strong extrapolation about the generality of recursive decomposition. The existing data mainly capture MRAC trajectories and process states, and they do not yet prove that MRAC convergence or task decomposition necessarily produces a proportional increase in real delivery reliability.

As a foundational concept paper on MRAC, this paper does not expand the evidence analysis further. The levels of theoretical and evidentiary support required for claims at different levels, including positive and negative evidence and broader questions of whether recursive decomposition can lead to convergence, are developed further in the MRAC L1–L5 research program [2].

7.5 Controls and Evaluation

The engineering value of MRAC needs to be evaluated by separating different mechanisms rather than reducing the question to whether “multi-round audit is better than single-round audit.”

Future work can establish at least the following controls:

1. **No independent audit:** to observe the performance of execution results without audit feedback;

2. **Independent audit and repair:** to measure the engineering value introduced by independent audit itself;
3. **Fixed-round audit–repair:** to generate multi-round issue trajectories without changing the process based on those trajectories;
4. **MRAC:** to use the same type of multi-round trajectory and make decisions based on second-order observation about whether to continue repair, proceed with the task, or adjust task boundaries.

The comparison between fixed-round audit–repair and MRAC is especially important. Both include multiple rounds of auditing and repair, but only MRAC uses cross-round trajectories as a basis for process decisions. This comparison can therefore more directly test whether second-order observation itself adds engineering value.

Auditing configurations using the same model, multiple models, multiple perspectives, or human–AI hybrids can also be compared.

Evaluation metrics should include post-merge defects, error severity, rework volume, rollback rate, delivery time, token and compute costs, the proportion of issues introduced by repairs, and the relationship between trajectory judgments and final delivery outcomes.

7.6 Core Research Questions

Future research can concentrate on four directions:

1. **Independent-audit effect:** under what task conditions does independent audit after candidate generation significantly reduce the difficulty of subsequent handling?
2. **Explanatory power of second-order observation:** to what extent do different issue trajectories support judgments about the match between coverage capability and search space, and under what audit configurations does this interpretation remain valid?
3. **Effect of task decomposition:** after persistent non-convergence, does redefining task boundaries consistently improve MRAC trajectories, and does it further improve independently measured real delivery outcomes?
4. **Engineering efficiency:** how can shared blind spots, repair-induced issues, and audit cost be balanced to increase the net benefit of MRAC relative to other verification methods?

These questions require both theoretical analysis and the continued accumulation of engineering evidence. This paper establishes only the basic mechanism and provides preliminary evidence that it has appeared in multiple real tasks.

8. Conclusion

According to SCR (Search Space–Coverage Capability–Delivery Reliability), the delivery reliability of an AI task depends on the match between coverage capability and search space. This paper proposes that a continuing execution–independent audit–repair process produces an issue trajectory with version continuity, and that MRAC performs second-order observation on this trajectory.

MRAC directly observes the issue trajectory rather than the coverage state itself. When significant issues decrease overall and tend toward stability, the trajectory exhibits convergence; when new significant issues continue to emerge despite continued repair, it exhibits non-convergence. When task boundaries, audit protocols, and verification conditions remain comparable, and later audits retain effective discovery

capability, convergence can serve as positive process evidence that the system has sufficient coverage capability to handle the task consistently; persistent non-convergence provides stronger negative evidence.

When a task remains persistently non-convergent, the system can either enhance coverage capability or adjust the search space through decomposition and rerun MRAC under the new boundaries. In the current engineering records, 33 of 35 tasks or subtasks resulting from decomposition have reached the convergence condition, completed verification, and been merged into the main branch, while 2 remain in progress. Three parent tasks that were persistently non-convergent under their original boundaries were decomposed into 10 subtasks, all of which reached the convergence condition and completed verification. The current evidence supports the operational feasibility of MRAC as a mechanism for search-space coverage assessment and task adjustment, and provides preliminary engineering evidence for “non-convergence → decomposition → renewed convergence.” It does not prove that any task can always be made to converge through recursive decomposition, nor can it replace independent evaluation of real delivery reliability.

The overall relationship can be summarized as:

Execution produces a candidate result → independent audit discovers issues → repair produces a new version → multi-round results form an issue trajectory → MRAC performs second-order observation on the trajectory → SCR is used to interpret coverage status → the system decides whether to proceed, enhance capability, or adjust task boundaries.

This paper establishes the basic concepts, observability premises, and engineering scope and limits of MRAC. The theoretical boundaries and evidence strength of claims at different levels are further developed in the MRAC L1–L5 research program [2].

References

- [1] LIU Ming. *From the SCR Relationship to Search Space Governance: A Foundational Mechanism for Delivery Reliability in AI Software Engineering* [EB/OL]. Zenodo, 2026. DOI: <https://doi.org/10.5281/zenodo.21902420>.
- [2] LIU Ming. *The Theoretical Starting Point and Progressive Research Program of MRAC: From Non-Convergence Diagnosis to Engineering AGI* [EB/OL]. Zenodo, 2026. DOI: <https://doi.org/10.5281/zenodo.21917629>.
- [3] YANG J, JIMENEZ C E, WETTIG A, et al. SWE-agent: Agent-computer interfaces enable automated software engineering [C]//Advances in Neural Information Processing Systems 37. 2024: 50528-50652. DOI: <https://doi.org/10.52202/079017-1601>.
- [4] CALBOREANU E. Iterative audit convergence in LLM-managed multi-agent systems: A case study in prompt-engineering quality assurance [J]. *Software*, 2026, 5(2): 26. DOI: <https://doi.org/10.3390/software5020026>.
- [5] MADAAN A, TANDON N, GUPTA P, et al. Self-Refine: Iterative refinement with self-feedback [C]//Advances in Neural Information Processing Systems 36. 2023. DOI: <https://doi.org/10.52202/075280-2019>.
- [6] HUANG J, CHEN X, MISHRA S, et al. Large language models cannot self-correct reasoning yet [C/OL]//International Conference on Learning Representations. 2024. https://openreview.net/forum?id=Ik_mD3fKBpQ.

- [7] DICKLESWORTHSTONE. *Automated Plan Reviser Pro: Iterative specification refinement tool*, version 1.2.2 [EB/OL]. GitHub repository, accessed 2026-08-13.
https://github.com/Dicklesworthstone/automated_plan_reviser_pro.
- [8] YANG M C K, CHAO A. Reliability-estimation and stopping-rules for software testing, based on repeated appearances of bugs [J]. *IEEE Transactions on Reliability*, 1995, 44(2): 315-321. DOI: <https://doi.org/10.1109/24.387388>.
- [9] BRIAND L C, EL EMAM K, FREIMUT B G, et al. A comprehensive evaluation of capture-recapture models for estimating software defect content [J]. *IEEE Transactions on Software Engineering*, 2000, 26(6): 518-540. DOI: <https://doi.org/10.1109/32.852741>.