

多轮审计收敛（MRAC）：复杂 AI 工程任务的搜索空间覆盖判断 与二阶观测

作者：刘明

日期：2026 年 8 月 14 日

版本：V2.1.9

状态：正式稿

摘要

大模型及其 Agent 已经能够处理部分真实的软件工程任务，但一次执行成功仍不足以证明具体结果已经达到可交付、可依赖的程度。前置研究提出 SCR（Search Space–Coverage Capability–Delivery Reliability Relationship，即“搜索空间—覆盖能力—交付可靠性关系”），指出交付可靠性取决于完整 AI 开发系统的覆盖能力与任务搜索空间的匹配程度。然而，这一匹配关系不能被直接观察，工程系统需要从执行与验证过程中获得判断依据。

本文首先分析执行、独立审计与修复之间的关系。执行形成候选结果后，独立审计可以围绕具体对象寻找遗漏、冲突和验证缺口，在比原始生成任务更受约束的问题范围内工作；问题修复后，下一轮审计面对更新后的工程对象。因此，执行、审计与修复并非对同一问题的简单重复，而是让有限覆盖能力分阶段作用于不同问题，使完整系统有可能处理部分超出单次执行稳定覆盖范围的任务。

在此基础上，本文提出**多轮审计收敛（Multi-Round Audit Convergence, MRAC）**。每轮审计产生当前工程对象上的问题发现；连续的审计、修复和再审计进一步形成关于新增问题数量、严重度和类型变化的轨迹。MRAC 对这一多轮问题轨迹进行**二阶观测**：新的重要问题总体减少并逐步稳定时，轨迹表现为收敛；持续修复后仍不断出现新的重要问题时，轨迹表现为不收敛。多轮的意义不在于简单增加审计次数，而在于形成这一二阶观测对象。

MRAC 直接得到的是问题轨迹。结合 SCR，在任务边界、审计协议和验证条件具有可比性，且后续审计仍对剩余重要问题保持有效发现能力时，收敛可以作为“当前覆盖能力足以稳定处理该任务”的正面过程证据，但不能证明结果绝对正确，也不能排除共同盲区；持续不收敛则更直接地说明当前系统尚未表现出稳定处理该任务的能力。因此，两类证据并不对称：不收敛是较强的负面证据，收敛则依赖审计持续有效这一附加条件。

截至本文数据截点，当前项目的 MRAC 观察记录覆盖 35 个任务或拆分后的子任务，其中 33 个已经达到项目采用的收敛条件、完成验证并合入主线，2 个仍在进行中。另有 3 个父任务在原边界下持续不收敛后触发拆分，共形成 10 个子任务；这些子任务均达到收敛条件并完成验证，3 个父任务随后关闭。当前记录中尚未出现拆分后因子任务持续无法收敛而不能完成闭环的案例。上述小样本不能证明任意工程任务均可通过递归拆分达到收敛，也不能单独证明真实交付可靠性同步提高，但表明“不收敛—拆分—重新收敛”的过程已经在多个真实工程任务中重复出现。

本文据此将 MRAC 定位为 SCR 基础关系之上的一种搜索空间覆盖判断与任务调整机制。不同层级命题的证据要求与外推范围留待后续 MRAC L1-L5 研究讨论。

关键词： 多轮审计收敛；SCR；二阶观测；搜索空间；覆盖能力；交付可靠性；独立审计；任务拆分；软件工程 Agent

1. 引言

前置研究提出 SCR（搜索空间—覆盖能力—交付可靠性关系）：一项 AI 任务的交付可靠性，取决于完整 AI 开发系统的覆盖能力与任务搜索空间的匹配程度[1]。已有研究表明，语言模型 Agent 能够在真实仓库中自主编辑文件、运行测试并解决部分软件工程任务[3]；但局部运行成功或通过部分已知验证，仍不足以说明当前任务已经得到充分覆盖。

这一基础关系留下两个直接的工程问题。

首先，覆盖能力与搜索空间的匹配不能被直接观察。工程系统通常无法预先枚举全部关键条件，也无法确认模型、工具与验证反馈是否已经覆盖决定性交付风险。

其次，如果单次执行不足以稳定处理当前任务，独立审计与修复为什么可能改善这一状态？又如何从连续审计—修复过程中获得单轮问题发现之外的信息，用于判断系统是否已经能够稳定处理任务？

本文先分析执行、独立审计和修复面对的问题为何不同，再提出 MRAC 的核心观测机制：每轮审计产生一阶问题发现，多轮结果形成问题轨迹，MRAC 对跨轮变化进行二阶观测。随后，本文说明在何种可观测性前提下，可以依据 SCR 将收敛与不收敛解释为覆盖能力与搜索空间匹配的过程证据，并讨论持续不收敛后的能力增强与任务边界调整。

2. 理论前提：执行、审计与修复面对的问题

2.1 搜索空间、覆盖能力与交付可靠性

本文沿用前置研究提出的 SCR（搜索空间—覆盖能力—交付可靠性关系）[1]。该研究以完整 AI 开发系统为考察对象，包括基础模型、开发 Agent、检索与编辑工具、代码库和执行环境、测试与其他验证反馈，以及预算范围内的多轮修正过程。

其中三个基础概念分别是：

- **搜索空间 (S)**：完成特定任务时，围绕需求解释、工程状态、实现路径和验证路径展开的相关可能性范围；
- **覆盖能力 (C)**：完整 AI 开发系统在给定任务类型、执行预算和反馈条件下，稳定识别关键条件、区分候选路径、排除决定性错误并完成充分验证的能力；
- **交付可靠性 (R)**：一项具体结果满足关键要求、相关影响得到充分验证，并在预期条件下保持有效，因而能够被作为完成交付加以信赖的程度。

三者的基础关系是：

AI 任务的交付可靠性 (R)，取决于系统覆盖能力 (C) 与任务搜索空间 (S) 之间的匹配程度。

因此，提高交付可靠性有两个基本方向：扩展覆盖能力，或治理搜索空间，使任务进入当前系统能够稳定处理的范围[1]。

本文不重新论证 SCR，而是进一步讨论执行、审计和修复为什么能够改变系统后续需要处理的问题，以及如何从这一连续过程中获得判断覆盖能力与搜索空间是否匹配的过程证据。

2.2 覆盖关系的观测问题

覆盖能力与搜索空间的匹配不能从代码已经生成、测试局部通过或某次审计结果直接推出。工程系统通常无法预先枚举全部关键条件，也无法确认现有验证手段是否覆盖所有决定性交付风险。

实际判断只能依赖测试结果、运行反馈、静态检查、审计发现问题修复轨迹等代理证据。

审计—修复过程的一个重要性质，是它会产生具有版本连续性的多轮记录。系统因此可以从“这一轮发现了什么”进一步观察“持续修复后，新发现的问题如何变化”。要理解这一轨迹的含义，首先需要区分执行与审计面对的问题。

2.3 执行与独立审计面对不同的问题

单次执行面对的是一个相对开放的生成问题。

为了完成一个工程任务，系统可能需要同时处理：

- 需求和意图的解释；
- 代码库当前状态的识别；
- 相关模块、依赖和影响面的定位；
- 多种实现路径之间的选择；
- 边界条件和异常路径；
- 验证方式和范围。

在候选结果产生之前，这些可能性共同构成执行阶段需要处理的搜索空间。

当执行已经形成具体工程对象后，后续审计面对的问题发生了变化。审计通常不需要重新寻找全部实现方式，而可以围绕已有结果检查：

- 当前实现是否违反已知要求；
- 是否遗漏必要条件；
- 修改是否影响其他已有行为；
- 当前测试是否留下明显验证缺口；
- 是否与现有架构、接口或数据契约冲突；
- 哪些部分仍缺少足以支持可靠交付的证据。

原始任务的可能性并未消失，但候选结果已经提供了明确对象，使后续审计需要主动搜索的问题范围进一步受到约束。开放的“寻找一个解”被部分转化为围绕既有结果的“寻找失败证据”。

本文所称的**独立审计**，主要指审计目标和执行路径之间的操作性分离：审计不只是继续沿原执行过程检查，而是重新从任务要求、当前工程对象和验证证据出发，以发现失败、遗漏和边界缺口为目标组织搜索。

这种独立性不等于统计意义上的独立，也不要求必须使用不同模型。不同 Session、重新加载任务条件、不同审计提示、不同审计视角或不同模型，都可以提高执行与审计之间的分离程度，其实际效果仍需实验确定。

因此：

执行寻找候选解；独立审计以候选解为条件寻找其不能被信赖的证据。

这样，系统可以把有限的覆盖能力集中到比原始生成任务更明确、更受约束的问题上。

2.4 为什么每轮审计后需要先修复

审计发现问题本身并不会改变工程对象。

如果连续多轮只审计同一个结果而不修复，得到的主要是审计器对固定对象的重复检查。这可以增加问题发现机会，却不能反映工程对象持续修复后的变化。

因此，为形成具有解释意义的 MRAC 问题轨迹，一轮审计发现的问题需要在下一轮开始前得到必要处理：

当前对象 → 审计发现 → 修复 → 更新后的对象

下一轮审计面对更新后的版本，继续检查尚未解决或新暴露的问题。过程可以连续进行：

执行 → 独立审计 → 修复 → 再审计 → 再修复 → 再审计

中间修复使跨轮变化不仅反映审计器的重复采样，也反映工程对象本身的持续变化。但问题并不必然逐轮减少：修复可能引入新缺陷、影响此前正确的行为、暴露被前置问题遮蔽的更深层问题，也可能使结果逐渐偏离原始目标。

因此，审计—修复循环首先提供的是形成多轮问题轨迹的工程过程，而不是预设的收敛结论。

2.5 对单次执行覆盖范围的扩展

完整系统不必要求一次执行同时覆盖原始任务中的全部关键条件。

执行先形成候选结果，审计围绕具体对象寻找失败证据，修复再生成新的工程状态。有限覆盖能力因此可以分阶段作用于不同问题。

由此，执行—审计流程有可能处理部分超出单次执行稳定覆盖范围的任务。这并不意味着基础模型本身获得了新的固有能力，其适用范围和实际提升幅度仍需实证研究。

3. 多轮审计收敛

3.1 定义：一阶审计结果与二阶观测

多轮审计收敛 (Multi-Round Audit Convergence, MRAC)：在同一任务边界下，对具有版本连续性的工程对象持续执行审计、修复和再审计，通过观察多轮新增有效问题在数量、严重度和类型上的变化，判断问题轨迹是否逐步趋于稳定的现象与判定机制。

这里的“有效问题”是指经必要去重和确认后，会影响需求满足、工程边界、实现正确性、变更影响或验证充分性的问题。措辞差异、重复报告以及不影响交付判断的风格建议，不应机械计入新增问题。

MRAC 涉及两个层次。每轮审计首先产生当前工程对象上的问题发现，例如需求遗漏、实现错误或验证缺口，这些是一阶审计结果。多轮一阶结果按照审计—修复顺序形成连续记录后，构成新的观测对象：**多轮问题轨迹**。

MRAC 进一步观察这些一阶结果之间的变化，例如新增问题是否减少、高严重度问题是否持续出现、新的问题类型是否仍不断产生。本文将这种跨轮观察称为**二阶观测**。“二阶”仅表示对一阶审计结果之间关系的进一步观察，不对应统计学、微积分或其他形式体系中的二阶量。

因此：

每轮审计产生一阶结果；多轮一阶结果形成问题轨迹；MRAC 对这一轨迹进行二阶观测。

“多轮”的意义不在于简单增加审计次数，而在于形成这一观测对象。

3.2 多轮问题轨迹

MRAC 关注的问题轨迹至少包括以下变化：

- 新增有效问题是否总体减少；
- 高严重度问题是否持续出现；
- 是否仍不断出现新的问题类型和边界冲突；
- 修复后是否继续暴露新的重要问题；
- 问题是否逐渐集中到少量低影响细节；
- 整体轨迹是否趋于稳定。

这些信息来自轮次之间的关系，而不是任何一个单独的审计结果。一轮没有发现新问题、某轮重新出现高严重度问题，或某一轮问题数量明显下降，都不能脱离完整轨迹单独决定 MRAC 状态。

因此，MRAC 新增的信息来自**跨轮关系本身**，而不是简单累加更多单轮结果。

3.3 收敛与不收敛：问题轨迹的两种状态

如果连续审计和修复后：

- 新增重要问题总体减少；
- 高严重度问题逐步减少；
- 新的问题类型不再持续出现；
- 多轮结果最终表现出稳定趋势；

则该问题轨迹表现为**收敛**。

MRAC 不要求问题数量严格单调下降。真实工程中，修复可能暴露此前不可见的问题，也可能出现某轮无新增问题、随后又发现重要问题的情况。收敛描述的是多轮轨迹的总体变化及最终稳定，而非相邻轮次必须持续改善。

与此相对，如果持续审计和修复后：

- 新的重要问题仍反复出现；
- 高严重度问题长期存在；
- 新的问题类型、边界冲突或遗漏不断出现；
- 在预定观察范围内始终没有稳定趋势；

则该问题轨迹表现为**不收敛**。

收敛与不收敛都是对多轮问题轨迹的描述，而不是对单轮审计结果的分类。

3.4 从问题轨迹到覆盖判断

MRAC 直接得到的是多轮问题轨迹。要解释这一轨迹对覆盖状态意味着什么，需要回到 SCR 所描述的搜索空间、覆盖能力和交付可靠性关系。

这里存在一个关键的**可观测性前提**：如果任务中仍有当前系统尚未处理的重要条件，持续有效的后续审计应仍有非忽略的机会将其暴露为新增有效问题。该前提不要求审计发现全部剩余错误，只要求审计尚未退化为无法继续暴露重要问题的重复过程。

在任务边界和审计条件具有可比性、审计轮次有效且满足上述前提时，如果持续修复后新增重要问题总体减少并趋于稳定，可以把这种收敛视为条件性的正面过程证据，支持：

当前系统的覆盖能力已经足以稳定处理这一任务。

这一解释仍受具体模型、工具、上下文、审计方向和验证条件约束。收敛不能证明结果绝对正确，也不能排除多个轮次共享同一盲区；如果后续审计已经失去发现剩余重要问题的能力，表面收敛就不能稳定解释为覆盖充分。

不收敛的证据性质不同。若工程对象持续得到修复，而新的重要问题仍不断出现，说明系统仍在暴露此前未处理的重要条件。在问题发现真实有效、任务边界稳定且修复过程正常时，持续不收敛能够较强地支持：

当前系统还不能稳定处理这一任务。

因此，两类证据并不完全对称：**持续不收敛**可以较强地否定“当前系统已经稳定覆盖该任务”；**收敛**则只有在审计仍保持有效发现能力时，才能作为覆盖充分的正面过程证据。

MRAC 负责观测问题轨迹；覆盖状态则是依据 SCR 对轨迹作出的进一步解释。两者应当区分。

3.5 二阶观测与覆盖解释成立的条件

MRAC 的二阶观测要具有稳定解释意义，需要满足以下基本条件。

第一，任务边界应保持相对稳定。

如果目标和范围在多轮过程中不断变化，各轮问题可能来自任务本身的变化，而不能直接解释为同一任务上的收敛或不收敛。

第二，每轮发现的问题应在下一轮前得到必要处理。

如果工程对象没有随审计结果相应变化，多轮结果主要反映对同一对象的重复检查，而不是持续修复后的变化轨迹。

第三，核心审计协议应保持可比。

审计范围、问题标准和基本验证条件应保持稳定，或按预先规定的方式变化。若有意改变模型、提示、上下文加载方式或审计视角，应记录这些变化，使不同轮次仍可解释地比较。

“可比”并不要求所有轮次使用完全相同的模型或提示。预先设计的多视角、交叉模型或审计策略变化同样可以形成有效轨迹，前提是变化被明确记录并可解释。

第四，审计轮次本身必须有效。

工具失败、上下文缺失、异常输出或未按预定范围完成的审计轮次，不能作为正常的问题发现结果解释。

第五，后续审计仍应具有发现剩余重要问题的能力。

这是由问题轨迹解释覆盖状态的关键可观测性条件。若多轮审计反复采用相同路径并共享同一盲区，即使轨迹表面稳定，其正向解释也会明显减弱。

因此，MRAC 不把“轨迹趋稳”机械等同于“任务正确”，而要求对任务边界、审计协议、修复过程和持续发现能力同时加以约束与记录。

3.6 MRAC 判定规则的操作化

MRAC 的理论定义不要求所有任务采用统一阈值。不同任务类型、错误严重度和审计方式可以设置不同的工程判定规则。

一个具体实现至少应预先明确：

- 何种问题计入有效问题；
- 问题严重度如何划分；
- 使用多少轮次或多长观察窗口判断趋势；
- 允许的新增问题阈值；
- 需要连续稳定多少轮；
- 哪些验证条件必须同时满足。

这些参数可以因任务而异，但应在观测前确定，或按预先规定的方式变化，并在轮次之间保持可解释的比较基础。

因此，MRAC 是一个参数化的二阶观测框架，而不是对所有任务统一适用的固定停止数字。

4. 不收敛后的任务调整

4.1 从不收敛到搜索空间治理

如果 MRAC 持续表现为不收敛，依据第 3 章的解释，当前系统尚不能稳定处理该任务。

根据 SCR，此时有两个基本方向：

1. **增强覆盖能力**：更换或增强模型、工具、上下文、验证方式等；
2. **治理搜索空间**：改变任务边界，使任务进入当前覆盖能力更容易稳定处理的范围。

本文重点讨论第二种方式中的任务拆分。

较大的工程任务可能同时包含多组需求解释、工程状态、依赖关系、实现选择和验证路径，使当前系统难以稳定处理。如果原边界持续不收敛，继续追加局部修补未必能改变这种失配，还可能引入新的约束和问题。此时可以停止原任务，返回上一级目标重新拆分。

拆分不是机械地把工作切成更小的文本或代码块，而是尽量减少单个子任务需要同时处理的状态、依赖、约束、决策和影响范围。

每个子任务随后在新的边界下重新执行并运行 MRAC。收敛则继续推进；仍不收敛时，可以继续拆分，也可以转向增强覆盖能力。多个子任务完成后，其组合结果仍需接受必要的审计与验证。

这一过程可以概括为：

持续不收敛 → 调整任务边界 → 在新边界下重新执行与运行 MRAC → 根据新的轨迹继续判断。

当任务可以继续合理拆分时，这一过程能够递归发生。

MRAC 不要求审计对象采用固定形式。它可以作用于任务方案、需求说明、设计、代码变更或其他能够保持边界相对稳定并支持持续修复的工程对象。本文案例中的 Spec 和 PR 只是两种实践形式。

4.2 如何判断拆分是否有效

拆分本身不保证任务更容易处理。错误的边界可能增加跨任务依赖、重复判断、组合冲突、状态同步成本和验证负担。

因此，拆分是否有效不能只看子任务数量或表面规模，还应观察新边界下的 MRAC 轨迹是否改善。如果原任务持续不收敛，而拆分后的子任务分别出现收敛，这为新的任务边界改善覆盖能力与搜索空间匹配提供正向过程证据。

如果拆分后的任务仍持续不收敛，应继续调整拆分方式、进一步拆分，或转向增强模型、工具和验证能力。如果继续拆分不能减少需要同时处理的状态和依赖，或者任务已经无法合理拆分，则应调整能力或目标、交由人工处理或终止任务，而不是无限递归。

需要区分的是：拆分后 MRAC 轨迹改善，直接说明新的任务边界更容易被当前系统稳定处理；是否进一步提高最终交付可靠性，仍需独立结果指标验证。

5. 初步工程证据

5.1 数据性质、总体样本与判定方式

以下记录来自作者实际软件项目中的 MRAC 实践。当前保存的数据包括：

- 各任务使用的完整审计提示；
- 各有效审计轮次的审计结果；
- 对应的问题修复记录；
- 父任务与拆分子任务之间的元数据；
- 各任务的最终状态和关闭原因。

这些记录不仅包含最终任务状态，也保留了从审计、不收敛判断、任务拆分到子任务重新审计和关闭的过程。

截至本文数据截点，MRAC 观察记录覆盖 **35 个任务或拆分后的子任务**，其中 **33 个**已经达到项目采用的收敛条件、完成验证并合入主线，**2 个**仍在进行中。进行中的任务同样计入观察分母，以避免只报告已经完成的成功样本。

当前项目采用：

连续两次有效的全范围审计无新增实质问题

作为实用停止条件。该规则只用于本文工程案例的状态判定，不属于 MRAC 的理论定义。它也不具有预设的统计置信度：即使任务仍有未发现问题，重复审计也可能偶然出现连续无新增。因此，其实实际可靠性取决于审计协议持续发现重要问题的能力；不同模型、任务和审计配置所需的稳定轮次仍需实验校准。

这些数据来自同一内部软件工程环境，尚未公开完整仓库和全部审计轨迹，不能视为外部独立复现证据。本文只报告支持基本机制所需的代表性轨迹和聚合结果；不同层级命题的理论边界、证据要求及研究路线已在 MRAC L1-L5 研究纲领中进一步展开[2]。

5.2 一个持续不收敛轨迹示例

任务：n306-14 — Gate 原生单位闭环与 Conversion 假轴退休

该任务在原任务边界下连续完成了 20 轮有效审计。为便于展示，本文按有效审计顺序重新编号为 R1-R20；原始审计记录中的 Round 2 对应本文 R1。每轮发现的问题均在下一轮审计开始前进行相应修复。

轮次	P1	P2	轮次	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

20 轮共记录 70 次 P1 和 25 次 P2 问题发现。这里的计数表示各轮接受的问题发现次数，不主张不同轮次中的问题彼此统计独立，也不将其解释为 95 个互不相关的独立缺陷。

从跨轮轨迹看，局部轮次的问题数量虽然有所波动甚至下降，但 **20 个有效轮次均出现 P1，单轮 P1 数量始终为 2-6 个；即使到最后一轮，仍新增 4 个 P1。**

因此，尽管工程对象经过持续修复，新的高严重度问题始终没有停止出现，整个轨迹在 20 个有效轮次内没有表现出接近稳定的状态。按照本文定义，该任务在原边界下表现为明确的不收敛。

项目随后停止继续扩展原任务，并将其拆分为 4 个子任务：n306-32、n306-33、n306-34、n306-35。四个子任务此后均达到项目采用的 MRAC 收敛条件并完成验证。该结果在下一节与另外两个拆分闭环一并报告。

5.3 三个拆分闭环

截至当前记录，共有 3 个已完成并关闭的拆分聚合任务。三个父任务均在原边界下持续不收敛，共拆分形成 10 个子任务。

父任务	子任务数	子任务结果	父任务状态
n306-4	3	3/3 收敛并完成验证	Closed
n306-14	4	4/4 收敛并完成验证	Closed
n306-29	3	3/3 收敛并完成验证	Closed

父任务	子任务数	子任务结果	父任务状态
合计	10	10/10 收敛并完成验证	3/3 Closed

三个父任务的关闭原因均记录为：

Split child tasks completed and verified.

父子任务对应关系为：

- 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

所有相关子任务均已关闭，父子关系和拆分元数据完整保存；审计提示和逐轮记录也可用于回溯实际 MRAC 轨迹，而非仅根据任务管理系统中的 closed 状态推断收敛。

截至本文数据截点，尚未观察到原任务持续不收敛后重新拆分，而子任务仍持续无法收敛并导致闭环失败的案例。这一观察仅描述当前记录，不构成对未来任务或其他工程环境的普遍保证。

5.4 当前证据能够说明什么

当前记录已经包含多个完整拆分闭环，并以 35 个任务或拆分后子任务作为总体观察口径，其中仍有 2 个任务在进行中。

三个持续不收敛的父任务被重新拆分后，形成的 10 个子任务均达到项目采用的收敛条件并完成验证，父任务随后关闭。因此，当前记录支持一个有限但明确的观察：

“持续不收敛 → 重新拆分 → 在新边界下重新运行 MRAC → 子任务收敛”的过程已经在多个真实工程任务中重复出现。

这些记录直接支持的是：重新划定任务边界后，MRAC 轨迹和当前系统对任务的可处理性发生了改善。它们不能单独证明最终交付可靠性已经提高，后者仍需要合入后缺陷、返工、回退、生产运行等独立结果指标验证。

当前数据也不能证明：

- 任意工程任务都可以通过递归拆分最终达到收敛；
- 拆分是这些任务最终收敛的唯一原因；
- 结果可以直接推广到其他项目、模型或工程环境；
- 10 个子任务可以被视为 10 个统计独立样本；

- 当前样本已经足以确定最优拆分方式或递归深度；
- MRAC 收敛能够替代独立的真实交付结果评价。

本文也不以“最终收敛”反向定义拆分是否充分。拆分的目的仍是减少单个任务需要同时处理的状态、约束、依赖和影响范围；后续 MRAC 用于判断新边界是否实际改善了任务的可处理性。

6. 与已有工作的关系

6.1 迭代式自我反馈与修订

Self-Refine 让同一大模型依次承担生成、反馈和修订角色，并在多类任务上展示了迭代式自我反馈改善初始输出的可能性[5]。与此同时，Huang 等人在推理任务上的研究表明，在缺少外部反馈时，模型的内在自我纠正并不稳定，有时还会使结果变差[6]。两项研究共同说明，反复生成和修订本身并不自动等于可靠改进，反馈来源与验证条件会影响迭代结果。

MRAC 与这类方法共享“反馈—修订—再次检查”的迭代形态，但研究对象不同。Self-Refine 主要关注输出经过反馈后能否改善；本文关注每轮问题发现之间的关系，以及多轮问题轨迹能否作为覆盖判断的观测对象。

MRAC 因而不是一种新的单次输出优化方法。

6.2 Iterative Audit Convergence 研究

Calboreanu 对 AEGIS 多智能体编排系统由八份文档、约 7152 行内容组成的提示规范进行了九轮审计案例研究，共报告 51 个一致性缺陷；各轮新增缺陷数为 15、8、12、2、8、1、4、1 和 0，呈现非单调下降，最终在第九轮达到一次全范围审计无新增问题[4]。该研究的审计范围和协议在过程中有所扩展，因此作者明确将曲线解释为问题修复与审计方式完善共同作用下的观察结果，而不是固定审计函数的独立重复测量。

正式发表版本还在公开的合成规范上进行了跨模型部分复现和一致性检验，但仍属于单一生产系统案例，且没有设置单轮全范围审计对照[4]。该研究指出，干净轮次只能说明既定协议范围内的内部一致性，不能证明真实正确性，也没有测量修复是否改善系统性能、可靠性或运行结果。论文建议后续复现实验采用更严格的“连续两次全范围审计无新增问题”规则。

该研究为多轮审计可能呈现非单调收敛提供了公开案例。本文进一步依据 SCR 解释为什么执行、审计和修复能够改变后续需要处理的问题，并把多轮轨迹本身作为覆盖判断的二阶观测对象。

6.3 规范修订与传统缺陷轨迹研究

开源项目 *Automated Plan Reviser Pro* (APR) 对复杂 Spec 进行多轮 AI 修订并保存轮次历史。其收敛分析综合输出规模趋势、修改速度和相邻轮次内容相似度，形成对文档稳定程度的估计[7]。

APR 表明，多轮规范修订和稳定性分析已经进入公开工程实践。但输出变短、修改变慢和文本更相似主要反映文档变化趋缓，不能直接说明重要问题减少、关键约束得到覆盖或具体结果已经可以被可靠交付。

软件工程领域已有研究使用缺陷发现轨迹、可靠性增长模型和捕获—再捕获方法估计剩余缺陷，并据此讨论测试或审查的停止条件[8,9]。

MRAC 与这些工作共享“利用跨次问题记录获得额外信息”的基本思路，但关注的是 LLM 参与的**执行—独立审计—修复过程**：每轮审计结果形成一阶记录，多轮记录构成问题轨迹，轨迹被进一步用于解释当前 AI 系统是否已经能够稳定处理任务。

6.4 本文的位置

本文不主张首次发现“反复审计或修订可能趋于稳定”这一一般现象。本文主要推进三个方面：

1. **基础机制**：说明候选结果形成后，独立审计和修复为什么能够逐步改变后续需要处理的问题；
2. **二阶观测**：区分单轮审计结果与多轮问题轨迹，并把跨轮变化作为 MRAC 的直接观测对象；
3. **覆盖解释与任务调整**：依据 SCR，在明确可观测性前提下，将问题轨迹解释为覆盖能力与搜索空间匹配的过程证据，并用持续不收敛触发能力增强或搜索空间治理。

因此，MRAC 的关键不是“多审计几轮”或“问题数量最终下降”本身，而是：

在持续修复的 AI 工程过程中，将跨轮问题轨迹作为新的观测对象，并在审计仍保持有效发现能力时，用其辅助判断不可直接观测的覆盖关系。

本文的工程记录用于说明这一机制已经在多个实际任务中出现，而不试图在此完成更远范围的外推证明。

7. 局限性与研究议程

7.1 独立审计的有效性 with 共同盲区

即使审计在操作上与执行分离，同一模型、相似提示和相近上下文下的多轮审计仍可能采用相同路径并共享盲区；使用不同模型也不能预设其错误模式真正独立。

因此，问题轨迹趋稳既可能意味着重要问题已经很少，也可能只是当前审计方式无法继续发现问题。后续研究需要比较同模型独立 Session、不同审计视角、多模型交叉审计、工具验证和人工抽检，明确不同配置能够覆盖的错误类型，并检验提高审计分离程度是否带来更多有效发现。

7.2 候选结果并不必然显著降低审计难度

执行产生候选结果后，审计通常能够围绕具体对象工作，但并非所有任务都会因此明显变得容易。

如果候选结果严重偏离目标、任务要求本身不明确、审计仍需要重新理解几乎全部工程状态，或者关键失败只能通过开放探索发现，审计需要处理的问题仍可能很大。

本文目前给出的是机制解释，尚未直接量化候选结果对后续审计难度的影响。

7.3 协议变化、修复扰动与目标漂移

如果审计提示、上下文范围、问题标准或关注重点发生未记录的变化，多轮问题轨迹可能同时受到审计协议变化影响。

新增问题既可能是此前遗漏，也可能由上一轮修复新引入。反复修复还可能使工程对象逐渐偏离原始目标。

因此，后续实验应尽量保持审计协议可比，并保留原始任务描述、初始工程对象、各轮审计结果、协议变化和修复差异，使不同来源的变化能够分别分析。

7.4 当前证据强度

第 5 章的记录已经包含多个完整拆分闭环和未完成任务，但所有数据仍来自同一软件项目与工程流程，样本有限，也缺少外部独立复现。

因此，当前证据足以说明相关机制已在真实工程中重复出现并可执行，却不足以支持对递归拆分普遍性的强外推。现有数据主要反映 MRAC 轨迹和流程状态，也尚不能证明 MRAC 收敛或任务拆分必然带来真实交付可靠性的同比例提升。

本文作为 MRAC 基础概念论文，不继续扩大证据分析。不同层级命题所需的理论与证据强度、正反向证据以及更广泛的递归可收敛性问题，已在 MRAC L1-L5 研究纲领中进一步讨论[2]。

7.5 对照与评价

MRAC 的工程价值需要区分不同机制分别评价，而不能简单归结为“多轮审计是否优于单轮审计”。

后续至少可以设置以下几类对照：

1. **无独立审计**：用于观察执行结果在缺少审计反馈时的表现；
2. **独立审计与修复**：用于测量独立审计本身带来的工程价值；
3. **固定轮数审计—修复**：形成多轮问题轨迹，但不根据轨迹改变流程；
4. **MRAC**：利用相同类型的多轮轨迹，根据二阶观测决定继续修复、推进任务或调整任务边界。

其中，固定轮数审计—修复与 MRAC 的比较尤其重要。二者都具有多轮审计和修复，但只有后者把跨轮轨迹作为流程判断依据，因此可以更直接检验二阶观测本身是否增加工程价值。

同模型、多模型、多审计视角以及人工—AI 混合审计，可以作为不同审计配置进一步比较。

评价指标应包括合入后缺陷、错误严重度、返工量、回退率、交付时间、Token 与计算成本、修复新引入问题的比例，以及不同轨迹判断与最终交付结果之间的关系。

7.6 核心研究问题

后续研究可以集中于四个方向：

1. **独立审计效果**：候选结果形成后，独立审计在什么任务条件下能够显著降低后续处理难度；
2. **二阶观测的解释力**：不同问题轨迹在多大程度上支持覆盖能力与搜索空间匹配的判断，以及这一解释在何种审计配置下保持有效；
3. **任务拆分效果**：持续不收敛后重新拆分任务，是否能够稳定改善 MRAC 轨迹，并进一步改善独立测量的真实交付结果；
4. **工程效率**：如何在共同盲区、修复引入问题和审计成本之间取得平衡，提高 MRAC 相对于其他验证方式的净收益。

这些问题需要理论分析与持续积累的工程证据共同推进。本文只建立基本机制，并提供说明该机制已在多个实际任务中出现的初步证据。

8. 结论

根据 SCR（搜索空间—覆盖能力—交付可靠性关系），AI 任务的交付可靠性取决于覆盖能力与搜索空间的匹配。本文提出，连续的执行—独立审计—修复过程会形成具有版本连续性的问题轨迹，MRAC 对这一轨迹进行二阶观测。

MRAC 直接观测问题轨迹，而不是覆盖状态本身。重要问题总体减少并趋于稳定时，轨迹表现为收敛；持续修复后仍不断出现新的重要问题时，则表现为不收敛。在任务边界、审计协议和验证条件具有可比性，且后续审计仍保持有效发现能力的前提下，收敛可以作为覆盖能力足以稳定处理任务的正面过程证据；持续不收敛则构成更强的负面证据。

当任务持续不收敛时，系统可以增强覆盖能力，也可以通过拆分调整搜索空间，并在新边界下重新运行 MRAC。当前工程记录中，35 个任务或拆分后子任务已有 33 个达到收敛条件、完成验证并合入主线，2 个仍在进行；3 个原边界持续不收敛的父任务拆分后形成 10 个子任务，均达到收敛条件并完成验证。现有证据支持 MRAC 作为搜索空间覆盖判断与任务调整机制的可执行性，也为“不收敛—拆分—重新收敛”提供了初步工程证据，但不能证明任意任务都可递归收敛，更不能替代对真实交付可靠性的独立评价。

整体关系可以概括为：

**执行产生候选结果 → 独立审计发现问题 → 修复形成新版本 → 多轮结果形成问题轨迹 →
MRAC 对轨迹进行二阶观测 → 依据 SCR 解释覆盖状态 → 决定继续推进、增强能力或调整任务边界。**

本文建立的是 MRAC 的基本概念、可观测性前提与工程作用边界；不同层级命题的理论边界和证据强度，由 MRAC L1–L5 研究纲领进一步展开[2]。

参考文献

- [1] 刘明. 从 SCR 关系到搜索空间治理：AI 软件工程交付可靠性的基础机制[EB/OL]. Zenodo, 2026.
DOI: <https://doi.org/10.5281/zenodo.21902420>.
- [2] 刘明. MRAC 的理论起点与递进研究纲领：从不收敛诊断到工程 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=IkmD3fKBPQ>.
- [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>.