用于软件设计评审的大象-金鱼模型：全新审查者能够看到什么

Sabry E. Farrag

建筑、计算与工程学院，伦敦东部大学，伦敦，英国。通讯作者：sabryelgatem@gmail.com

arXiv:2611.32668v1 [cs.SE] 24 Nov 2026

摘要

一项设计可能很容易理解，却仍然规定了错误的系统。Google 的大象-金鱼模型要求新代理在不阅读产生该设计的对话的情况下审查设计。我们研究了这一选择如何影响对遗漏、协议违反和内部缺陷的检测。我们在 47 份公开软件设计文档中各植入九个缺陷，并为每份文档提供一段脚本化起草对话。将该对话作为其自身先前交流内容提供给全新审查者时，它检测出了 94.8% 的遗漏决策实例；而对同一模型仅提供文档时，检测率为 62.4%。对于仅对话要求的违反，它检测出了 98.3%。针对文档可见缺陷，提供对话时的召回率为 87.5%，不提供对话时为 88.4%；按照预先规定的 15 个百分点差异界限，两者等效。三名模型评审者对检测结果进行评分，并使用构造对照进行校准。在仅文档运行中，该插件的五阶段评审几何平均耗时为 9.03 分钟，而一名审查者为 1.81 分钟。在两种判定程序下，其估计召回率都较低，其中一种判定程序下的差异可与零区分开来。后续的通用提示比较得出了相似结果。该实验区分了自动化设计审查的两个目标：检验新读者能否重构设计，以及检查设计是否保留了支配实现的各项决策。

关键词 大象-金鱼模型、自动化设计审查、软件规格说明、编码代理、上下文隔离、需求检查

1 引言

一名开发者要求将待处理的聊天命令设置为 60 秒后过期，而设计规定的是五分钟。审查者可以仔细阅读设计，准确说明命令何时过期，并认可一份违反协议的设计。当设计要求进行事件日志记录，却遗漏了在测试期间禁用磁盘写入的决定时，则会出现另一个问题。即使错过设计会议的读者，也可能会问测试如何隔离文件系统。这两个例子都来自我们实验中使用的同一份预设文档。审查者需要了解什么，取决于审查旨在捕获哪种错误。

大象-金鱼模型（EGM）将这一问题转化为一项工程决策。在 Rensin 的论述中，大象与人类通过长时间的对话共同完成设计。金鱼在新的会话中阅读结果，并解释其含义（Rensin 2026年）。Google插件将这一理念转化为起草、检查和解决软件规格说明的工作流（Google 2026年；Suchy等人 2026年）。一名新读者会对文档进行严格检验：一个错过这段对话的人，能否理解需要构建什么？团队还需要知道文档是否忠实记录了已达成的协议。一项清晰但错误的超时设置能够通过第一项测试。

我们在文档内实验室实验中研究这些目标。我们在对话与规格说明之间的边界两侧设置缺陷。一处缺陷删除了一个已达成一致的决策，却保留了受影响组件的相关内容。另一处缺陷更改了一项需求，而其已达成一致的值只能在对话中找到。第三处缺陷使文档就其自身而言变得不一致、不正确或不完整。每位审查者都从全新会话开始。

将文字记录提供给其中一个会话，会在保持模型身份和会话新鲜度不变的情况下，改变其对协议的访问权限。提示词还将文字记录描述为审查者自己先前进行的交流。

我们提出四个问题。**研究问题1**。 向新审阅者提供起草对话后，对各缺陷类型的检测会发生怎样的变化？**研究问题2**。 与一名新审阅者相比，EGM 的汇总报告能检测出什么，以及需要多长时间？**研究问题3**。 召回率如何在两种 EGM 配置之间迁移，审阅者又能在历史修复中检测出什么？**研究问题4**。 可比工具向其审阅者提供了哪些证据来源？我们首先回答文档性问题，以区分已发布工作流所允许的内容与每个实验条件实际获得的内容。

2 大象-金鱼模型如何审查设计

2.1 从协议到批准

EGM 为开发者、起草会话和评审会话分配了不同的职责（表 1）。随着开发者确定权衡方案，大象可以询问约束存在的原因、比较替代方案并修订草稿。规格说明随后成为交给另一位读者的材料。金鱼开始工作时没有起草历史。它能否访问其他证据是一个单独的选择：在 0.1.1 版本中，编排器可以向审阅者传递阅读清单，但会在理解测试中将其隐藏。最终，当有人决定哪些发现项足以支持修改规格说明时，人类判断再次发挥作用（Google2026；Suchy等人 2026）。

五个审查阶段分别是初步高层次评审、理解、技术批评、第二次高层次评审和实施准备情况。理解读者首先对提案和现有系统进行释义。技术批评者可以检查另一种关系，例如规格说明与其所引用组件之间的协议。整合会在开发者处理这些问题之前，对观察结果进行合并和排序。我们的主要工作流成果是送达开发者的报告。

2.2 研究问题4。该审查者能知道什么?

有四种来源可区分评审配置：起草对话、拟议文档、所引用代码，以及独立的需求声明。额外证据会改变理解测试成功的含义。在进行符合性检查时，如果不提供约束性要求，正确答案可能无从得知。我们对八种工具在固定修订版本下已记录的权限进行了编码。其中七种描述了审查步骤。在这七种工具中，四种要求使用全新的上下文，一种使用起草讨论，另有两种则未说明

表1： EGM v0.1.1 中的审查活动、输入和输出。该实验对已完成的草稿使用了评审命令。

活动	可用证据	输出或决策
探索并起草	项目背景、需求以及 与开发者的讨论	已达成一致的概念简报和设计 草案
测试理解程度	仅阅读目标文档的 全新智能体	对提案的释义以及 现有系统，并找出差距
检查设计	草稿、阶段说明以及任何 提供的阅读清单	关于设计选择、技术 细节和实施准备情况的发现项
整合并解决	阶段报告、设计和开发人员 优先事项	汇总报告，以及关于 发现项的决策，以及获批准或修订的 设计

对话访问权限。四种工具指示审查者查看代码，另有两种明确将原始需求列为输入。这些是已记录的权限；运行记录则证实了我们所执行配置中的实际输入。

EGM允许使用所引用代码。跨上下文评审的实现仅接受工件本身。cc-sdd 设计评审器可以使用讨论历史、需求和代码。超时冲突在每种情况下具有不同的可观测概率。在线资源 1中的证据图列出了全部八种工具及其修订版本（Google 2026年；Song2026a；gotalab 2026年；BMad Code 2026年；Vincent 2026年；GitHub 2026年；Fission AI 2026年；Builder Methods 2026年）。

2.3 与检查和自动化评审的关系

在一项受控实验中，场景引导检查员比临时检查员或检查表审阅者发现了更多需求缺陷（Porter 等 1995）。让检查员从不同视角进行阅读并汇总结果的团队扩大了覆盖率（Basili 等 1996）。汇总会议可能会舍弃发现项，而汇总独立审阅者的结果则可能增加更多误报（Porter 等 1995；Porter 和 Johnson 1997）。EGM 将这两个问题引入自动化评审：角色画像分配注意力，而整合决定哪些内容得以保留。我们将交付报告与单人评审进行比较，并单独检查三次独立评审的汇总结果。

即使决策已记录，架构师头脑中的设计知识仍可能留存（Jansen 等，2008）。代理辅助开发为这些知识提供了另一个存放位置：起草对话。规范驱动开发的实践者描述了将审查工作委托给代理并有选择地阅读规格说明的做法（Agarwal 等，2026）。有关架构决策的研究评估了向语言模型提供设计上下文的效果（Dhar 等，2024）；文档一致性实验表明，即使是表格的呈现形式也会改变检测结果（Fukuda 等，2025）。Gärtner 和 Göhlich（2024）将需求形式化，以检测陈述之间的矛盾。我们的超时示例检验的是另一种关系：规格说明本身一致，却与记录在其他地方的决策相冲突。在问题规范研究中，Wang 等（2026）在固定生成模型和提示词的同时改变了检索机制。我们的转录对比固定模型和审查请求，同时改变对协议的访问权限及其作者身份框架。

跨上下文评审在代码、教程和演示文稿脚本上评估了全新会话。其重新分析未能解决与一次同会话评审之间的差异（Song 2026b，c）。模型可能偏袒自己的输出（Panickssery 等，2024），而长上下文可能使中间部分的证据更难检索（Liu 等，2024）。这两项发现都说明了冷读的必要性。Jin 和 Chen（2026）指出，审查者在依据需求评判代码时可能会拒绝正确的实现，并使用可执行检查来探究这些判断。我们考察的是更早阶段的工程决策：在实现尚不存在时，规格说明应要求何种行为。我们于2026年11月21日再次进行文献检索，在检索的数据库中发现了有关 EGM 的实践者报告，但未发现将该插件与一位全新审查者进行对比的已收录受控比较。在线资源 1 记录了检索边界。

3 研究设计

我们针对实施前文档开展了配对实验室实验（Stol 和 Fitzgerald 2018年；Wohlin 等 2012年）。每个预设缺陷的设计都经过四种主要评审条件的评审。固定文档后，我们可以比较每种条件下哪些个别缺陷最终进入报告。

3.1 文档与缺陷

我们检索了 EGM 的设计目录、其他公共设计目录，以及用于 EGM 输出的文件名。在 469 个候选项中，有 154 个符合纳入规则（其中 15 个来自 EGM 的代码库

，139 个来自其他代码仓库）。我们剔除了 7 份试点文档，以及历史集合中已有记录的 11 份文档。48 份样本特意纳入了剩余的 11 份 EGM 文档，以及来自其他代码仓库的 37 份文档。其中一份抽取的文档在尝试 12 次后仍未能生成种子列表。经分析的 47 份文档来自 31 个代码仓库。检索采用至少 100 行的最低要求，以及基于字符的语言过滤器；其中一份主要使用葡萄牙语的文档通过了筛选。各文档的权重相同。按照框架规模对两个分层进行加权后，每项观察到的对比变化最多为 1.6 个百分点。

一个播种代理为每份文档引入了三个遗漏决策、三个与仅对话需求相关的冲突，以及三个文档可见缺陷。最后一类包括矛盾陈述、缺失的失败案例和错误事实。在聊天命令设计中，超时值需要开发者同意后才能进行检查。关于发出十个事件的说法，可以与列表中的十二个条目进行核对。对于仅对话冲突，我们拒绝了那些被仅文档探针标记的种子，其中包括使用单人审查流程的三个探针。因此，在没有对话的情况下检测率较低，属于操纵检验。遗漏决策没有对应的可见性筛选。

一个单独的写作者会话生成了每段起草对话的双方内容。每个被删除的决策和约束性要求都出现了一次。同样形式的另外七到八条一致意见涉及文档中已陈述或已满足的细节。主实验对话记录包含1,802–2,429个单词。一个长对话记录组在9,912–11,021个单词中保留了15份文档的目标。一个历史组使用了11份修复前文档，其中两份来自EGM；每份文档都有一个由后续修正提交锚定的缺陷。该组的对话记录仅取材于修复前文档。该组用于测试在不引入额外对话证据的情况下对自然缺陷的检测。

3.2 评审条件

表 2 定义了各项配置。C1 和 C2 使用相同的 Opus模型，以及提出所要查找的缺陷类型的相同请求。C1 还收到了文字记录，并将其设定为自身较早的一次交流。C3 在 Claude Code 中运行了已发布插件，使用原生角色子代理。C4 则通过 OpenCode 的通用子代理路径运行相同的技能和角色文件，并配置了 MiMo。两者都使用不间断预设、空的阅读清单，以及不包含仓库代码的目标文档。在 C3 与 C4 之间，提供方、测试框架和调度同时发生变化。

每个阶段内的任务均以随机顺序启动。C1 和 C2 各生成了 141 份报告，C3 和 C4 则各生成了 47 份。11 份历史文档在每个主实验条件下各接受了一次审查。在主实验结果之后，我们从 C2 的请求中移除了缺陷列表，并在每份主实验文档上运行了三次该通用版本。我们在这些运行之前固定了其措辞和分析方式。它使用的是约在主实验比较一周后发布的较新版本 Claude Code。

3.3 测量与推断

我们的结果指标是审查是否报告了某一特定缺陷。来自三个系列（Opus、GLM 和 Qwen）的评审者收到了中性的发现标识符，以及随机排序的描述

表 2： 输入与复现。除非另有说明，计数均指 47 份种子文档。每次会话均从头开始。

ID	审查者及提供的材料	每份文档的运行次数	时间安排
C1	单人 Opus 审查者、文字记录和文档	3	Main
C2	单人 Opus 审查者，文档	3	Main
C3	EGM、Claude Code 和 Opus，文档	1	Main
C4	EGM、OpenCode 和 MiMo，文档	1	Main
C1L	与 C1 相同，但文字记录较长（15 份文档）	3	探索性
C2G	与 C2 相同，但使用通用型审查请求	3	事后分析

其中包括真实缺陷和虚假缺陷。他们筛查这些发现项，并在新的调用中重新考虑其提出的匹配结果。两名评审者一致筛查或确认，即计为检测到该缺陷。第四个系列撰写了阳性对照，这些对照单独植入，或嵌入其他发现项中。每份文档包含六个虚假缺陷，作为阴性对照。评审者既看不到审查条件，也看不到对照标签。作者简要审查了模型评审者的评估结果。报告的协议统计数据描述的是三名模型评审者。生成式模型协助准备材料、生成和评审审查结果，以及起草和重构本稿。在线资源 1介绍了所用工具，并提供了审查提示词。

试点字符串匹配产生了依赖条件的错误。第一个语义判定程序（v3）在前12份主实验文档中，C3的伪缺陷率超过了5%的伪缺陷阈值。修订后的措辞（v4）明确了发现项必须与缺陷匹配到何种程度。我们在该阶段完成了开发，并在判定其余35份文档之前将其固定下来。在四个主要组别中，v4找回了3,384个植入发现项中的3,362个，并将2,256个伪缺陷中的26个计为发现项。在真实单元上的Fleiss’ κ 为0.954。对照项上的协议和识别能力不同于审查者原生发现的准确率；独立的人工标注可以直接检验这种关系（Wang等，2025；Verga等，2024）。

我们对每份文档的各次运行结果取检测均值，并在文档内比较不同条件。因此，C1中某一缺陷类型的423个机会项代表141个不同的种子，每个种子均被审查三次。对于观测召回率 p、植入灵敏度 s 和条件及类型内的伪缺陷率 f，调整后的召回率为 $(p-f)\,/\,(s-f)$（Rogan和Gladen，1978）。全类型对比对各类型赋予相同权重。我们采用包含2,000次重抽样样本的文档自助法，重新估计校准和对比。配对的 t 区间描述观测差异；以文档为聚类单位的逻辑回归广义估计方程提供了第二种实现方式（Liang和Zeger，1986）。调整假设对照错误率可以迁移到审查者自身的发现项。一个被计入的伪缺陷也可能反映了一项匹配正确的错误指控。

我们预先设定实际意义界值为15个召回百分点，并采用双单侧检验评估等效性（Schuirmann，1987）。只有当观测区间和调整后区间均在同一侧排除零时，才判定存在方向性效应。只有当两个等效区间均落在该界值范围内时，才判定等效。对于较小且估计精确的差异，这些结论可以同时成立。对于三个对话记录对比，我们采用98.33%的效应区间和96.67%的等效区间，以控制多重性。工作流和配置对比分别采用95%和90%的区间。设计冻结时间戳在本地记录。我们分别报告两种判定程序、留出阶段以及主实验结果得出后选择的分析所得结果。

4 结果

4.1 研究问题1. 访问对话可找回遗漏决策

表3 给出了观测到的检测计数。审查者将文字记录作为独立对话交流内容查看时，检测出94.8%的遗漏决策；相比之下，仅文档审阅者检测出62.4%。访问权限带来的调整后差异为32.6个百分点（98.33%区间为23.6至41.4）。在36份文档中，差异方向一致；10份持平，1份仅有利于文档。图1 展示了三种类型的文档间分布。同一模型在能够读取需求时，检测出423项包含仅对话要求的需求中的416项冲突。未提供该需求时检测出的1项冲突，证实基于探针的选择产生了预期的可见性边界。

对于两位读者都能直接检查的缺陷，C2 在 423 个机会项中比 C1 多发现了 4 个。调整后差异为 0.6 分，倾向于 C2。两个等效区间的并集范围为 -2.9 至 $+4.5$ 分，远在预先设定的界值之内。在 47 份文档中，有 27 份两种条件打平。因此，文字记录为一类缺陷补充了大量缺失证据，而对文档可见类缺陷造成的可测量代价很小。

表3： v4 版本中按缺陷类型检测到的种子-审查机会项。每个短转录文本或单一审查者行涵盖每份文档的三次运行；每个 EGM 行涵盖一次运行。括号内为观测百分比。

审查者	遗漏决策	一致性冲突	文档可见
文字记录（C1）	401/423（94.8）	416/423 (98.3)	370/423 (87.5)
单人 (C2)	264/423 (62.4)	1/423 (0.2)	374/423 (88.4)
EGM–Opus (C3)	76/141 (53.9)	2/141 (1.4)	112/141 (79.4)
EGM–MiMo (C4)	77/141 (54.6)	6/141（4.3）	117/141（83.0）
通用型（C2G）	266/423（62.9）	1/423（0.2）	357/423（84.4）

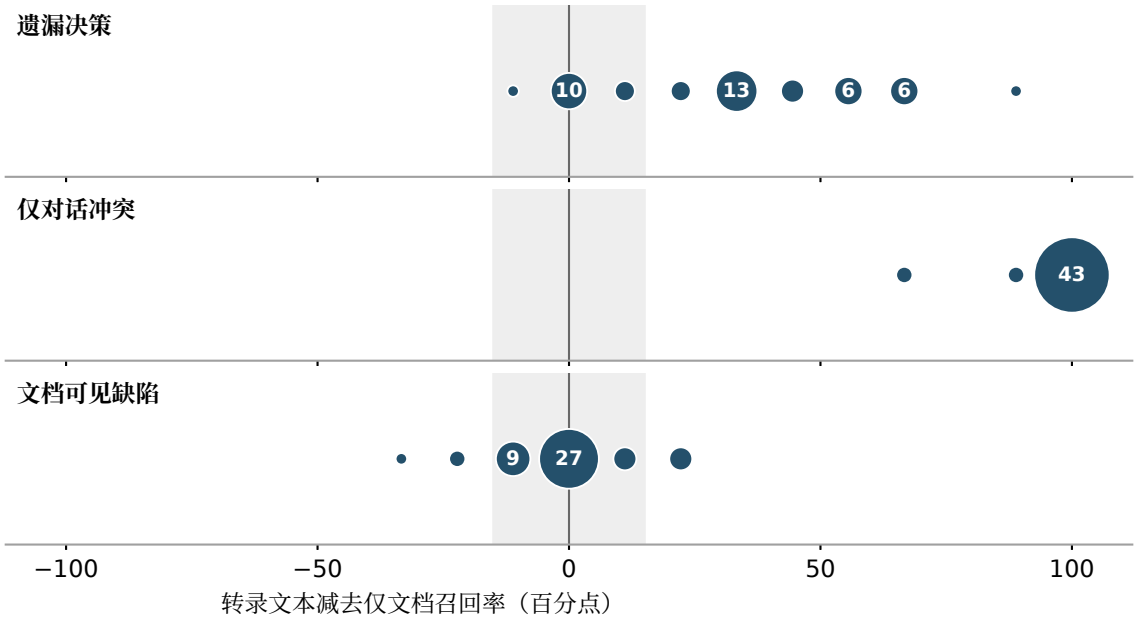

图1： 同一批47份文档的观测配对差异，三个面板均面向转录稿审阅者。圆圈面积表示具有相应差异的文档数量；标签标识包含六份或以上文档的组别。阴影标示均值对比的预先设定的15点界限。

我们考察了哪些遗漏促成了这一差异。在 141 个被移除的决策中，C2 每次运行都发现了 79 个，间歇性发现了 19 个，完全没有发现 43 个。对于最后一组机会项中的 129 个，C1 发现了 110 个。根据 C2 漏掉的项目进行划分，必然会使其在该组中的表现显得较差。因此，我们在描述性再分析中轮换留出运行：对于每项遗漏，如果 C2 的两次运行都漏掉了它，就将其选入；然后对第三次 C2 运行及相应的 C1 运行进行评分。在 140 个符合条件的留出机会项中，C2 找回了 11 个，C1 找回了 121 个。某项遗漏可能进入不止一折，因此这些只是描述性计数。在其余 283 个机会项中，计数分别为 253 和 280。EGM 也发现了一些 C2 连续三次都漏掉的遗漏（ C3 中 43 个里有 9 个，C4 中有 7 个）。有些遗漏决策会留下线索，可通过另一种阅读方式找回。在线资源 2 根据单项决策重现了这一划分。

当一致意见在文本中所占比例较小时，文字记录也很有帮助。分配到该组的15份文档中，遗漏项的长文字记录召回率为124/135，而仅对话冲突为133/135，文档可见缺陷为121/135。对于遗漏项，长文字记录召回率减去短文字记录召回率为 -0.044 （95%区间 -0.113 至 0.025）；长文字记录与仅文档相比为 $+0.244$。这一比较针对的是仅陈述一次、并在额外材料中得以保留的明确决策。

4.2 研究问题2：五个审查阶段与一次全新审查的对比

表4 展示了五项预定对比。在仅文档设置中，C3 检测出 190/423 个预设机会（44.9%）。C2 检测出 639/1,269 个（50.4%）。根据 v4，调整后差异为 -7.8 个百分点，观察到的差异为 -5.4 个百分点。两个区间均不包含零，且两个等效区间均落在 15 个百分点的界限内。较早的判定程序也显示 C3 低于 C2，不过其区间包含零。每种条件的分母中都有很大一部分是两者都很少检测到的仅对话冲突。将比较范围限定为遗漏决策和可见缺陷后，v4 下的调整后差异为 -11.2 个百分点（95%区间 -17.7 至 -5.3）。此时，界限内等效不再成立。全类型召回率回答的是关于这类特定缺陷组合的问题。

两项检验有助于解读工作流比较。对于从 v4措辞开发中留出的 35 份文档，C3 减去 C2 的结果为 -11.1 调整后分数（95%区间 -16.1 至-6.5）；而在最初的 12 份文档中，该结果为 $+1.9$（-5.3 至 $+8.6$）。各阶段之间测试框架的版本有所变更。在 v3 下，阶段对比仍然存在，因此不能将其完全归因于新的判定措辞。其次，后续的通用请求（C2G）删除了 C2 中明确列出的缺陷。与之相比，在 v4 下，C3 的调整后分数低 6.4 分（95%区间为低 2.3 至 10.8 分），在 v3 下低 1.5 分（区间为低 5.7 分至高 2.5 分）。通用单一审查者仍具竞争力。C2G 在较晚阶段运行，且未与 C3 同时重新运行。

机器时间体现了运营权衡的幅度。EGM–Opus 的几何平均耗时为 9.03 分钟，定向单一审查者为 1.81 分钟，比例为 5.0（95%区间为 4.66–5.32）。相较于通用型审查者，该比例为 6.1。汇总报告平均包含 21.3 个发现项，而 C2 为 23.0 个。其中，分别有 17.72 个和 18.46 个发现项与任何预设项都不匹配。这些发现项可能揭示预设集合之外的真实问题。其数量反映阅读负荷；其正确性和严重程度则需要单独评判。

三个独立 C2 运行的预先指定的并集，召回率达到 54.4%，高于 C3 的 44.9%。它们的算术平均机器时间为每份文档 5.79 分钟，而 C3 为 9.42 分钟。合并后，读者在去重前会接触到约 69 个发现项。反过来，将 EGM 的阶段报告与最终报告合并，使 C3 的观测召回率升至 52.0%。虚假归因也有所增加：最终报告中为 11/282，合并报告中则为 24/282。更长的清单为评审者匹配预设项创造了更多机会。表 5展示了各项缺陷出现于哪些输出中的探索性比较。EGM 报告比三次审查的并集多发现了 11 项，其中 9 项是遗漏决策。并集比 EGM 多发现了 51 项，其中包括 19 项文档可见缺陷。将 EGM 与每次 C2 运行分别比较，EGM 独有的发现项为 14–18 项，C2 独有的则为 36–45 项。这些比较统计的是获得归因的不同缺陷。单次运行比较中，每种方法对每份文档都提供一份交付报告，机器时间如上所述。

表4： 召回率的配对差异，单位为百分点。前 3 行括号内为 98.33% 的效应区间，后两行为 95% 的效应区间。调整估计值分别对每种条件和类型进行校准。v3 列为敏感性分析。

对比	观察值（v4）	调整后（v4）	已调整（v3）
遗漏，C2−C1	-32.4	-32.6 $[-41.4, -23.6]$	-31.2
冲突，C1−C2	$+98.1$	$+98.8$ $[+96.2, +100.7]$	$+98.0$
可见，C2−C1	$+0.9$	$+0.6$ $[-3.3, +4.7]$	$+1.8$
全部，C3−C2	-5.4	-7.8 $[-12.5, -3.3]$	-3.9
全部，C4−C3	$+2.4$	$+3.8$ $[-0.6, +8.4]$	$+0.8$

估计量和自助法不受约束；冲突区间在按物理上限截断前即被打印。

表 5： 每份文档中，三次定向单独评审的并集（C2）与一份合并后的 EGM 报告（C3）之间的重叠情况。计数采用主要判定流程（v4）。每个注入缺陷仅出现在一个单元格中。

缺陷类型	Both	仅 C2	仅 EGM	两者都不是
遗漏决策	67	31	9	34
仅对话冲突	0	1	2	138
文档可见缺陷	112	19	0	10
全部 423 个缺陷	179	51	11	182

4.3 哪些内容传达给了开发者？追踪已发布工具中的一条轨迹

在每种 EGM 配置下，47 份已分析的文档各自产生了一份理解报告和一份汇总报告：共 94 组成对输出。我们检查了其中一组，以说明这些文件如何回答召回率分数无法回答的问题。我们按已分析标识符的字典序，选取第一份 EGM 所有的文档及其原生插件运行；这是一种可复现的事后选择。设计方案提出支持三种编码智能体宿主。理解审查器正确找出了这三种宿主，以及分别测试安装、工作流和隔离的计划。随后，它区分了关于现有宿主的推断与其能够证实的事实，并询问支持表位于何处。最终报告将这一观察与若干术语问题归在一起。行动 A19 要求开发者指出包含支持表的文件。经过整合后，这项观察仍以可供他人采取行动的形式保留下来。

种子指标关注报告是否指出了某个已知缺陷。释义检验提案能否被还原；行动清单则显示工作流会将哪些不确定性延续下去。这是同一次审查的三种不同记录。这条轨迹展现了流水线中一条实际观察到的路径。在线资源 1 提供了锚点和选择规则；存档则为两种配置中的阶段文件和最终文件编制了索引。

4.4 RQ3. 配置迁移与历史修复

在 OpenCode 中，使用 MiMo 的 EGM 检测出 423 个机会项中的 200 个（47.3%）。相较于 EGM–Opus，其调整后优势为 3.8 个百分点，联合等效区间为 -1.1 至$+7.6$ 个百分点。该结果涉及两种配置下的召回率。两者的运行成本差异显著。OpenCode 运行每次审查的几何平均耗时为 59.65 分钟（为 Opus 配置的 6.6 倍），且平均多返回 9.1 个未匹配发现（95%区间为 6.81–11.36）。该比较无法区分模型、测试框架和子代理调度各自的贡献。

在 11 个历史修正中，C1、C2、C3 和 C4 分别标记了 3、4、3 和 5 个。修正提交识别出一个真实问题，但未标注其他合理发现。由于每种条件仅进行一次审查，且区间存在重叠，这些计数描述的是个别案例的覆盖率。预设项集合与历史集合回答的是有关工具的不同问题。

5 自动化设计审查的启示

5.1 从问题中选择输入

不参考起草历史的首次阅读与一致性检查，服务于不同的工程目的。团队若要决定是否将一份草稿交给新的实现人员，可以请一位审查者在首次阅读时重构草稿内容，并找出其中需要自行推测的部分。团队若要决定草稿是否保留了经协商确定的超时设置，则需要访问超时决策记录。将这两项检查混为一谈，可能会让一份流畅的摘要看起来像是符合性证据。在我们测试的框架下，获得完整脚本化对话的审查者很好地完成了第二项任务。A

更简短的决策记录可以降低阅读成本，使协议内容更易获取。其保真度，尤其是在需求变更之后，仍是一个需要通过实证回答的问题。

这种区分也为工具构建者提供了一个实用的接口选择。可以根据验证某项发现所需的材料为其标注：草稿中的两段内容、对话中的一项决策，或一项规范声明及其引用文件。这样，开发者在修改设计之前就能选择要检查的证据。第三类对 EGM 尤其重要。该插件支持引用源，并会指派审查者检查其与代码的契合度。我们的纯文档测试为一项交叉实验提供了基线：在该实验中，EGM 和一名单人审查者分别在有和没有冻结的源代码快照的情况下审查相同的设计。那些只有通过代码才能判定真伪的缺陷，可为这项比较提供有区分力的评估终点。

5.2 在恰当的阶段评估交付成果

对于收到汇总报告的开发者来说，早期发现的缺陷只有在可采取行动的信息进入报告后才有价值。对于想确认规格说明是否自足的开发者来说，即使没有种子匹配，释义本身也可能很有价值。基准测试可以通过存储释义、每项发现的证据及其处置结果，保留这两种结果。现有的文件记录链使这成为可能。下一项切实可行的衡量指标是工程师是否接受该发现，以及随后的修改是否纠正了设计；仅凭报告长度无法回答这两个问题。

6 有效性威胁

抽样与实验条件。 将每份文档与其自身进行比较，可以控制文档难度差异。对话撰写者明确说明了每项协议；而在真实会话中，决策可能会被推翻，或隐含在工具输出中。转录稿审阅者并非该设计的作者身份，并且收到提示，需将这段交流视为独立整体。与真正由作者持续进行的会话相比较，可以衡量作者身份和会话历史带来的额外影响。11份文档来自 EGM 自己的代码库。事后进行的代码库自助法得出的观测 95%区间为：遗漏方面 C2 减 C1 的结果介于 -39.4 至 -23.7 个百分点之间；总体上 C3 减 C2 的结果介于 -10.3 至 -1.8 个百分点之间。

实际意义。 15 个百分点的容差代表一种设计选择。在安全关键或高成本决策中，一个缺陷可能比若干处说明性文字中的缺陷更重要。可见缺陷比较支持将边际收紧至约 4.5 个百分点，而完整的 C3–C2 比较则需要约 11.8 个百分点。团队可以结合自身情境中漏掉问题的代价来解读这些界限。

测量与迁移。 同一模型族准备了材料，并提供了四位主要审阅者中的三位。两位外部评审者和构造对照在一定程度上解决了共同模型偏差问题。v4 和 v3 的结果限定了对工作流差异较小这一发现的解读：在 114 个流程存在分歧的真实或模拟单元中，两位额外的模型裁决者有 81 个意见不一。无论采用哪种流程，遗漏数量差异显著这一结论以及可见缺陷方面的等效性均成立。关于 EGM 的推论仅适用于无人值守评审中的一个版本，且不涉及代码库文件，也没有人工解决步骤。每份文档和每种配置仅运行一次 EGM，因此其独有检测结果的可重复性尚无定论。本实验测量了机器运行分钟数、检测结果和未匹配发现项。开发者投入的时间成本以及已实现系统的质量仍有待评估。

7 结论

在大象-金鱼模型中，启动新会话与决定该会话可以读取哪些内容，是彼此独立的工程选择。在 47 个设计中，将脚本化对话作为独立交流内容提供给新审阅者后，针对三项预设遗漏，每份文档大约多找出一项遗漏决策，而对文档可见缺陷的召回率

则基本保持不变。在仅文档设置下，EGM 的多阶段报告耗费的机器时间是一轮定向新评审的五倍，而且在两种判定程序下都没有显示出召回率优势。评估设计的团队可以通过冷读检验文档传达了什么，也可以通过对照已记录的决策进行审查，检验文档承诺了什么。这样，每项判定对于必须批准该设计的人来说，都有明确的含义。

补充信息

在线资源 1提供方案、评分详情、敏感性分析、读取权限图和演示轨迹。在线资源 2提供机器可读的检测决策、耗时、校准与对比摘要，以及用于复现计数、缺陷级重叠、留出遗漏分析和全部五项预定对比区间的脚本。

声明

数据可用性。 支持文中所报告对比的分析数据随本文作为在线资源 2提供。其中包括脚本化转录文本、单评审者报告、94组 EGM 理解报告与汇总报告、源文件修订、全部47个植入缺陷设计经哈希验证的转换、依据两种判定程序得出的检测标签，以及耗时。重放分析不依赖模型访问权限。全新审查运行需要相应的托管模型。在线资源 1列明了审查和评分说明。

资助和利益冲突。 作者声明未获得资助，也不存在利益冲突。

作者贡献。 SabryE. Farrag构思并设计了这项研究，准备了材料，开展了实验并分析了结果。

伦理。 本研究使用了公开的软件设计文档和模型生成的评审，未涉及人类参与者。

参考文献

Agarwal S、Singhal A、Macvean A、Vasilescu B、Breaux T（2026）《规范驱动开发：规范要求与实际实践》。arXiv 预印本 arXiv:2608.25202。https://doi.org/10.48550/arXiv.2608.25202Basili VR、Green S、Laitenberger O、Lanubile F、Shull F、Sørumgård LS、Zelkowitz MV（1996）《基于视角的阅读的实证研究》。Empir. Softw. Eng. 1（2）：133–164。https://doi.org/10.1007/BF00368702BMad Code（2026）《BMAD 方法》。https://github.com/bmad-code-org/BMAD-METHOD。软件代码库。提交版本 a867e1604b36（2026-11-08）；文档已存档；访问日期为 2026-11-19。Builder Methods（2026）《Agent OS》。https://github.com/buildermethods/agent-os。软件代码库。提交版本 e892e61（2026-10-07）；文档已存档于 2026-11-10。Dhar R、Vaidhyanathan K、Varma V（2024）《大语言模型能生成架构设计决策吗？——一项探索性实证研究》。载于：第21届 IEEE 国际软件架构会议，ICSA 2024，印度海得拉巴，2024年6月4日至8日，第79–89页。https://doi.org/10.1109/ICSA59870.2024.00016Fission AI（2026）《OpenSpec》。https://github.com/Fission-AI/OpenSpec。软件代码库。提交版本 579fcac57ae0（2026-11-09）；文档已存档；访问日期为 2026-11-19。Fukuda T、Nakagawa T、Miyazaki K、Tokumoto S（2025）《使用大语言模型开发自动化软件设计文档评审方法》。载于：IEEE 国际

软件分析、演化与再工程会议，SANER 2025，加拿大魁北克省蒙特利尔，2025年3月4日至7日，第91–101页。https://doi.org/10.1109/SANER64311.2025.00017Gärtner AE、Göhlich D（2024）《通过形式逻辑和大语言模型进行自动化需求矛盾检测》。Automated Software Engineering 31：49。https://doi.org/10.1007/s10515-024-00452-xGitHub（2026）《Spec Kit》。https://github.com/github/spec-kit。软件代码库。提交版本 08252af8c5fb（2026-11-10）；文档已存档；访问日期为 2026-11-19。Google（2026）《elephant-goldfish-model》（EGM）。https://github.com/google/elephant-goldfish-model/tree/dea7761de06ce5b51c177ebb9bba0469f24af5a4。软件代码库。版本 0.1.1，提交版本 dea7761；文档已存档于 2026-11-08；访问日期为 2026-11-19。gotalab（2026）《cc-sdd》。https://github.com/gotalab/cc-sdd。软件代码库。提交版本 e2a0c671aef3（2026-09-23）；文档已存档；访问日期为 2026-11-19。Jansen A、Bosch J、Avgeriou P（2008）《事后文档化：恢复架构设计决策》。J. Syst. Softw. 81（4）：536–557。https://doi.org/10.1016/J.JSS.2007.08.025Jin H、Chen H（2026）《大语言模型是可靠的代码审查员吗？需求符合性判断中的系统性过度纠正》。Automated Software Engineering 33：90。https://doi.org/10.1007/s10515-026-00638-5Liang KY、Zeger SL（1986）《使用广义线性模型进行纵向数据分析》。Biometrika 73（1）：13–22。https://doi.org/10.1093/biomet/73.1.13Liu NF、Lin K、Hewitt J、Paranjape A、Bevilacqua M、Petroni F、Liang P（2024）《迷失于中间：语言模型如何使用长上下文》。Trans. Assoc. Comput. Linguistics 12：157–173。https://doi.org/10.1162/TACL_A_00638Panickssery A、Bowman SR、Feng S（2024）《大语言模型评估器能够识别并偏好自身生成的内容》。载于：神经信息处理系统进展 37：2024年神经信息处理系统年度会议，NeurIPS 2024，加拿大不列颠哥伦比亚省温哥华，2024年12月10日至15日。http://papers.nips.cc/paper_files/paper/2024/hash/7f1f0218e45f5414c79c0679633e47bc-Abstract-Conference.htmlPorter AA、Johnson PM（1997）《软件评审会议评估：两项实验研究的比较分析结果》。IEEE Trans. Software Eng. 23（3）：129–145。https://doi.org/10.1109/32.585501Porter AA、Votta LG、Basili VR（1995）《软件需求检查的检测方法比较：一项复现实验》。IEEE Trans. Software Eng. 21（6）：563–575。https://doi.org/10.1109/32.391380Rensin D（2026）《大象、金鱼与软件工程的新黄金时代》。Medium，2026年4月27日，
h
t
t
p
s://
medium.com/@drensin/elephants-goldfish-and-the-new-golden-age-of-software-engineering-c33641a48874。文章；文本取自作者于 2026-11-19 公开发布的 RSS 源。Rogan WJ、Gladen B（1978）《根据筛查试验结果估计患病率》。American Journal of Epidemiology 107（1）：71–76。https://doi.org/10.1093/oxfordjournals.aje.a112510Schuirmann DJ（1987）《双单侧检验程序与功效方法在评估平均生物利用度等效性方面的比较》。Journal of Pharmacokinetics and Biopharmaceutics 15（6）：657–680。https://doi.org/10.1007/BF01068419Song TE（2026a）《CCR：跨上下文评审》。https://github.com/SongT-50/ccr。软件代码库，跨上下文评审的参考实现。提交版本 b7eb80a（2026-04- 03）；文档已存档于 2026-11-10。Song TE（2026b）《跨上下文评审：通过分离

生产与评审会议。arXiv预印本 arXiv:2603.12123。
https://doi.org/10.48550/ARXIV.2603.12123Song TE（2026c）何时第二个模型会有所帮助？大语言模型验证中的跨模型评审。arXiv预印本 arXiv:2610.01471。
https://doi.org/10.48550/arXiv.2610.01471Stol KJ、Fitzgerald B（2018年）软件工程研究ABC。ACM软件工程与方法学汇刊。27(3):11:1–11:51。https://doi.org/10.1145/3241743Suchy B、Quinn J、Ewing E、Newth O、Rensin D（2026年）介绍大象-金鱼模型插件：扩展设计驱动的人工智能软件开发。Google开源博客，10月22日。
https://opensource.googleblog.com/2026/10/introducing-elephant-goldfish-model-plugin.htmlVerga P、Hofstätter S、Althammer S、Su Y、Piktus A、Arkhangorodsky A、Xu M、White N、Lewis P（2024年）以评审团取代评审者：使用多样化模型组评估大语言模型生成结果。arXiv预印本 arXiv:2404.18796。https://doi.org/10.48550/ARXIV.2404.18796Vincent J （ 2026 年） Superpowers。https://github.com/obra/superpowers。软件代码库。提交 22f0b463e7a5（ 2026-11-03）；文档已存档；访问日期：2026-11-19。Wang R、Guo J、Gao C、Fan G、Chong CY、Xia X（2025年）大语言模型能取代人工评估者吗？软件工程中大语言模型作为评审者的实证研究。ACM软件工程学报论文集。2（国际软件测试与分析研讨会）：1955–1977。
https://doi.org/10.1145/3728963Wang Z、Li J、Guo X、Tsuchiya T（2026年）从噪声反馈到证据感知型问题规范：一种智能体治理的检索增强生成方法。自动化软件工程 33:103。
https://doi.org/10.1007/s10515-026-00640-xWohlin C、Runeson P、Höst M、Ohlsson MC、Regnell B、Wesslén A（2012年）软件工程实验研究。施普林格。
https://doi.org/10.1007/978-3-642-29044-2