ScaleQuality Code Maturity Model (SQCM)

Um modelo de maturidade de código baseado em evidências para a era do software gerado por IA

Versão 2.1 · Setembro de 2026

Erik Fernandes · ScaleQuality

Calibração de pontuação descrita: v2.2

DOI: 10.5281/zenodo.22781322

Tradução da edição canônica em inglês

Resumo

O ScaleQuality Code Maturity Model (SQCM) organiza evidências observáveis de software em cinco domínios: Segurança, Confiabilidade, Manutenibilidade, Durabilidade de Código de IA e Cadeia de Suprimentos. Seu propósito é apoiar decisões sobre uma base de código específica e seus artefatos medidos, com limites explícitos para o que foi observado. Não é uma certificação de segurança, um substituto para uma avaliação de maturidade organizacional ou um preditor de incidentes validado em uma população longitudinal.

A versão 2.1 consolida a metodologia e documenta a implementação auditada em setembro de 2026, incluindo a calibração de pontuação **v2.2**. A mudança central é o uso da cobertura medida de testes como entrada contínua, distinguindo-a da evidência estrutural de testes. A revisão explica a proveniência da cobertura, a execução parcial de testes, o escopo de repositórios, a ausência de evidência e as condições para comparar pontuações. Também corrige afirmações anteriores sobre monotonicidade da evidência e reprodutibilidade. Dezenove suítes internas de testes direcionados, com 194 testes aprovados, sustentam os contratos de implementação examinados aqui; não estabelecem validade preditiva nem precisão e revocação gerais dos scanners.

Nota de publicação. Este documento substitui o SQCM v2.0, DOI 10.5281/zenodo.21879222. O texto em inglês é canônico; este PDF em português é sua tradução. Versões do paper identificam publicações. O campo scoreVersion de uma execução identifica sua calibração de pontuação. As duas sequências são deliberadamente distintas.

1. Conceito, escopo e evidência

O SQCM define maturidade de código como uma avaliação versionada da condição e das práticas de engenharia de apoio observáveis no código-fonte, em seu histórico e nos artefatos de análise disponíveis. Sua unidade é um repositório medido ou um projeto composto por análises de repositórios. Os processos de uma organização e o comportamento em execução do serviço implantado são unidades de análise diferentes.

Trabalhos estabelecidos oferecem fundamentos complementares: modelos de qualidade de produto [1][2], modelos de dívida técnica e manutenibilidade [3][4][5], maturidade de processos organizacionais [6] e pesquisa de desempenho de entrega [7]. O SQCM combina sinais selecionados para uma avaliação operacional de código. Não afirma substituir esses referenciais nem demonstrar conformidade com seus requisitos pela simples produção de uma pontuação.

Nenhuma resposta de questionário entra no compositor de pontuação de código descrito aqui. Um fluxo separado do produto pode comparar a avaliação declarada por um time com evidências de código e ajustar seu resultado no nível do time. Esse fluxo, as médias de organizações e unidades de negócio, as métricas de entrega e as projeções financeiras não são domínios adicionais do SQCM de código e não são validados por este paper.

Cada evidência deve ser interpretada com sua origem e seu escopo. Um relatório gerado por testes instrumentados, um relatório enviado por um cliente de CI, um achado estático, uma heurística do histórico Git e uma inferência assistida por modelo são tipos diferentes de evidência. A pontuação é um apoio à decisão construído com essas entradas; ela não as transforma em observações equivalentes.

A descoberta automática é o padrão na avaliação de repositórios. Toolchains suportadas, dependências acessíveis, comandos de teste, caminhos de relatórios e exclusões ainda afetam o sucesso da medição. Existe configuração explícita de projeto para algumas dessas condições. Referências anteriores a configuração zero descrevem o caminho padrão de entrada, não a execução universal da infraestrutura privada de testes de qualquer repositório.

2. Os cinco domínios

Domínio	Evidência principal	Limite de interpretação
Segurança	Fraquezas estáticas, segredos detectados, práticas de segurança e configurações inadequadas de infraestrutura	Um achado não é uma avaliação completa de explorabilidade; ausência de achados não prova segurança.
Confiabilidade	Achados de confiabilidade, práticas de teste e cobertura medida quando disponível	Linhas executadas e práticas detectadas não estabelecem eficácia dos testes nem confiabilidade em produção.
Manutenibilidade	Achados de manutenibilidade, duplicação, práticas de engenharia e evidência de cobertura	Uma avaliação composta não é uma estimativa direta do custo futuro de manutenção.

Domínio	Evidência principal	Limite de interpretação
Durabilidade de Código de IA	Sobrevivência de código atribuído a IA em uma janela limitada do histórico Git	Atribuição e sobrevivência são aproximações; não identificam todo uso de IA nem estabelecem correção.
Cadeia de Suprimentos	Advisories conhecidos de dependências e de imagens de contêiner fornecidas	Mede-se a presença de advisories; alcançabilidade e explorabilidade em execução não são estabelecidas.

2.1 Segurança

Segurança combina achados e evidências de práticas sem tratar cada verificação como um defeito intercambiável. Segredos na árvore atual e segredos encontrados apenas no histórico recebem tratamentos diferentes. Uma exposição crítica pode limitar o resultado geral mesmo quando outros domínios são fortes. Achados de infraestrutura contribuem para Segurança, mas não acionam, isoladamente, esse limite de exposição crítica.

Essa distinção corrige uma simplificação anterior: um segredo atual detectado não coloca necessariamente o **próprio domínio Segurança** em sua faixa mais baixa. Em uma fixture controlada, sem outros problemas, um segredo crítico atual produziu Segurança 60 e limitou a pontuação geral a 55. A severidade do achado, a faixa do domínio e o limite geral descrevem coisas diferentes.

As verificações de infraestrutura inspecionam artefatos do repositório, como definições de contêiner e templates de orquestração. Algumas verificações de referência são moderadas quando expressam intenção arquitetural em vez de defeito. Isso é calibração do instrumento, não uma determinação de conformidade da implantação com uma norma de segurança.

2.2 Confiabilidade e Manutenibilidade

Esses domínios combinam achados estáticos e práticas de engenharia observadas. A cobertura fornece uma entrada para ambos; a seção 3 define seu significado. Encontrar um framework de testes ou de testes de mutação é evidência de presença, não prova de execução de uma suíte eficaz nem uma pontuação medida de mutação.

A duplicação é medida em formatos de código-fonte selecionados, com tolerância para repetição limitada. Artefatos gerados, catálogos de tradução e lockfiles poderiam, de outra forma, dominar o denominador. O registro de desenvolvimento da v2.0 relatou duplicação de 8,5% antes de restringir o escopo ao código-fonte e de 1,7% depois, no mesmo objeto. Esses valores ilustram um erro histórico de escopo, não um novo benchmark ou fator de correção universal.

Pesquisas sobre churn, autoria e concentração de conhecimento oferecem contexto relevante [10][11][12][13]. Elas não implicam que o SQCM calcule atualmente toda métrica discutida nesses estudos. Na composição auditada, são usadas convenções Git e evidência de durabilidade; uma medida de truck factor não é entrada da pontuação de código.

2.3 Durabilidade de Código de IA

O motor de durabilidade examina uma janela de 90 dias do histórico e permite 14 dias para a estabilização do código recém-introduzido antes de avaliar sua sobrevivência. Ele separa atividade atribuída a IA, atividade humana comum e automação conforme suas heurísticas. O domínio de IA usa o percentual de sobrevivência da coorte atribuída a IA. A sobrevivência humana é uma comparação apresentada junto ao resultado, não um denominador pelo qual a nota de IA é normalizada.

Histórico indisponível, ausência de commits recentes, coorte ainda aguardando estabilização e ausência de atribuição a IA têm estados distintos. Não devem ser interpretados como durabilidade de 0% ou 100%. A implementação também limita o processamento de blame, com limite padrão de 500 arquivos. Em repositórios grandes, isso pode deixar parte da coorte sem exame e enviesar a sobrevivência para baixo; o resultado não deve ser interpretado como linhagem exaustiva do repositório inteiro.

Sobrevivência do código não é correção. Refatorações necessárias podem reduzir a sobrevivência; código defeituoso preservado pode aumentá-la. A atribuição de autoria depende de metadados e padrões reconhecíveis e pode não detectar assistência sem marcação. Esse domínio não é um inventário de serviços externos de IA, uma medida de governança de modelos ou prova de que uma pessoa específica usou um assistente.

Estudos sobre segurança de assistentes [15][16], observações comerciais de churn e duplicação [17][18][19], pesquisas controladas de produtividade [20] e pesquisas de entrega [21][22] motivam a investigação, mas diferem em população, métodos e possíveis vieses. Relatos de adoção [23][24][25] oferecem apenas contexto histórico. Um preprint empírico recente examina problemas associados a commits com autoria de IA verificada [26]; outro estudo combina uma revisão de 109 artigos, workshops e experimentos com patches [27]. Nenhum valida os pesos do SQCM nem estabelece que sua pontuação de durabilidade prediz incidentes.

2.4 Cadeia de Suprimentos

A análise de dependências compara componentes com dados de advisories [32]. O domínio combina unidades de advisory deduplicadas e ponderadas por severidade por meio de uma função de decaimento limitada. Sua calibração evita a saturação precoce em que pequenos conjuntos de advisories levavam quase todos os objetos ao mínimo. Os coeficientes internos exatos não são publicados.

Quando um cliente de CI fornece uma análise da imagem de contêiner construída, sua evidência de advisories pode contribuir para o mesmo domínio. A receita de contêiner de um repositório não é uma medição da imagem implantada. Obrigações de licença são relatadas separadamente e **não alteram a pontuação numérica**; sua aceitabilidade depende do uso e da política.

SLSA e OpenSSF Scorecard oferecem referenciais relacionados de garantia [28][29], enquanto a ordem executiva dos EUA e o regulamento europeu citados fornecem contexto histórico e específico de jurisdição [30][31]. O SQCM não certifica conformidade com eles. Um pacote vulnerável conhecido pode ser inalcançável em execução; a presença de um advisory não resolve essa questão. Inversamente, a ausência de um caminho de chamada demonstrado não prova que o componente seja inofensivo.

3. Cobertura: o que foi medido, onde e como

3.1 Quatro caminhos de evidência

Caminho	O que acontece	O que o resultado pode sustentar
Diagnóstico de repositório conectado	Um executor suportado faz checkout de uma revisão fixada, executa testes e lê um relatório recente.	Cobertura do escopo instrumentado naquela revisão, sujeita ao estado de execução e às exclusões.
Importação de relatório de CI	O cliente empacota os relatórios disponíveis e os envia com a análise.	Cobertura relatada pelo ambiente do cliente; não é atestação independente desse ambiente.
Validação antes/depois dos agentes	Recibos pareados verificam revisão, atualidade do relatório, resultados dos testes, identidade do relatório e consistência de escopo.	Uma comparação de cobertura mais rigorosa para o par validado.
Alternativa estrutural	Mapeamento entre testes e módulos e evidências de práticas disponíveis são examinados quando não há cobertura de execução.	Evidência de estrutura ou postura de testes, não um percentual medido de linhas executadas.

Os leitores de relatórios suportam JaCoCo, LCOV, Cobertura, Istanbul, Clover e cobertura Go. Suportar um formato não garante executar toda linguagem, sistema de build ou estrutura de testes capaz de produzi-lo. Configuração, arquivos gerados, seleção de testes, credenciais e serviços externos podem afetar a execução e o denominador.

No diagnóstico conectado, a medição por execução é tentada antes do veredito final quando habilitada. O repositório selecionado e a revisão são verificados. Execuções com falha, indisponíveis ou incompatíveis recorrem à evidência estrutural, com um motivo. A conclusão de um diagnóstico não prova, isoladamente, que toda a suíte passou.

3.2 Execução completa e parcial de testes

O executor geral de cobertura pode aceitar um relatório recente produzido pelos testes que passaram mesmo quando parte da suíte falhou, com uma observação de execução parcial. Esse percentual descreve o escopo executado e deve ser lido com essa observação. Não é intercambiável com cobertura de uma suíte completa e bem-sucedida.

O caminho de validação pareada dos agentes aplica critérios mais fortes: os relatórios precisam ser recentes, os testes devem ter passado, falhas e execuções incompletas ou não suportadas não devem ser tratadas como medições válidas, e o escopo comparado precisa permanecer consistente. Se não for possível estabelecer um par comparável, a cobertura medida não é creditada como melhoria verificada entre antes e depois. Esses critérios mais rigorosos não devem ser generalizados para todo número de cobertura exibido na plataforma.

3.3 Entrada contínua na calibração v2.2

Para um relatório de cobertura de linhas, a proporção subjacente é:

cobertura = linhas instrumentadas cobertas / total de linhas instrumentadas.

O escopo e as exclusões do relatório determinam o denominador. O SQCM normaliza o percentual aceito para uma proporção entre 0 e 1. Na calibração v2.2, um relatório medido válido contribui continuamente para Confiabilidade e Manutenibilidade. Um limite qualitativo de postura de testes usado para evidência estrutural não substitui essa proporção medida.

O tratamento categórico anterior podia deixar de distinguir melhorias após a cobertura atingir o limite de uma categoria. O caminho atual preserva diferenças acima desse limite. A fixture controlada do compositor a seguir mantém as demais entradas fixas; são resultados sintéticos de regressão, não medições de repositórios de clientes.

Cobertura medida	Confiabilidade	Manutenibilidade	Geral
70%	84	92	93
80%	86	93	94
90,5%	89	94	95
100%	91	95	96

Esta não é uma tabela universal de conversão. Outros achados, práticas, disponibilidade de domínios, arredondamento e limites de exposição crítica afetam o resultado. Em uma fixture com limite aplicado, a cobertura maior melhorou seus domínios contribuintes enquanto a pontuação geral permaneceu em 55. Um percentual maior não garante aumento visível no resultado geral arredondado.

Cobertura também não é eficácia de testes [8]. Executar todas as linhas pode coexistir com asserções fracas ou casos comportamentais ausentes. Testes de mutação oferecem outra perspectiva [9], mas o SQCM deve distinguir um framework de mutação detectado de um experimento de mutação efetivamente executado.

3.4 Escopo com múltiplos repositórios

O caminho de cobertura por execução conectado examinado nesta revisão atua sobre o repositório principal analisado com sucesso. Ele não executa a suíte de cada repositório conectado simplesmente porque o projeto contém vários repositórios.

Na agregação de evidência arquitetural, a cobertura aceita de relatórios tem precedência sobre estimativas estruturais. As proporções do grupo selecionado são combinadas por média aritmética, e não ponderadas pelo total de linhas instrumentadas de cada repositório. Consequentemente, um percentual no nível do projeto não é necessariamente cobertura de linhas do projeto inteiro. Uma comparação defensável exige identificar os repositórios e relatórios incluídos; adicionar um repositório pode alterar tanto o escopo quanto o número.

4. Composição, níveis e ausência de evidência

As pontuações dos domínios e o resultado geral usam uma escala de 0 a 100. A composição geral combina os domínios disponíveis e aplica regras de calibração, incluindo limites de exposição crítica. Segurança, Confiabilidade e Manutenibilidade formam o núcleo necessário para um snapshot pronto no compositor auditado. Durabilidade de Código de IA e Cadeia de Suprimentos contribuem quando mensuráveis; sua ausência não impede que o snapshot do núcleo esteja pronto.

Domínios opcionais indisponíveis são excluídos da composição, e as contribuições restantes são normalizadas. Isso corrige a descrição anterior que exigia os quatro domínios originais antes da existência de um snapshot temporal. Um zero numérico é um resultado baixo medido; um valor indisponível não é um resultado numérico. Estado e proveniência devem acompanhar a interpretação. Nem toda projeção legada preserva essa distinção igualmente, portanto um escalar isolado é evidência insuficiente de uma medição concluída.

As cinco faixas gerais têm limites em 40, 60, 80 e 90. Esses limites e os pesos internos são calibração de projeto, influenciada por convenções estabelecidas de maturidade [6], não limiares estimados por um estudo longitudinal de resultados. As categorias de confiança indicam condições da evidência; não são probabilidades calibradas de que o software seja seguro ou de que uma previsão esteja correta.

A implementação separa a composição da pontuação da geração de narrativa. A avaliação qualitativa assistida por modelo pode fornecer evidência estrutural da postura de testes quando não há relatórios medidos. Portanto, não é correto descrever o fluxo completo como isento de inferência. A afirmação é mais restrita: a cobertura medida aceita não é substituída por um percentual inventado pelo modelo.

5. Limites de medição e comparabilidade

5.1 Adicionar evidência pode alterar o escopo disponível

A versão 2.0 afirmava que fornecer um relatório de imagem só poderia reduzir uma pontuação e que omitir evidência não poderia melhorá-la. Essa garantia geral é retirada.

Com um conjunto fixo de domínios medidos e as demais entradas fixas, adicionar evidência adversa de advisories não melhora o domínio Cadeia de Suprimentos. Porém, tornar mensurável um domínio antes indisponível também muda a composição geral. Uma fixture controlada, inicialmente sem Cadeia de Suprimentos disponível, passou de 95 para 96 no resultado geral quando evidência de uma imagem sem achados introduziu um domínio medido. Omitir evidência adversa também pode fazer a avaliação parecer melhor. A fórmula não estabelece resistência geral à omissão seletiva de evidências.

A comparação adequada é, portanto, **pontuação mais escopo mais proveniência**. Uma mudança na origem da cobertura, no conjunto de domínios medidos, nos repositórios, na profundidade dos scanners, nas exclusões ou na versão de cálculo pode explicar uma diferença sem alteração correspondente no código subjacente.

5.2 Reproduzir exige mais que o mesmo commit

A composição é determinística para entradas aceitas e calibração fixas. Coletar essas entradas novamente no mesmo commit é outra operação. Bases de advisories mudam, janelas do histórico Git se deslocam com o relógio, ferramentas e regras mudam, e avaliações qualitativas assistidas por modelo podem variar.

Um registro reprodutível deve identificar revisão, escopo de repositórios e artefatos, identidade do relatório e exclusões, resultado da execução, horário da análise, versões de ferramentas e regras, versão de cálculo e versões relevantes de dados externos. Esses são requisitos metodológicos para uma reprodução defensável, não uma afirmação de que toda exportação atual já captura cada item integralmente.

Resultados de diagnóstico concluídos têm proteções de persistência contra substituição tardia, e a versão de cálculo acompanha a execução. A lógica dedicada de relatórios de evolução suprime diferenças entre versões de cálculo. Entretanto, nem todo gráfico de resumo aplica atualmente as mesmas restrições de comparação. O leitor deve examinar os registros versionados antes de tratar toda tendência visual como melhoria entre condições equivalentes. Esta publicação não renomeia nem recalcula silenciosamente execuções históricas.

6. Profundidade da análise estática e escopo de artefatos

A análise rápida combina verificações estáticas selecionadas, detecção de segredos, análise de dependências, duplicação, verificações de infraestrutura quando aplicáveis e evidência de práticas. A análise profunda acrescenta um motor de grafos de propriedades do código [33] com especificação proprietária de taint. A seleção auditada escolhe a linguagem suportada dominante entre Java e JavaScript/TypeScript; não implica que todo módulo de um repositório poliglota receba o mesmo tratamento profundo.

Os catálogos contêm dez classes de vulnerabilidades Java e sete JavaScript/TypeScript. Quando a execução profunda está indisponível, a análise rápida ainda pode produzir achados. Um resultado rápido concluído não deve ser interpretado como execução bem-sucedida de cada verificação profunda. A granularidade da proveniência permanece uma limitação; ela não estabelece universalmente o sucesso de cada consulta.

Restrições de licenciamento influenciaram a escolha da infraestrutura de análise. Os termos históricos do CodeQL citados em [34] devem ser consultados em suas condições reais; disponibilidade para alguns usuários não implica direito irrestrito de redistribuir um serviço comercial de análise hospedada. Este paper não oferece uma opinião geral sobre licenciamento.

Uma SBOM CycloneDX pode acompanhar o resultado de um repositório. Sua disponibilidade depende da coleta bem-sucedida e dos limites de transporte; a presença de uma opção de exportação não prova que foi produzido um inventário completo. Desempenho em produção, comportamento da aplicação em operação e explorabilidade em execução permanecem fora deste diagnóstico de código. Executar testes em um ambiente controlado não amplia esse limite para telemetria de produção.

7. Validação e resultados

7.1 Auditoria de implementação de setembro de 2026

A revisão seguiu contratos da interface, chamadas de serviços, composição, persistência, fronteiras de integração, tratamento de falhas e testes de regressão direcionados. Definições de tarefas de produção e identificadores de imagens em execução foram inspecionados separadamente do código local. Isso confirma o snapshot de implantação auditado; não constitui novos testes completos de cada integração de cliente.

Grupo de validação	Suítes	Testes aprovados
Composição, cobertura, upload, estados e diferenças dos agentes	7	36
Recibos, verificações de revisão, descoberta/atualidade de relatórios e suítes ausentes	6	83
Persistência, projeção de CI, cobertura, cards e contratos MCP	5	71
Exemplos e contraexemplos controlados específicos do paper	1	4
Total	19	194

Os quatro testes específicos do paper exercitam cobertura contínua, limite de exposição crítica, efeito de tornar Cadeia de Suprimentos mensurável, um segredo atual e sensibilidade à postura estrutural em suas asserções. São verificações de regressão da implementação, escritas e executadas pelo fornecedor. Não são amostra de validação independente, teste de intrusão ou estudo de correlação do SQCM com resultados em campo.

Os identificadores da implementação auditada e os resultados das fixtures estão registrados na nota suplementar de validação, com linhas ilustrativas de cobertura. Código-fonte interno, coeficientes exatos e catálogos proprietários de regras não são disponibilizados com este paper. O suplemento oferece rastreabilidade, mas não reprodução independente irrestrita do instrumento proprietário completo.

7.2 Observações históricas de desempenho

O SQCM v2.0 relatou três observações de upload até veredito no fluxo rápido de CI, em um único commit: **5,25, 5,29 e 7,17 segundos**, usando um contêiner de staging com duas vCPUs e oito GB. Eram uma pequena amostra histórica, não uma distribuição de latência. A etapa relatada de leitura de cobertura não deve ser interpretada como execução completa de suítes arbitrárias nesse tempo.

Essas observações não foram repetidas para a v2.1. A cobertura conectada pode exigir instalação de dependências e execução de testes, e sua orquestração pode esperar até aproximadamente 20 minutos antes de recorrer à alternativa. Portanto, o exemplo anterior de cinco segundos não é uma promessa de latência completa do fluxo conectado atual.

O registro da análise profunda da v2.0 relatou aproximadamente 33, 67 e 101 segundos em oito, quatro e duas CPUs virtuais, respectivamente, para seu objeto Java de 277 arquivos-fonte. Esses valores também permanecem históricos. As duas lições anteriores

continuam úteis: contenção de CPU pode distorcer a atribuição de tempo por etapa, e resolução do polling pode distorcer o tempo aparente de conclusão. Nenhuma substitui uma nova campanha de benchmark.

7.3 Calibração histórica de detecção

A publicação anterior relatou 23 achados classificados em uma aplicação Java deliberadamente vulnerável, cinco achados reais em uma aplicação Node vulnerável e nenhum achado em uma aplicação Java de referência com 49 arquivos. Também relatou ausência de falsos positivos nos serviços internos e frontend examinados. O exercício anterior de refinamento preservou cinco caminhos positivos conhecidos enquanto reduzia os falsos positivos relatados de 137 para zero no objeto de calibração.

Esses resultados são mantidos como descrições do registro de desenvolvimento da v2.0, não como resultados reproduzidos na v2.1. Aplicações didáticas e código interno selecionado pelo autor não estabelecem precisão, revocação gerais ou ausência de vulnerabilidades. Falhas de codificação de caracteres e resolução de tipos encontradas naquele trabalho reforçam a necessidade de validar conjuntamente o artefato de análise distribuído e seu ambiente.

8. Limitações e agenda de pesquisa

A questão em aberto mais importante é a validade preditiva. Nem as faixas nem os pesos dos domínios foram estabelecidos por estudo longitudinal que relacione níveis SQCM a taxas de incidentes, custo de remediação ou resultados de manutenção. Estimativas de custo da má qualidade [14] motivam o problema, mas não calibram previsões financeiras no nível do repositório.

Outras limitações incluem cobertura incompleta da análise estática, atribuição incerta de IA, processamento limitado do histórico, execução possivelmente parcial dos testes, relatórios fornecidos pelo cliente, bases de advisories mutáveis, evidência alternativa assistida por modelo, agregação de múltiplos repositórios e aplicação incompleta de comparabilidade em algumas apresentações. Acesso ao repositório também não é acesso a cada artefato implantado ou à implementação de SaaS externos.

A agenda de validação é publicar um corpus independente mais amplo de defeitos, uma campanha de latência com versões fixadas em diferentes tamanhos e toolchains, proveniência e controles de comparação mais completos, contabilização mais clara da cobertura do projeto inteiro e uma coorte longitudinal com fatores de confusão documentados. São objetivos de pesquisa e implementação, não capacidades estabelecidas por esta publicação.

9. Histórico de versões e correções

Publicação	Contribuição principal	Registro permanente anterior
v1.0	Conceito de maturidade de código com quatro domínios e durabilidade de IA	10.5281/zenodo.21251863
v2.0	Cadeia de Suprimentos, limites dos artefatos e medições históricas do instrumento	10.5281/zenodo.21879222
v2.1	Método consolidado; cobertura medida no cálculo v2.2; correções de proveniência, escopo e comparabilidade	Esta publicação

As correções são substantivas: a cobertura medida é contínua em vez de uma categoria de cobertura alta; o núcleo exige três domínios para um snapshot pronto, não todos os quatro originais; evidência opcional pode alterar a pontuação geral normalizada; repetir a coleta não garante igualdade apenas pelo commit; a severidade de um segredo crítico é distinta da faixa de seu domínio; e a fundamentação bibliográfica é distinguida das métricas efetivamente implementadas. As referências [26] e [27] identificam suas versões e condições de publicação atualizadas.

O conceito de cinco domínios permanece. A contribuição desta revisão é uma descrição mais precisa do que o instrumento mede e das condições para usar seus números de maneira responsável.

Referências

A bibliografia de 34 itens é mantida para continuidade. As referências têm papéis distintos: normas e fundamentos conceituais, estudos empíricos, relatórios de fornecedores, notícias históricas de adoção e documentação de ferramentas ou políticas. A inclusão não implica validação independente do SQCM. Os títulos bibliográficos são preservados no idioma original.

[1] ISO/IEC 25010:2023. *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model*. International Organization for Standardization, 2023. https://www.iso.org/standard/78176.html

[2] ISO/IEC 5055:2021. *Information technology - Software measurement - Software quality measurement - Automated source code quality measures.* International Organization for Standardization, 2021 (developed by CISQ/OMG). https://www.iso.org/standard/80623.html · https://www.it-cisq.org/standards/code-quality-standards/

[3] Letouzey, J.-L. "The SQALE method for evaluating Technical Debt". *Third International Workshop on Managing Technical Debt (MTD 2012)*, IEEE, 2012, pp. 31-36. DOI: 10.1109/MTD.2012.6225997. See also: Letouzey, J.-L.; Ilkiewicz, M. "Managing Technical Debt with the SQALE Method". *IEEE Software* 29(6), 2012, pp. 44-51. https://ieeexplore.ieee.org/document/6225997

[4] Oman, P.; Hagemeister, J. "Construction and testing of polynomials predicting software maintainability". *Journal of Systems and Software* 24(3), 1994, pp. 251-266. DOI: 10.1016/0164-1212(94)90067-1. https://doi.org/10.1016/0164-1212(94)90067-1

[5] Heitlager, I.; Kuipers, T.; Visser, J. "A Practical Model for Measuring Maintainability". *QUATIC 2007*, IEEE, pp. 30-39. DOI: 10.1109/QUATIC.2007.7. Operationalization: *SIG/TÜViT Evaluation Criteria Trusted Product Maintainability.* https://dl.acm.org/doi/10.1109/QUATIC.2007.7 · https://www.softwareimprovementgroup.com/wp-content/uploads/SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainability.pdf

[6] CMMI Institute / ISACA. *Capability Maturity Model Integration (CMMI) V3.0*, 2023. https://cmmiinstitute.com/ · https://cmmiinstitute.com/learning/appraisals/levels

[7] Forsgren, N.; Humble, J.; Kim, G. *Accelerate: The Science of Lean Software and DevOps.* IT Revolution Press, 2018. ISBN 978-1942788331. Research program: https://dora.dev/

[8] Inozemtseva, L.; Holmes, R. "Coverage Is Not Strongly Correlated with Test Suite Effectiveness". *ICSE 2014*, ACM. DOI: 10.1145/2568225.2568271. https://dl.acm.org/doi/10.1145/2568225.2568271

[9] Just, R.; Jalali, D.; Inozemtseva, L.; Ernst, M. D.; Holmes, R.; Fraser, G. "Are Mutants a Valid Substitute for Real Faults in Software Testing?". *FSE 2014*, ACM. DOI: 10.1145/2635868.2635929. https://dl.acm.org/doi/10.1145/2635868.2635929

[10] Rahman, F.; Devanbu, P. "How, and Why, Process Metrics Are Better". *ICSE 2013*, IEEE. https://ieeexplore.ieee.org/document/6606589/

[11] Nagappan, N.; Ball, T. "Use of Relative Code Churn Measures to Predict System Defect Density". *ICSE 2005*, ACM, pp. 284-292. DOI: 10.1145/1062455.1062514. https://dl.acm.org/doi/10.1145/1062455.1062514

[12] Bird, C.; Nagappan, N.; Murphy, B.; Gall, H.; Devanbu, P. "Don't Touch My Code! Examining the Effects of Ownership on Software Quality". *ESEC/FSE 2011*, ACM, pp. 4-14. DOI: 10.1145/2025113.2025119. https://dl.acm.org/doi/10.1145/2025113.2025119

[13] Avelino, G.; Passos, L.; Hora, A.; Valente, M. T. "A Novel Approach for Estimating Truck Factors". *ICPC 2016*, IEEE. https://arxiv.org/abs/1604.06766

[14] Krasner, H. / CISQ. *The Cost of Poor Software Quality in the US: A 2022 Report.* Consortium for Information & Software Quality, December 2022. https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/

[15] Pearce, H.; Ahmad, B.; Tan, B.; Dolan-Gavitt, B.; Karri, R. "Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions". *IEEE Symposium on Security and Privacy (S&P) 2022*. https://arxiv.org/abs/2108.09293

[16] Perry, N.; Srivastava, M.; Kumar, D.; Boneh, D. "Do Users Write More Insecure Code with AI Assistants?". *ACM CCS 2023*. DOI: 10.1145/3576915.3623157. https://dl.acm.org/doi/abs/10.1145/3576915.3623157

[17] GitClear. *Coding on Copilot: 2023 Data Suggests Downward Pressure on Code Quality.* 2024. https://www.gitclear.com/coding_on_copilot_data_shows_ais_downward_pressure_on_code_quality

[18] GitClear. *AI Copilot Code Quality: 2025 Data Suggests 4x Growth in Code Clones.* 2025. https://www.gitclear.com/ai_assistant_code_quality_2025_research

[19] GitClear. *The Maintainability Gap: AI Code Quality in 2026.* January 2026. https://www.gitclear.com/the_ai_code_quality_maintainability_gap

[20] Peng, S.; Kalliamvakou, E.; Cihon, P.; Demirer, M. "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot". arXiv:2302.06590, 2023 (preprint). https://arxiv.org/abs/2302.06590

[21] DORA / Google Cloud. *Accelerate State of DevOps Report 2024.* https://dora.dev/research/2024/dora-report/

[22] DORA / Google Cloud. *State of AI-assisted Software Development 2025.* September 2025. https://dora.dev/dora-report-2025/

[23] Fortune. "Google's code is now more than 25% AI-generated, CEO Sundar Pichai says". October 2024. https://fortune.com/2024/10/30/googles-code-ai-sundar-pichai/

[24] CNBC. "Satya Nadella says as much as 30% of Microsoft code is written by AI". April 2025. https://www.cnbc.com/2025/04/29/satya-nadella-says-as-much-as-30percent-of-microsoft-code-is-written-by-ai.html

[25] Semafor. "Google CEO says 75% of company's new code is AI-generated". April 2026. https://www.semafor.com/article/04/24/2026/google-ceo-says-75-of-companys-new-code-is-ai-generated

[26] Liu, Y.; Widyasari, R.; Zhao, Y.; Irsan, I. C.; Chen, J.; Lo, D. "Debt Behind the AI Boom: A Large-Scale Empirical Study of AI-Generated Code in the Wild". arXiv:2603.28592v2, revised April 26, 2026 (preprint). https://arxiv.org/abs/2603.28592v2

[27] Sun, X.; Ståhl, D.; Sandahl, K.; Kessler, C. "Quality Assurance of LLM-generated Code: Addressing Non-Functional Quality Characteristics". Journal of Systems and Software, 2026. DOI: 10.1016/j.jss.2026.112885. Author version: arXiv:2511.10271v2, March 12, 2026. https://doi.org/10.1016/j.jss.2026.112885 · https://arxiv.org/abs/2511.10271v2

[28] OpenSSF. *SLSA - Supply-chain Levels for Software Artifacts.* Open Source Security Foundation. https://slsa.dev/

[29] Open Source Security Foundation. *OpenSSF Scorecard.* https://securityscorecards.dev/

[30] The White House. *Executive Order 14028: Improving the Nation's Cybersecurity.* May 2021 (established software bill of materials expectations for federal software supply). https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

[31] European Union. *Regulation (EU) 2024/2847 of 23 October 2024 - Cyber Resilience Act*, on horizontal cybersecurity requirements for products with digital elements. https://eur-lex.europa.eu/eli/reg/2024/2847/oj

[32] Google. *OSV - Open Source Vulnerabilities.* Distributed vulnerability database and schema. https://osv.dev/ · https://ossf.github.io/osv-schema/

[33] Joern - Open-source code analysis platform based on Code Property Graphs. Apache License 2.0. https://joern.io/ · https://github.com/joernio/joern

[34] GitHub. *CodeQL - Terms and Conditions.* Licensing conditions depend on the permitted use and applicable agreement; cited as historical context for the instrument selection, not a general statement of current entitlement. https://github.com/github/codeql-cli-binaries · https://docs.github.com/en/code-security/codeql-cli

---