ScaleQuality Code Maturity Model (SQCM)

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

Versão 1.0 · Julho de 2026

Autor: Erik Fernandes · Fundador, ScaleQuality

Contato: erik.fernandes@scalequality.io · scalequality.io

DOI: 10.5281/zenodo.21251863

Resumo executivo

Organizações de todos os tamanhos aceleraram a produção de software com IA generativa. A velocidade é real e mensurável. A capacidade de confiar no que foi produzido não acompanhou. Entre o final de 2024 e o início de 2026, a fração de código novo gerado por IA em uma das maiores empresas de tecnologia do mundo saltou de cerca de 25% para 75%, segundo declarações públicas de sua liderança [23][25]. No mesmo período, a pesquisa da indústria documentou duplicação de código crescendo, refatoração em queda acentuada e instabilidade de entrega persistente em times que adotaram IA [17][18][19][21][22].

Este paper apresenta o **ScaleQuality Code Maturity Model (SQCM)**, um modelo que responde a uma pergunta que os modelos existentes não respondem: **"posso confiar neste código?"**. O SQCM define *maturidade de código* como um construto mensurável no nível do repositório, avaliado exclusivamente a partir de **evidência observável** (estrutura, manifestos, histórico de versionamento, testes, automação e análise estática), sem questionários e sem auto-declaração, em minutos e sem configuração.

O modelo avalia quatro domínios: três herdados da tradição consolidada de qualidade de software (**Segurança**, **Confiabilidade** e **Manutenibilidade**, presentes na ISO/IEC 25010 [1] e na ISO/IEC 5055 [2]) e um domínio novo, que nenhum padrão anterior cobre: **Durabilidade de código IA**, a medida de quanto do código gerado por IA sobrevive no tempo em vez de ser retrabalhado ou abandonado. O veredito é expresso em uma escada de cinco níveis (Reativo, Inicial, Estruturado, Gerenciado, Otimizado), com pontuação por domínio, nível de confiança explícito por medida e reprodutibilidade garantida por versionamento de score.

O SQCM não é um detector de defeitos nem um modelo de processo organizacional. É um instrumento de medição de maturidade do artefato código, desenhado para a decisão prática que times enfrentam todos os dias na era da IA: o que este repositório aguenta, e o que falta para ele aguentar mais.

1. O problema: o código nasce mais rápido do que a capacidade de confiar nele

Três fatos, tomados em conjunto, definem o momento atual da engenharia de software.

Primeiro: a adoção de IA na escrita de código é massiva e acelera. Em outubro de 2024, o CEO do Google declarou que mais de 25% do código novo da empresa era gerado por IA e revisado por engenheiros [23]. Em abril de 2025, a Microsoft declarou algo entre 20% e 30% [24]. Em abril de 2026, o número declarado pelo Google já era 75% do código novo, ante 50% no outono anterior [25]. São números autodeclarados, sem metodologia pública, e a definição de "gerado por IA" é elástica (inclui sugestões de autocomplete aceitas). Ainda assim, como sinal de escala e de velocidade de mudança, são inequívocos: em 18 meses, a participação declarada triplicou.

Segundo: o ganho de velocidade é real. Em experimento controlado conduzido por pesquisadores do GitHub e da Microsoft, desenvolvedores com acesso ao GitHub Copilot completaram uma tarefa de implementação 55,8% mais rápido que o grupo de controle [20]. É um estudo de tarefa única, conduzido pelos próprios interessados e publicado como preprint, mas o efeito direcional foi replicado na prática percebida da indústria. O SQCM não parte de ceticismo quanto à velocidade: ela é o motivo pelo qual a adoção aconteceu.

Terceiro: a qualidade do que é produzido nessa velocidade tem perfil de risco documentado. Em avaliação sistemática de 2022, cerca de 40% dos programas gerados pelo GitHub Copilot em 89 cenários construídos sobre as fraquezas mais críticas do catálogo CWE continham vulnerabilidades [15]. Os cenários foram desenhados em torno de padrões de alto risco e os dados refletem os modelos de 2021, portanto o número deve ser lido como marco fundacional e não como taxa atual. Mais preocupante que a taxa em si é o efeito comportamental documentado por pesquisadores de Stanford em estudo publicado na ACM CCS 2023: participantes com acesso a um assistente de IA escreveram código significativamente menos seguro em quatro de cinco tarefas e, ao mesmo tempo, **acreditaram mais** que seu código estava seguro [16]. O risco não é apenas o código pior. É o código pior acompanhado de mais confiança.

A consequência econômica de qualidade ruim em software já era conhecida antes da IA generativa: o Consortium for Information & Software Quality estimou o custo da má qualidade de software nos Estados Unidos em pelo menos US$ 2,41 trilhões em 2022, dos quais aproximadamente US$ 1,52 trilhão em dívida técnica acumulada [14]. O número agrega categorias distintas (falhas operacionais, cibercrime viabilizado por defeitos, dívida técnica) e é uma estimativa de indústria, não um resultado revisado por pares. Mas a ordem de grandeza dispensa precisão: qualidade de código é um problema de bilhões, e a produção de código acaba de ser multiplicada.

A pergunta que emerge desse cenário não é "a IA escreve código bom ou ruim?". É outra, mais operacional: **dado um repositório concreto, hoje, o que ele aguenta?** Aguenta produção?

Aguenta mudança? Aguenta o próprio código que a IA gerou nele na semana passada? Essa pergunta precisa de uma régua. A seção seguinte mostra por que as réguas existentes não a respondem.

---

2. Por que as réguas existentes não respondem "posso confiar neste código?"

A tradição de medição de qualidade e maturidade de software é rica, e o SQCM se apoia nela. Mas cada família de modelos existentes responde a uma pergunta diferente da que está posta.

Modelos de qualidade de produto (ISO/IEC 25010). A ISO/IEC 25010:2023 define o modelo de qualidade de produto de software vigente, com nove características que incluem Segurança, Confiabilidade e Manutenibilidade [1]. É o vocabulário canônico da área, e o SQCM o adota deliberadamente. Mas a 25010 é uma taxonomia: diz *o que* observar, não *como* medir a partir de um repositório real, e não produz um veredito graduado.

Medição automatizada de qualidade de código (ISO/IEC 5055). A ISO/IEC 5055:2021, desenvolvida pelo CISQ, foi o primeiro padrão ISO a definir medidas de qualidade tomadas automaticamente da estrutura interna do código-fonte, em quatro características: Segurança, Confiabilidade, Eficiência de Performance e Manutenibilidade [2]. É o ancestral metodológico mais direto do SQCM: estabelece que medir qualidade direto do código, sem intermediação humana, é prática padronizável. Dois limites, porém: a 5055 mede a **qualidade presente** (violações estruturais existentes), não a maturidade das práticas que produzem qualidade futura; e não contém nenhuma disposição sobre código gerado por IA, um fenômeno posterior à sua publicação.

Notas de manutenibilidade por análise estática (SIG/TÜViT). O modelo prático de manutenibilidade da Software Improvement Group, publicado academicamente em 2007 e operacionalizado como certificação com ratings de 1 a 5 estrelas [5], é o precedente comercial mais próximo do SQCM: prova que atribuir uma **nota graduada** a código, derivada de análise estática e ancorada em um padrão (ISO 25010), é aceito pelo mercado e certificável. As diferenças definem o espaço que o SQCM ocupa: o modelo SIG cobre um único domínio (manutenibilidade), tradicionalmente opera em contexto de avaliação assistida, e não possui dimensão alguma de código IA.

Dívida técnica quantificada (SQALE, Maintainability Index). O método SQALE formalizou a quantificação de dívida técnica como custo de remediação derivado de análise de código [3], e é a base conceitual de ferramentas amplamente adotadas. O Maintainability Index, proposto em 1992-1994, mostrou que índices compostos calculados do código têm décadas de tradição [4]. A trajetória dessa família, porém, aponta na direção que o SQCM segue: índices únicos e opacos perderam espaço para modelos multidimensionais com características explícitas, exatamente o desenho da ISO 25010 e do modelo SIG [1][5].

Maturidade de processo (CMMI). O CMMI, hoje na versão 3.0, consolidou a ideia de maturidade em cinco níveis [6]. Sua herança conceitual é inegável, e a escada do SQCM é tributária dela. Mas o CMMI avalia o **processo organizacional**: políticas, práticas institucionais, capacidade de gestão. Duas organizações no mesmo nível CMMI podem abrigar repositórios em estados radicalmente diferentes. O CMMI responde "essa organização trabalha de forma madura?"; não responde "esse código específico é confiável?". Além disso, appraisals formais são investimentos de semanas, incompatíveis com a cadência de decisão da era da IA.

Métricas de entrega (DORA). O programa de pesquisa DORA demonstrou, com a mais longa pesquisa longitudinal do gênero, que capacidades técnicas observáveis predizem performance de entrega e resultados organizacionais [7]. O SQCM adota essa epistemologia (capacidades observáveis como preditores) mas aplica a lente ao artefato: DORA mede o fluxo de entrega do time; o SQCM mede o estado do código que esse fluxo produz.

Em síntese: sabemos medir qualidade presente do código (5055), dar nota de manutenibilidade (SIG), quantificar dívida (SQALE), avaliar processo (CMMI) e medir entrega (DORA). O que nenhum desses instrumentos responde é a pergunta composta que a era da IA tornou urgente: **este repositório, agora, com esta fração de código gerado por IA, tem as práticas e a estrutura que predizem comportamento confiável no tempo?** Essa é a pergunta do SQCM.

---

3. O construto: maturidade de código

3.1 Definição

Maturidade de código é a propriedade de um repositório que prediz seu comportamento ao longo do tempo: sua resistência a incidentes, sua capacidade de absorver mudança com segurança e a taxa com que o trabalho investido nele sobrevive em vez de ser retrabalhado.

Três distinções delimitam o construto:

1. **Maturidade não é qualidade instantânea.** Qualidade, no sentido da ISO 5055, é uma fotografia: as violações estruturais presentes no código hoje [2]. Maturidade é prognóstico: inclui a fotografia, mas pesa também as práticas que determinam as próximas fotografias (existência e eficácia de testes, automação de integração, disciplina de mudança legível no histórico). Um repositório pode ter poucas violações hoje e nenhuma prática que impeça sua degradação amanhã.
2. **Maturidade de código não é maturidade de processo.** O objeto é o repositório, não a organização [6]. A unidade de medição do SQCM permite que uma mesma organização enxergue repositórios em níveis diferentes, porque é isso que a realidade contém.

3. **Maturidade inclui, necessariamente, a dimensão da autoria por IA.** Na era em que a maior parte do código novo pode ser gerada por máquina [25], um modelo de maturidade que não distingue como o código gerado se comporta no tempo está medindo o passado.

3.2 Os quatro domínios

O SQCM avalia quatro domínios. Três são herdados diretamente da tradição ISO [1][2]:

- **Segurança**: o repositório expõe padrões exploráveis? Contém credenciais versionadas? As fraquezas presentes correspondem a categorias conhecidas de vulnerabilidade?
- **Confiabilidade**: o comportamento do código é verificado por testes? Os caminhos críticos têm cobertura? Existe integração contínua que capture regressões antes da entrega?
- **Manutenibilidade**: a estrutura sustenta mudança? Há tipagem, padrões arquiteturais discerníveis, disciplina de lint e formatação, histórico de commits legível?

O quarto domínio é a contribuição original do modelo:

- **Durabilidade de código IA**: das linhas geradas por IA introduzidas no repositório, quantas sobrevivem? Qual a taxa de retrabalho sobre elas? Como essa sobrevivência se compara à do código de autoria humana no mesmo repositório?

A ISO/IEC 5055 define quatro características, incluindo Eficiência de Performance [2]. O SQCM v1.0 deliberadamente não inclui performance: ela depende de carga de trabalho e de contexto de execução, e não é mensurável com honestidade a partir de sinais estáticos do repositório em uma avaliação zero-config. A seção 8 registra essa fronteira como limitação explícita e trabalho futuro.

3.3 Por que durabilidade de IA é um domínio de primeira classe

A evidência disponível, lida em conjunto, descreve um padrão consistente: código gerado por IA envelhece diferente.

A série de relatórios da GitClear, cobrindo centenas de milhões de linhas alteradas entre 2020 e 2026, documentou: churn (linhas revertidas ou substancialmente alteradas em menos de duas semanas) crescendo de 3,1% em 2020 para 5,7% em 2024; blocos duplicados aparecendo com frequência 8 vezes maior em 2024; a fração de código copiado-e-colado subindo de 8,3% para 12,3% entre 2021 e 2024, ultrapassando pela primeira vez a fração de código refatorado (movido), que caiu de 24,1% em 2020 para 9,5% em 2024 e 3,8% na edição de 2026; e a atividade de manutenção de código legado caindo 74% [17][18][19]. Registre-se a cautela devida: a GitClear é uma empresa comercial de análise de código, os dados são correlacionais e a metodologia é proprietária. O que dá peso à série é a consistência direcional através de três edições anuais e a especificidade dos números.

No plano acadêmico, o maior estudo empírico de código verificadamente gerado por IA "in the wild" até o momento (302,6 mil commits de cinco assistentes, em 6.299 repositórios do GitHub) encontrou mais de 484 mil problemas introduzidos, dos quais 89,3% code smells, com mais de 15%

dos commits de cada assistente introduzindo ao menos um problema; e, criticamente para o construto de durabilidade, **22,7% dos problemas introduzidos por IA sobrevivem até a versão mais recente dos repositórios** [26]. O estudo é um preprint (março de 2026), com a ressalva correspondente, mas seu achado central converge com a evidência de indústria: o que a IA introduz tende a ficar.

No plano da entrega, a pesquisa DORA documentou em 2024 que o aumento de adoção de IA estava associado a queda de 1,5% em throughput e de 7,2% em estabilidade de entrega [21]. Em 2025, com cerca de 90% dos ~5.000 respondentes usando IA, a relação com throughput tornou-se positiva, mas a relação negativa com estabilidade **persistiu**, e a tese central do relatório passou a ser que a IA amplifica o sistema existente: organizações sem controles de qualidade transformam volume de mudança em instabilidade [22]. A leitura fiel da evolução 2024 para 2025 importa: a velocidade se pagou; a estabilidade continua sendo o preço.

Finalmente, uma revisão sistemática de 108 estudos sobre qualidade de código gerado por LLMs, combinada com workshops com a indústria, encontrou que a pesquisa acadêmica se concentra em segurança e performance enquanto **manutenibilidade é a característica mais subestudada**, sendo justamente a prioridade declarada dos praticantes [27]. Existe, portanto, uma lacuna reconhecida entre o que se estuda e o que os times precisam medir. O domínio de durabilidade do SQCM é uma resposta operacional a essa lacuna.

3.4 Como o SQCM define durabilidade

A durabilidade no SQCM é medida como **taxa de sobrevivência**: das linhas atribuídas a autoria de IA introduzidas no repositório em uma janela de análise, a fração que permanece no código atual, comparada com a taxa equivalente para linhas de autoria humana no mesmo repositório, com decomposição por ferramenta de origem quando identificável.

Duas decisões de desenho merecem registro. Primeiro, a métrica é sempre **relativa** (taxas e proporções, nunca volumes absolutos): a literatura de predição de defeitos por churn estabeleceu que medidas absolutas de churn são maus preditores, enquanto medidas relativas normalizadas predizem densidade de defeitos com alta acurácia [11]. Segundo, a atribuição de autoria a IA é heurística (baseada em sinais de commit e padrões de contribuição) e o modelo a trata como tal: alimenta um domínio com nível de confiança explícito, nunca uma acusação por linha (a seção 6 detalha o tratamento de confiança).

---

4. O método: da evidência observável ao veredito

4.1 Princípio: evidência, não declaração

O SQCM avalia exclusivamente o que pode ser observado no repositório. Não há questionário, não há auto-avaliação, não há entrada declarativa de nenhum tipo na composição da nota. Essa escolha tem custo (há práticas valiosas que não deixam rastro observável) e o modelo o aceita em troca de uma propriedade que nenhum assessment declarativo tem: **a nota não depende do que o time acredita sobre si mesmo**. O estudo de Stanford citado na seção 1 é o argumento em uma frase: na era da IA, a confiança declarada e a segurança real divergem [16].

A epistemologia é a mesma validada pelo programa DORA: capacidades observáveis como preditores de resultado [7]. E cada família de sinal que o SQCM lê tem validade empírica publicada:

- **Sinais de teste**: a presença e a proporção de testes informam, mas o SQCM segue a literatura ao não tratar cobertura como proxy suficiente de eficácia. A correlação entre cobertura e eficácia de suíte, controlando o tamanho da suíte, é baixa a moderada [8]. Por isso cobertura entra como sinal parcial, com confiança explícita, e a presença de **mutation testing** (cuja detecção de mutantes correlaciona com detecção de falhas reais [9]) é tratada como sinal de nível superior.
- **Sinais de histórico**: métricas de processo extraídas do versionamento predizem defeitos melhor que métricas estáticas de código [10]. O SQCM lê o histórico: disciplina de commits, distribuição de contribuição (cuja relação com falhas é documentada [12]), concentração de conhecimento estimável por truck factor [13], e os sinais de sobrevivência de linhas que alimentam a durabilidade [11].
- **Sinais estruturais e de automação**: presença e configuração de integração contínua, tipagem estática, lint e formatação, containerização, infraestrutura como código, e padrão arquitetural discernível da estrutura de diretórios.
- **Sinais de análise estática**: achados de segurança, confiabilidade e manutenibilidade produzidos por scanners estabelecidos de código e de segredos, com proveniência registrada por achado (qual ferramenta produziu qual evidência).

4.2 Composição da nota

Cada domínio recebe uma pontuação de 0 a 100, composta a partir dos sinais pertinentes segundo três princípios publicáveis (os pesos exatos são propriedade do modelo e ficam fora deste paper, pela mesma razão que o questionário de scoring de um appraisal comercial não é público):

1. **Ponderação por severidade e por domínio.** Um mesmo achado pesa diferente conforme o domínio que afeta: uma fraqueza de segurança de severidade alta pesa mais no domínio Segurança do que um achado informacional de estilo pesa em Manutenibilidade. A ponderação é monotônica na severidade reportada pelo scanner.

2. **Tetos de penalidade por classe de evidência.** Nenhuma classe isolada de achados de análise estática pode, sozinha, zerar um domínio. O modelo impõe tetos que impedem que volume de achados de baixa severidade domine o veredito.

3. **Exceções deliberadas para evidência crítica.** A exceção ao princípio anterior é explícita e documentada: a presença de **credenciais versionadas no repositório** (segredos) rebaixa o domínio de Segurança à faixa crítica por si só. Um repositório com chaves expostas não é um repositório maduro, independentemente de qualquer outra virtude.

As pontuações de domínio agregam-se em uma pontuação geral, e as faixas de status são públicas e visíveis no produto: abaixo de 60, crítico; de 60 a 79, atenção; 80 ou acima, saudável.

4.3 Zero-config como requisito, não conveniência

Uma avaliação SQCM parte de um apontamento (URL de repositório, conexão de provider ou upload de arquivo) e produz o veredito em minutos, sem nenhuma configuração. Isso não é apenas experiência de usuário: é um requisito epistemológico do modelo. Avaliações que exigem configuração extensa medem, em parte, a qualidade da configuração. Avaliações que exigem semanas medem um repositório que já mudou. A janela de decisão da era da IA (em que 18 meses triplicaram a participação de código gerado [23][25]) exige que a medição caiba dentro do ciclo de mudança que pretende medir.

---

5. A escada: cinco níveis de maturidade

A pontuação geral converte-se em um nível de 1 a 5, com denominações que descrevem o estado do repositório:

Nível	Nome	Faixa	Significado
1	**Reativo**	< 40	O repositório não contém as estruturas mínimas de confiança. Qualidade, quando existe, é esforço individual e invisível. Incidentes são descobertos por usuários.
2	**Inicial**	40 a 59	Existem fragmentos de prática (alguns testes, alguma automação), sem sistematicidade. O comportamento sob mudança é imprevisível.
3	**Estruturado**	60 a 79	As práticas fundamentais existem e operam: testes com presença real, integração contínua, disciplina estrutural discernível. O repositório sustenta mudança rotineira.
4	**Gerenciado**	80 a 89	As práticas são consistentes e verificáveis em todos os domínios, com sinais fortes de teste e automação. O repositório sustenta mudança sob pressão.
5	**Otimizado**	≥ 90	Excelência mensurável nos quatro domínios, incluindo práticas de verificação avançadas. O repositório é ativo estratégico, não passivo de risco.

Os thresholds são publicados deliberadamente. Eles são visíveis no produto (a nota e o nível aparecem juntos) e escondê-los seria falsa opacidade. A herança conceitual da escada de cinco níveis vem do CMMI [6]; a diferença está no objeto (o repositório, não a organização) e no método (evidência observável, não appraisal).

Cada transição de nível é acompanhada, no produto, da lista concreta do que a destravaria: quais riscos abertos, quais práticas ausentes, qual domínio segura o avanço. A escada não é apenas classificação; é backlog priorizado.

---

6. Confiança, estados não medidos e reprodutibilidade

Três propriedades do SQCM existem para proteger a honestidade do veredito, e são parte do modelo tanto quanto a nota.

Confiança explícita por domínio. Cada domínio carrega um nível de confiança (alto, médio, baixo) que reflete a densidade e a qualidade da evidência disponível. Um repositório sem histórico git acessível produz um domínio de durabilidade com confiança baixa; leituras assistidas por IA (como a postura direcional de cobertura) são sempre rotuladas como tais e nunca elevam a confiança acima do que a evidência determinística sustenta. A motivação vem da própria literatura: se cobertura sozinha não garante eficácia [8], um modelo honesto precisa dizer com que confiança está falando, medida a medida.

Estados não medidos não pontuam. Quando a evidência de um domínio é insuficiente, o domínio é declarado como não medido, e o veredito geral o reflete. O SQCM não interpola, não assume médias e não converte ausência de evidência em nota neutra. Tendências temporais só são registradas quando todos os quatro domínios estão efetivamente medidos, para que a série histórica nunca misture medições incompletas com completas.

Reprodutibilidade por versionamento de score. Todo veredito registra a versão do modelo que o produziu, e vereditos passados são imutáveis: a evolução do modelo (novas versões de pesos, novos sinais) se aplica a medições futuras, nunca reescreve o histórico. Duas execuções da mesma versão do modelo sobre o mesmo estado de repositório produzem o mesmo resultado. Essa disciplina é o que permite comparar a trajetória de um repositório no tempo sabendo que a régua não se moveu silenciosamente no meio do caminho.

Registre-se também o papel da IA dentro do próprio motor de avaliação, porque a distinção importa: a IA é usada como **extrator e sintetizador de sinal** (por exemplo, na leitura direcional de lacunas de teste), sempre com rótulo de confiança; a composição da nota, os pesos, os tetos e os thresholds são determinísticos. O SQCM usa IA para ler evidência; não usa IA para opinar sem evidência.

---

7. Da nota à ação

Um modelo de maturidade que termina na nota é um relatório. O SQCM foi desenhado para operar, e três mecanismos fazem a ponte entre medição e mudança:

Riscos com impacto de negócio. Cada achado relevante é expresso como risco, com severidade, localização e uma tradução de impacto em linguagem de negócio (o que este padrão de código pode causar, a quem). A lista de riscos de um veredito é a agenda de trabalho do repositório, ordenada por severidade.

Medição contínua com linha de corte. O veredito pontual (a fotografia) pode ser promovido a medição contínua (o filme): o repositório é reavaliado a cada mudança, e a organização define uma **linha de maturidade** abaixo da qual a esteira de entrega falha o build. A maturidade deixa de ser observação e vira contrato operacional. A escolha da linha é da organização; o modelo fornece a régua e o mecanismo.

Remediação executável. Os riscos identificados pelo veredito alimentam agentes de remediação que propõem correções como pull requests reais (testes ausentes, correções de integração contínua, correções de defeitos), sujeitos a revisão humana. Os detalhes desse mecanismo estão fora do escopo deste paper; o ponto arquitetural relevante é que a mesma evidência que produz a nota produz o plano, e o mesmo sistema que mede pode executar parte da recuperação, com o humano no circuito de aprovação.

---

8. Limitações e agenda de validação

Um modelo que pretende medir honestidade precisa declarar as próprias fronteiras. As limitações do SQCM v1.0, e o que pretendemos fazer com cada uma:

1. Evidência observável tem pontos cegos. Práticas valiosas que não deixam rastro no repositório (revisão de design, pareamento, gestão de incidentes) não entram no veredito. O SQCM mede o que o repositório mostra; organizações que precisam avaliar processo devem usar instrumentos de processo [6]. Não há plano de incorporar auto-declaração: o ponto cego é o preço do princípio.

2. Performance está fora do escopo da v1.0. Das quatro características da ISO 5055 [2], Eficiência de Performance não é avaliada, porque não é mensurável com honestidade a partir de sinais estáticos zero-config. Trabalho futuro: incorporação de sinais de performance quando houver telemetria de execução conectável, com a mesma disciplina de confiança explícita.

3. A atribuição de autoria a IA é heurística. A detecção de linhas geradas por IA baseia-se em sinais de commit e padrões de contribuição, e tem taxa de erro. Por isso a durabilidade opera em taxas agregadas e comparativas (IA versus humano no mesmo repositório), nunca em julgamento por linha, e carrega confiança explícita. Trabalho futuro: à medida que ferramentas de geração

adotarem marcação padronizada de proveniência, a heurística será substituída por atribuição declarativa verificável.

4. Os thresholds da v1.0 são calibração de projeto, não resultado empírico. As faixas de nível (40/60/80/90) e a semântica da escada derivam da tradição de cinco níveis [6] e da experiência de projeto, não de um estudo longitudinal próprio. A agenda de validação, nesta ordem: (a) correlacionar, em coorte de repositórios medidos continuamente, o nível SQCM com resultados observáveis (incidência de defeitos, churn subsequente, sobrevivência de mudanças), na linha da literatura de predição [10][11]; (b) publicar a distribuição de níveis por população de repositórios para calibração pública das faixas; (c) revisar os pesos à luz dos resultados, com incremento da versão de score e preservação do histórico (seção 6).

5. A evidência de terceiros que fundamenta o domínio de durabilidade tem vieses conhecidos. A série GitClear provém de empresa com interesse comercial no tema [17][18][19]; os estudos acadêmicos mais recentes são preprints [26][27]; os números de adoção são autodeclarações de executivos [23][24][25]. Este paper os cita com essas qualificações porque é a evidência que existe; a agenda do item 4 existe precisamente para que o SQCM produza, com o tempo, evidência primária própria.

6. Correlação não é causalidade, aqui também. O SQCM prediz; não prova que elevar a nota causa os resultados. A relação capacidades-resultados em que o modelo se apoia é a mais bem documentada da área [7], e a mesma cautela que a DORA declara sobre seus surveys [21][22] vale para este modelo.

---

9. Conclusão

A engenharia de software atravessou transições de escala antes: da montagem artesanal aos processos (CMMI), da qualidade declarada à qualidade medida no código (ISO 5055), da intuição de entrega à ciência de entrega (DORA). Cada transição exigiu uma régua nova porque a anterior media a pergunta errada.

A transição atual tem uma assinatura clara: o código passou a nascer mais rápido do que a capacidade de confiar nele. As réguas existentes medem qualidade presente, processo organizacional ou fluxo de entrega. Nenhuma responde à pergunta que times, líderes e investidores fazem diante de um repositório na era da IA: **isto aguenta?**

O ScaleQuality Code Maturity Model é a nossa resposta: maturidade de código como construto de primeira classe, medido por evidência observável, sem declaração, em minutos, com a dimensão de durabilidade de código IA que nenhum padrão anterior cobre, confiança explícita em cada medida e uma escada que converte diagnóstico em plano. A versão 1.0 é o começo deliberado de um programa: medir, publicar, validar contra resultados, recalibrar em versões futuras sem reescrever o passado.

O modelo está em operação em scalequality.io. A metodologia continuará publicada, versionada e aberta ao escrutínio, porque é isso que separa uma régua de uma opinião.

Referências

[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 (desenvolvida pelo 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. Ver também: 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. Operacionalização: *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. Programa de pesquisa: 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, dezembro de 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*. Janeiro de 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*. Setembro de 2025. https://dora.dev/dora-report-2025/

[23] Fortune. "Google's code is now more than 25% AI-generated, CEO Sundar Pichai says". Outubro de 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”. Abril de 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”. Abril de 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:2603.28592, março de 2026 (preprint). https://arxiv.org/abs/2603.28592

[27] “Quality Assurance of LLM-generated Code: Addressing Non-Functional Quality Characteristics”. arXiv:2511.10271, novembro de 2025 (preprint). https://arxiv.org/abs/2511.10271
