ScaleQuality Code Maturity Model (SQCM)

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

Versão 3.0 · Outubro de 2026

Erik Fernandes · ScaleQuality

Calibrações de pontuação descritas: v2.2 (padrão) e mainframe-v1.0

DOI: 10.5281/zenodo.23106322

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 3.0 amplia o modelo em duas direções. Primeiro, introduz perfis de pontuação e o primeiro perfil além do padrão: o **SQCM Mainframe 1.0**, uma régua própria para parques de código formados sobretudo por COBOL e controle de jobs, com domínios, faixas de métricas, pesos e travas de nível próprios, calibrado em nove repositórios COBOL públicos. Segundo, documenta a **evidência de licenças**: como o instrumento inventaria componentes de terceiros, avalia suas licenças contra a política da organização e produz registros exportáveis. Os resultados de licença ficam fora de toda pontuação numérica. O perfil padrão e sua calibração **v2.2** não mudam e são mantidos da v2.1, que esta edição substitui.

Nota de publicação. Este documento substitui o SQCM v2.1, DOI 10.5281/zenodo.22781322. 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. Existem agora duas sequências de pontuação: v2.2 para o perfil padrão e mainframe-v1.0 para o perfil Mainframe. As sequências de publicação e de pontuação continuam 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.

Perfis de pontuação. Um perfil de pontuação é uma régua completa: domínios, métricas, faixas, pesos, regras de composição e regras de nível. Cada execução é pontuada por exatamente um perfil, e a execução o registra pela sua versão de pontuação. Os perfis compartilham a escala de 0 a 100 e as faixas de nível para que um portfólio possa ser lido num só eixo, mas pontuações produzidas por perfis diferentes não são comparáveis como medições (seção 5.3). As seções 2 a 6 descrevem o perfil padrão; a seção 7 descreve o perfil Mainframe.

Evidência fora da pontuação. Parte da evidência apoia decisões sem entrar em nenhuma pontuação. A evidência de licenças é o caso principal: se uma licença é aceitável depende de como o software é usado e da política da organização, então o mesmo componente pode ser aceitável para uma organização e não para outra. A seção 8 descreve essa evidência e por que ela fica fora do resultado numérico.

2. Os cinco domínios do perfil padrão

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.
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**; a seção 8 descreve essa evidência.

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. Todo perfil de pontuação usa as mesmas faixas. Um perfil pode acrescentar travas de nível que seguram o nível abaixo da faixa que sua pontuação alcança; o perfil Mainframe faz isso (seção 7.5). 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.

5.3 Duas réguas de pontuação

Uma execução pontuada pelo perfil Mainframe registra scoreVersion mainframe-v1.0; uma execução pontuada pelo perfil padrão registra v2.2. Os dois números compartilham escala e faixas de nível, não uma medição: um 80 mainframe e um 80 padrão saem de domínios, métricas e pesos diferentes. A regra de comparabilidade da seção 5.2 vale entre perfis exatamente como vale entre calibrações de um mesmo perfil, e a mesma lógica de evolução que suprime diferenças entre versões de pontuação trata uma troca de perfil como troca de versão.

Um projeto pode mudar de perfil quando sua composição muda, por exemplo quando o código moderno acrescentado numa migração passa a predominar nas linhas lidas. Sua série então recomeça sob a nova régua. Execuções de projetos mainframe registradas antes da calibração do perfil Mainframe não têm pontuação e não são repontuadas retroativamente.

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.

Nenhum desses analisadores de uso geral lê COBOL, JCL ou definições CICS. Os fontes mainframe são lidos pelo leitor próprio descrito na seção 7.3.

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. O perfil Mainframe (SQCM Mainframe 1.0)

7.1 Por que um perfil próprio

O perfil padrão foi calibrado em repositórios construídos com gerenciadores de pacotes, executores de testes, pipelines de CI e histórico Git. Um parque mainframe se organiza de outro jeito. Layouts de registro compartilhados ficam em copybooks, não em pacotes. Programas são iniciados por controle de jobs (JCL) e por definições de transação, não por um ponto de entrada da aplicação. Testes unitários são raros. Os fontes muitas vezes são exportados de um gerenciador de bibliotecas, sem histórico Git.

Pontuado pela régua padrão, um sistema assim recebe um número cujas entradas significam outra coisa: não há manifestos de dependências, não há frameworks de teste e os analisadores gerais não leem COBOL. O perfil Mainframe substitui essas entradas por sinais próprios da plataforma e mantém a escala e as faixas de nível compartilhadas. Analistas do setor esperam que mais de 70% dos projetos de saída de mainframe iniciados em 2026 não entreguem os benefícios pretendidos, por superestimar a IA generativa [35]. Este paper não testa essa previsão; o perfil mede o parque de onde uma migração parte.

7.2 Quando o perfil se aplica

O perfil se aplica a um projeto quando todo repositório medido contém fonte mainframe e quando as linhas de código COBOL, as linhas de JCL e as linhas de PL/I somam pelo menos metade das linhas lidas nesses repositórios. O COBOL é contado em linhas de código, sem comentários; JCL, PL/I e arquivos com 34 extensões comuns de linguagens modernas são contados em todas as suas linhas. Módulos Assembler são inventariados, mas não contados. Projetos que não atendem a alguma das condições mantêm o perfil padrão, incluindo parques mistos em que o código moderno predomina e projetos em que um repositório não tem fonte mainframe.

7.3 O que o leitor lê

Nenhum analisador público do motor entende esses formatos, por isso o perfil usa um leitor léxico próprio. Ele lê texto. Não compila, não expande copybooks dentro dos programas, não resolve macros e não executa nada. O COBOL em formato fixo é lido na área de código das colunas 8 a 72, respeitando o indicador da coluna 7 para linhas de comentário, de depuração e de continuação [36]. Fontes em formato livre são reconhecidos pela diretiva de formato de fonte.

Fonte	Reconhecido por	O que é extraído
Programas COBOL	Extensões comuns de programa, ou membros exportados cujo cabeçalho contém IDENTIFICATION DIVISION e PROGRAM-ID	Identificadores de programa, inclusive programas aninhados; COPY e EXEC SQL INCLUDE; CALL estático e dinâmico; LINK e XCTL do CICS; arquivos declarados e tabelas DB2; parágrafos, seções, GO TO e ALTER; contagens de tratamento de erro
Copybooks	Extensões comuns de copybook	Quais programas e copybooks incluem cada um; uma impressão digital do seu código
Jobs e procedures JCL	Extensões, ou um comando JOB ou PROC no cabeçalho	Steps e os programas que iniciam, procedures, data sets e senhas escritas no JCL
Definições de recursos CICS	Arquivos de definição, ou entrada do DFHCSDUP dentro de um job [37]	Transações e os programas que iniciam
Mapas BMS	Arquivos de mapa, ou montagens de mapa dentro de um job [38]	Mapsets e mapas; o mapa simbólico de descrição gerado a partir de um mapa, que os programas copiam, conta como presente
Assembler e PL/I	Suas extensões	Nomes de módulo, pelo rótulo de entrada principal ou pelo nome do arquivo; inventariados, não analisados

Pontos de entrada são os programas iniciados de fora do código: steps de JCL, incluindo o programa de aplicação indicado num step do controlador de região do IMS e um programa indicado por um comando RUN PROGRAM do DB2; procedures; e transações CICS. A alcançabilidade é calculada a partir desses pontos de entrada pelos destinos estáticos de CALL, LINK e XCTL. Uma chamada ou transferência por variável só é resolvida quando o nome de um programa do repositório aparece como constante literal no programa chamador, numa cláusula VALUE ou num comando MOVE. Caso contrário, ela continua dinâmica e é contada como tal.

Um programa que nenhuma referência estática alcança é relatado como **não referenciado**, nunca como código morto, porque o despacho dinâmico não pode ser excluído lexicamente. Sem nenhum ponto de entrada, a alcançabilidade não é calculada e a proporção de não referenciados fica indisponível.

Um parágrafo só é contado como **morto** quando o controle não consegue alcançá-lo: nenhum PERFORM, intervalo PERFORM THRU ou GO TO o nomeia, sua seção não é executada por PERFORM e ele não é alcançado por queda sequencial a partir do início da procedure division, de um destino de GO TO ou dentro de uma seção executada. A queda sequencial para num parágrafo que deixa o fluxo por GOBACK, STOP RUN, EXIT PROGRAM, GO TO, ou RETURN ou XCTL do CICS. Parágrafos com sufixo -EXIT não são contados.

Módulos de sistema e de fornecedor, como programas utilitários, Language Environment, interfaces de CICS, DB2, IMS e MQ e serviços do z/OS, são excluídos da lista de fontes ausentes por padrões de nome. Chamadas entre programas aninhados de um mesmo fonte ficam dentro desse fonte. **Programas gerados** são reconhecidos por marcas de gerador no cabeçalho; continuam no inventário, mas ficam fora das métricas da seção 7.4. **Testes unitários** só são reconhecidos por marcas explícitas: suítes do COBOL Check, identificadas pela convenção de arquivos .cut desse projeto [39], e programas que chamam o runtime do IBM zUnit [40]. Um nome de arquivo com "test" não é evidência de teste.

Cada programa e copybook recebe também uma impressão digital da sua área de código, estável a edições de comentário e à renumeração de sequência. Um fluxo de modernização, fora da pontuação, a usa para detectar quais objetos do legado mudaram entre duas medições.

7.4 Domínios, métricas e faixas

Cada métrica vai de 0 a 100 em linha reta entre o valor que pontua 100 e o valor que pontua 0, com limite nas duas pontas. Uma métrica de cobertura lida em menos de cinco comandos não conta. Um domínio sem nenhuma métrica que conte é excluído e os pesos restantes são renormalizados, como no perfil padrão.

Domínio	Métrica	Pontua 100 em	Pontua 0 em
Confiabilidade	Arquivos declarados com FILE STATUS	100%	50%
Confiabilidade	Comandos CICS de dados com RESP ou NOHANDLE, ou num programa com HANDLE CONDITION ou HANDLE ABEND	100%	50%
Confiabilidade	Comandos SQL conferidos por SQLCODE ou SQLSTATE em até 15 linhas de código, ou WHENEVER SQLERROR	100%	50%
Modernizabilidade	Nomes referenciados sem fonte no checkout	0%	25%
Modernizabilidade	Programas que nenhum ponto de entrada alcança	5%	50%
Modernizabilidade	Programas com chamada ou transferência dinâmica sem destino resolvido	0%	20%
Modernizabilidade	Pontos de entrada conhecidos por JCL ou definições CICS	sim	não
Manutenibilidade	Comandos GO TO por mil linhas de código	2	25

Domínio	Métrica	Pontua 100 em	Pontua 0 em
Manutenibilidade	Lógica repetida, em janelas de 8 comandos normalizados	10%	50%
Manutenibilidade	Parágrafos mortos	2%	20%
Manutenibilidade	Programas acima de 2.000 linhas de código	0%	25%
Manutenibilidade	Mediana da média de linhas de código por parágrafo	30	80
Segurança	SQL dinâmico (PREPARE, EXECUTE IMMEDIATE) por mil linhas de código	0	1
Segurança	Senha escrita no JCL	nenhuma	qualquer
Testabilidade	Fontes de teste unitário por programa	0,5	0

A **Confiabilidade** faz a média das três coberturas ponderada pelo número de comandos que cada uma leu, de modo que 40 comandos SQL pesam mais que 5 arquivos declarados. Comandos CICS de dados são READ, WRITE, REWRITE, DELETE, os comandos de navegação e os comandos de fila; transferências como LINK e XCTL não contam. **Modernizabilidade** e **Manutenibilidade** são médias simples das métricas disponíveis. A **Segurança** é zero quando há senha escrita no JCL; caso contrário, segue a faixa de SQL dinâmico. A **Testabilidade** segue a razão entre fontes de teste unitário e programas.

Os programas de aplicação são as entradas das métricas de tratamento de erro, GO TO, tamanho, parágrafos e ALTER; programas gerados e programas de teste unitário ficam de fora. A lógica repetida é calculada sobre as procedure divisions dos programas de aplicação, porque declarações de dados se repetem por natureza quando vários programas leem o mesmo layout de registro.

A proporção de fontes ausentes divide os nomes referenciados que não têm fonte no checkout pelos nomes distintos que os programas referenciam por chamadas estáticas, transferências e copybooks. Programas que só um step de JCL ou uma definição CICS nomeia são contados como ausentes quando faltam, mas não entram no denominador. A proporção é, portanto, conservadora e pode passar de 100% num checkout que contém sobretudo definições de jobs. Alinhar o denominador a todos os tipos de referência está previsto para o perfil 1.1.

7.5 Pesos, composição e travas de nível

Domínio	Peso
Confiabilidade	30
Modernizabilidade	25
Manutenibilidade	25
Segurança	10
Testabilidade	10

A pontuação geral é a média ponderada dos domínios disponíveis. A Segurança pesa menos do que num repositório moderno porque o leitor tem hoje dois sinais de segurança, e os dois deram zero em todos os sistemas do corpus de calibração. Um domínio que mede pouco empurraria a pontuação para cima sem informação. A Segurança continua agindo por uma trava.

O nível segue as faixas compartilhadas em 40, 60, 80 e 90, sujeito a duas travas:

Nível	Condição adicional
3 em diante	No máximo 5% dos nomes referenciados sem fonte; nenhum comando ALTER em programas de aplicação; nenhum sinal de segurança; pelo menos um ponto de entrada conhecido
4 em diante	As condições do nível 3, pelo menos uma fonte de teste unitário e medição contínua ligada no projeto

Um sistema que não atende a uma condição do nível 3 fica no nível 2, qualquer que seja a pontuação. O ALTER modifica em execução o destino de um GO TO, de modo que o fluxo de controle não pode ser lido no fonte [42]. Ele era um elemento obsoleto do padrão COBOL de 1985 e foi removido na revisão de 2002 do padrão, embora o IBM Enterprise COBOL ainda o aceite [41].

O motor não consegue observar se um projeto mede continuamente. Por isso registra o nível sem medição contínua, junto com o nível que o projeto alcança com ela, e o serviço que guarda a configuração do projeto aplica o segundo quando a medição contínua está ligada. Num parque mainframe, medição contínua significa um pipeline quando o código fica no Git, ou uma conexão que envia cada promoção a partir de um gerenciador de bibliotecas. Essa conexão ainda não havia sido validada numa instalação de cliente quando este texto foi escrito. Um sistema medido apenas por arquivos enviados não tem medição contínua e, portanto, fica no nível 3 ou abaixo.

Durabilidade de Código de IA e Cadeia de Suprimentos não fazem parte deste perfil. O histórico Git costuma faltar nos fontes mainframe, e manifestos de dependências não existem. Os riscos de prática do perfil padrão sobre ausência de testes, de camada de tipos e de configuração de CI também não são relatados neste perfil, porque a evidência mainframe correspondente é lida diretamente.

7.6 Corpus de calibração e resultados

As faixas foram definidas em 30 de setembro de 2026 com nove repositórios públicos. Foram escolhidos por serem públicos e conterem COBOL de mainframe, a maioria com JCL, CICS ou DB2. As revisões abaixo foram registradas para que a leitura possa ser repetida; os resultados foram reproduzidos nessas revisões em 2 de outubro de 2026, com pontuações idênticas. Dois repositórios, uma demonstração de fornecedor e um benchmark sintético, não são nomeados (ver abaixo).

Repositório	Tipo	Revisão	Linhas de código COBOL	Programas
aws-samples/aws-mainframe-modernization-carddemo	Demonstração	59cc6c2fd7eb	32.562	44
cicsdev/cics-genapp	Demonstração	f6f3f4b2580d	6.794	31
Demonstração de fornecedor (não nomeada)	Demonstração de fornecedor	omitida	26.083	81
IBM/Bank-of-Z	Demonstração	ff2a59181bd8	23.697	43
etalab/taxe-fonciere	Código de produção publicado	6bd40b2d78ea	2.997	6
Benchmark sintético (não nomeado)	Benchmark sintético	omitida	6.177	42
walmartlabs/zECS	Serviço publicado	6d6bcbbc89c9	2.670	5
IBM/dbb	Amostras	ce2c1be58eb0	4.621	6
IBM/zOS-Client-Web-Enablement-Toolkit	Amostras	5117ca15e45e	1.319	1

As linhas de código COBOL incluem copybooks e excluem comentários; o corpus soma 106.920 [43]. O código da taxe foncière é o cálculo do imposto predial publicado pela administração tributária francesa. O zECS é um serviço de cache para z/OS publicado por engenheiros de uma varejista, sem declaração sobre uso em produção. A demonstração de fornecedor é distribuída sob uma licença que restringe seu uso a demonstrações, e o benchmark é um repositório público de código sintético gerado com um modelo de linguagem e publicado sem arquivo de licença; nenhum dos dois é nomeado aqui. O benchmark e as duas coleções de amostras serviram para exercitar o leitor; não são evidência sobre parques em operação. Uma das coleções de amostras, IBM/dbb, tem mais linhas modernas que mainframe e seria pontuada pelo perfil padrão pela regra da seção 7.2.

Sistema	Conf.	Mod.	Manut.	Seg.	Test.	Pontuação	Nível pela pontuação	Nível
CardDemo	97	85	83	100	0	81	4	2
GenApp	94	100	70	100	0	81	4	3
Demonstração de fornecedor	40	75	88	100	0	63	3	3
Bank of Z	88	63	67	100	0	69	3	2
taxe foncière	n/d	33	100	100	0	62	3	2
Benchmark sintético	79	65	100	100	0	75	3	2
zECS	100	92	94	100	0	87	4	3
Amostras DBB	n/d	39	97	100	0	63	3	2
Web Enablement Toolkit	n/d	67	89	100	0	70	3	2

O nível mostrado é o nível sem medição contínua, já que nenhum repositório público do corpus mede continuamente. Algumas observações orientaram a calibração:

- **As faixas separam os sistemas pelo motivo declarado.** A demonstração de fornecedor tem a menor Confiabilidade, 40, porque 39% dos seus comandos CICS de dados e 32% dos seus comandos SQL não são seguidos de tratamento de erro. O zECS chega a 100.
- **As travas pesam mais que a média numa decisão de migração.** O CardDemo pontua 81 e fica no nível 2: contém quatro comandos ALTER, e suas definições CICS nomeiam seis programas cujo fonte não está no repositório. O Bank of Z e o código da taxe foncière não expõem nenhum ponto de entrada que o leitor consiga resolver: o Bank of Z não traz definições CICS e seu único job inicia um programa PL/I, e o código da taxe foncière não traz JCL.
- **Contexto ausente reduz a Modernizabilidade.** O código da taxe foncière chama versões de anos anteriores dos seus módulos de cálculo cujos fontes não estão no repositório; faltam 59% dos nomes que ele referencia.
- **Dois domínios ainda não discriminam.** Segurança deu 100 e Testabilidade deu 0 em todos os sistemas. Os dois resultados estão corretos para este corpus, mas suas faixas só poderão ser calibradas quando forem medidos sistemas com esses sinais.

7.7 O que a calibração corrigiu no leitor

A calibração em código público foi, antes de tudo, um teste do leitor. Cada número que não batia com o fonte foi conferido no arquivo antes de uma regra mudar:

Correção	Efeito no corpus
Definições CICS instaladas por um job DFHCSDUP passam a ser lidas	GenApp: transações de 0 para 25, pontos de entrada de 0 para 10
Parágrafos mortos respeitam seções executadas e queda sequencial	Bank of Z: parágrafos mortos de 59,1% para 9,6%
Copybooks gerados a partir de mapas BMS contam como presentes	GenApp: de um copybook ausente para nenhum
Chamadas entre programas aninhados ficam dentro do fonte	Web Enablement Toolkit: de 22 nomes ausentes para nenhum; DBB: de 9 para nenhum
Módulos de sistema (CICSPlex SM, Workload Simulator, z/OS web enablement, Health Checker, runtime C) são excluídos	GenApp: de 5 programas ausentes para nenhum
Fontes PL/I passam a ser inventariados	Bank of Z: um programa PL/I chamado pelo código deixa de constar como ausente
Programas gerados saem das entradas de manutenibilidade	DBB: de 45 GO TO por mil linhas para 0
Testes exigem marcas explícitas	GenApp: 5 arquivos com nome de teste, que são telas da aplicação, deixam de contar

Achados confirmados no fonte foram mantidos, por exemplo um STARTBR do CICS sem tratamento de erro na demonstração de fornecedor, arquivos sequenciais sem FILE STATUS no Bank of Z e os comandos ALTER do CardDemo. Os nomes que continuaram ausentes foram conferidos como ausentes dos repositórios, como programas definidos nas definições CICS do CardDemo e um copybook que o benchmark copia mas não contém.

7.8 Limites do perfil

A calibração é provisória. O corpus tem nove repositórios e cerca de 107 mil linhas de código COBOL. Um contém código de produção publicado por uma administração pública; os demais são demonstrações de fornecedores, um serviço publicado, um benchmark sintético gerado com um modelo de linguagem e amostras escritas para ensinar. Código de demonstração pode conter defeitos de propósito e tende a ser mais limpo que um parque em operação com milhões de linhas. As faixas vêm de julgamento e da distribuição deste corpus; nenhum dado de resultado, como incidentes ou custo de migração, foi associado a elas.

A leitura é léxica. COPY REPLACING, pré-processadores e expansão de macros não são resolvidos; nomes montados em execução continuam dinâmicos, a menos que uma constante literal os identifique; Assembler e PL/I são inventariados sem análise. Testes são reconhecidos por arquivos do COBOL Check e por chamadas ao runtime do zUnit; não foi verificado se testes escritos para as ferramentas de teste mais novas da IBM mantêm essas chamadas. Os fontes precisam ser legíveis como texto compatível com ASCII, então membros em EBCDIC devem ser convertidos na exportação. Arquivos acima de 2 MB e checkouts acima de 20.000 arquivos são parcialmente ignorados.

O domínio de segurança tem dois sinais. O perfil será revisado, como versão 1.1, com o primeiro parque de cliente medido sob autorização, junto com o denominador de fontes ausentes descrito na seção 7.4.

8. Evidência de licenças fora da pontuação

8.1 Por que licenças ficam fora da pontuação

Uma obrigação de licença não é um defeito de código. Se um componente com copyleft forte ou copyleft de rede é aceitável depende de como o software é distribuído ou oferecido e da política da organização; o mesmo componente pode ser aceitável numa ferramenta interna e não num produto distribuído. Uma pontuação que se movesse com achados de licença faria o mesmo código ser lido de forma diferente em duas organizações, e o instrumento estaria respondendo por elas a uma questão jurídica.

O SQCM, portanto, registra a evidência de licenças e a avalia contra a política da própria organização, e nenhuma pontuação de domínio de nenhum dos perfis muda por causa de uma licença. O diagnóstico de código também lista dois avisos de baixa severidade, um para componentes com copyleft forte e outro para componentes cuja licença não pôde ser resolvida. Eles aparecem na lista de riscos e não têm peso em nenhum domínio.

8.2 Inventário

O inventário lê os lockfiles encontrados no repositório e lista cada dependência resolvida, direta ou transitiva, com sua licença declarada conforme informada pelo serviço deps.dev [44]. As versões vêm apenas dos lockfiles. Um manifesto sem lockfile é relatado como tal, porque suas versões, e portanto suas licenças, não estão fixadas. Os formatos suportados são os do scanner embarcado; CocoaPods não é suportado.

Quando o scanner não devolve licença para um pacote, a plataforma consulta o registro público do pacote (npm, PyPI e Maven Central quando este texto foi escrito) e, se necessário, o texto de licença distribuído com o pacote. As declarações são normalizadas para identificadores e expressões SPDX [45] nas grafias conhecidas e nas URLs canônicas de licença. Nomes ambíguos como "BSD" são mantidos como escritos, a menos que o texto de licença do próprio pacote os resolva, e referências do tipo "SEE LICENSE IN" viram identificadores LicenseRef próprios do documento. O que continua sem resolução fica visível como desconhecido e nunca é adivinhado.

Essa resolução envia os nomes e versões dos pacotes não resolvidos a esses registros públicos; nomes que um registro não conhece ficam em cache, para não serem enviados de novo a cada execução. O próprio scanner envia as coordenadas dos pacotes aos serviços deps.dev e OSV para obter licenças e advisories [32][44].

Arquivos carregados dentro do repositório são examinados de outras duas formas. Uma marca SPDX-License-Identifier ou uma declaração de licença reconhecível no cabeçalho de um arquivo versionado é registrada como aviso. Um arquivo idêntico a um arquivo de uma versão de pacote popular do npm ou do PyPI é reconhecido pelo seu resumo SHA-256 num índice mantido pela plataforma; arquivos presentes em mais de três pacotes e arquivos com menos de 512 bytes são ignorados, para que conteúdo trivial não gere correspondências.

Cada repositório guarda até 25.000 componentes ou 6 MB de inventário, com o corte sinalizado. A etapa de dependências, incluindo a resolução nos registros, tem quatro

minutos; quando o tempo acaba, a execução não registra inventário, em vez de registrar um parcial.

8.3 Artefatos construídos e código copiado

Um cliente de CI pode acrescentar artefatos construídos, como uma imagem descompactada ou um diretório de distribuição. Eles são analisados no modo de sistema de arquivos raiz do Trivy [46], que lê os metadados dos pacotes instalados, e entram no inventário como componentes de artefato, com as licenças que o scanner relata, sem normalização adicional. Suas vulnerabilidades conhecidas também entram no domínio Cadeia de Suprimentos (seção 2.4). Binários sem metadados de pacote, como código nativo ligado estaticamente ou firmware, não produzem lista de componentes, e nada é afirmado sobre eles.

A detecção de código copiado é opcional e vem desligada. Quando uma organização a liga, a plataforma executa o cliente da SCANOSS [47] com ofuscação de caminhos, de modo que ele envia impressões digitais dos arquivos, que são resumos do seu conteúdo, em vez do texto-fonte. O operador da base de conhecimento gratuita descreve identificação de trechos nesse serviço, e a identificação de trechos de alta precisão é oferecida nos planos comerciais [47]. Num teste de 2 de outubro de 2026 contra a base gratuita, um arquivo sem alterações de uma biblioteca muito usada foi identificado como correspondência de arquivo inteiro, e trechos de 120 e 200 linhas da mesma biblioteca inseridos em outros arquivos não foram identificados. Por isso este paper afirma apenas a identificação de arquivos inteiros na base gratuita. Uma organização também pode apontar o cliente para uma base de conhecimento comercial ou para um servidor SCANOSS próprio, que a plataforma aceita como endereço HTTPS próprio. As correspondências entram no inventário como componentes de trecho.

8.4 Política, exceções e verificação no pull request

Cada organização tem uma política, que atribui a cada licença uma decisão: permitida, em revisão ou proibida. As regras são definidas por família (permissiva, copyleft fraco, copyleft forte, copyleft de rede, outra e desconhecida) e podem ser sobrepostas por licença. As licenças são atribuídas às famílias pelo identificador. Esse agrupamento é um parâmetro da plataforma, não uma classificação jurídica; por exemplo, ele coloca licenças source-available e licenças Creative Commons não comerciais junto com o copyleft de rede. Numa expressão, OR adota a opção mais favorável e AND a menos favorável; várias licenças declaradas são tratadas como AND.

O modelo padrão permite licenças permissivas, manda para revisão copyleft fraco, copyleft forte, outras e desconhecidas, e proíbe copyleft de rede. Uma organização pode avaliar as dependências apenas de desenvolvimento com as mesmas regras ou sinalizá-las só quando proibidas. Pacotes sob os prefixos de nome internos da organização são permitidos. Só proprietários e administradores alteram a política, e cada alteração fica num histórico de revisões apenas acrescentado.

As decisões são recalculadas sempre que a evidência é lida. Uma mudança de política, portanto, vale para os inventários existentes sem nova análise. O resultado do pull request descrito abaixo é a exceção: ele fica registrado com sua execução.

Um proprietário ou administrador pode aceitar um componente em revisão ou proibido, para um pacote (uma versão ou todas as suas versões) ou para uma licença, na organização

inteira ou num projeto, com justificativa escrita e validade de no máximo 365 dias. Um componente aceito aparece como aceito, nunca como permitido, e volta à decisão da política quando a exceção vence.

Na medição contínua, uma execução que traz inventário é comparada com uma execução de referência: a última execução concluída da branch de destino do pull request, quando houver uma registrada, ou então a última execução concluída do projeto; sem execução anterior, todo componente conta como novo. Componentes proibidos sem exceção ativa que a referência não continha são relatados; componentes em revisão nunca são relatados por esta verificação. Uma mudança de versão conta como componente novo. O resultado padrão é um aviso. Uma organização pode optar por bloquear, e só então o cliente de CI falha o build. Sem inventário, nenhum resultado de licença é produzido. As execuções enviadas pelo cliente de CI ainda não registram sua branch, então a comparação com a branch de destino depende de execuções registradas para essa branch por outros caminhos; quando não há nenhuma, a alternativa pode comparar uma mudança com uma execução anterior da mesma mudança.

8.5 Saídas

Para cada execução, a plataforma exporta um documento SPDX 2.3 em JSON [45]; o SPDX 2.3 amplia a versão padronizada como ISO/IEC 5962:2021 [48]. O documento tem um pacote raiz por repositório e um pacote por componente, com sua expressão de licença declarada e sua URL de pacote. Licença concluída, texto de copyright e local de download são registrados como NOASSERTION, porque não são observados. Ela também exporta um texto de avisos de terceiros agrupado por licença, com cada texto de licença incluído uma vez; linhas de copyright não são inventadas quando os lockfiles não as trazem. A SBOM CycloneDX da seção 6 recebe as licenças dos componentes do mesmo inventário. Um documento gerado pelo código de produção a partir de um inventário de teste foi lido e convertido sem erros pelas ferramentas oficiais do SPDX (spdx-tools 0.8.5), e uma expressão de licença malformada de propósito foi rejeitada; isso verifica o formato, não a completude de qualquer inventário.

O dossiê de auditoria inclui, por projeto, o inventário julgado pela política vigente quando o dossiê é gerado, o documento SPDX e os avisos. No nível da organização, inclui a política com suas revisões no período e as exceções. Cada arquivo é listado com seu resumo SHA-256 no manifesto do dossiê.

8.6 Limites da evidência de licenças

A evidência relata licenças como declaradas por metadados de pacote, registros e cabeçalhos de arquivo. Ela não lê cada arquivo de licença de cada pacote, não estabelece a licença que um advogado concluiria e não determina se um componente é distribuído, ligado ou usado em execução. Não é aconselhamento jurídico. Os lockfiles limitam o que é visto: manifestos sem versão fixada, formatos não suportados e binários sem metadados ficam fora do inventário. Trechos copiados não são afirmados para a base de conhecimento gratuita, e os avisos vêm só de cabeçalhos de arquivo.

A evidência de licenças pode apoiar um programa de conformidade de licenças, por exemplo um organizado segundo a ISO/IEC 5230 [49]. Produzi-la não estabelece conformidade com essa norma nem com qualquer licença.

9. Validação e resultados

9.1 Auditoria de implementação de setembro de 2026

A revisão da v2.1 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 confirmou o snapshot de implantação auditado; não constituiu 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 auditados e os resultados das fixtures estão registrados na nota suplementar de validação da v2.1.

9.2 Acréscimos de outubro de 2026

O perfil Mainframe e a evidência de licenças foram examinados da mesma forma que na seção 9.1: contratos da interface, chamadas de serviços, persistência e testes de regressão direcionados. O código foi lido nas revisões de produção dos dois serviços que os implementam em 2 de outubro de 2026 (ai-governance `9c00c8187d77`, insights `4a81e4055953`), e os testes abaixo foram executados nessas revisões com a checagem de tipos desligada no executor de testes.

Grupo de validação	Suítes	Testes aprovados
Leitor mainframe e pontuação do perfil	2	25
Nível mainframe com medição contínua e versões de pontuação	1	3
Inventário de licenças, resolução em registros, normalização, avisos, índice de arquivos e análise de código copiado	8	62
Política de licenças, exceções, documentos exportados e verificação no pull request	4	37

Grupo de validação	Suítes	Testes aprovados
Total	15	127

A calibração da seção 7.6 foi reproduzida em 2 de outubro de 2026, com a leitura dos nove repositórios nas revisões registradas pelo leitor de produção. Cada pontuação de domínio, pontuação e nível coincidiu com os valores registrados em 30 de setembro de 2026. No mesmo dia, o teste de código copiado da seção 8.3 e a verificação de formato SPDX da seção 8.5 foram executados com os clientes e as ferramentas indicados nelas. Como na seção 9.1, são verificações de regressão escritas e executadas pelo fornecedor, não uma amostra de validação independente.

Código-fonte interno, coeficientes exatos do perfil padrão e catálogos proprietários de regras não são disponibilizados com este paper. O suplemento oferece rastreabilidade, mas não reprodução independente irrestrita do instrumento proprietário completo.

9.3 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 nem para a v3.0. 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.

9.4 Calibração histórica de detecção

A publicação da v2.0 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 nas edições posteriores. 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.

10. Limitações e agenda de pesquisa

A questão em aberto mais importante é a validade preditiva. Nem as faixas nem os pesos dos domínios de nenhum dos perfis 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. O perfil Mainframe acrescenta os limites da seção 7.8, e a evidência de licenças, os da seção 8.6.

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. Para o perfil Mainframe, acrescenta a medição de parques de clientes sob autorização, um leitor de PL/I, mais sinais de segurança e a validação das conexões com gerenciadores de bibliotecas. São objetivos de pesquisa e implementação, não capacidades estabelecidas por esta publicação.

11. 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	10.5281/zenodo.22781322
v3.0	Perfis de pontuação e SQCM Mainframe 1.0; evidência de licenças fora da pontuação	Esta publicação

O conceito do perfil padrão, a calibração v2.2 e as correções registradas na v2.1 não mudam. A contribuição desta revisão é uma segunda régua para sistemas que a primeira não foi feita para ler, e um limite documentado para a evidência que informa decisões sem entrar numa pontuação.

Referências

A bibliografia da v2.1 é mantida como itens 1 a 34, e os itens a partir do 35 são novos nesta edição. 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

[35] Gartner. "Gartner Predicts More Than 70% of Mainframe Exit Projects Will Fail Due to Overestimation of Generative AI's Capabilities". Press release, Stamford, June 18, 2026. Citado como contexto do setor; a previsão trata de projetos de saída iniciados em 2026 e não é testada aqui. https://www.gartner.com/en/newsroom/press-releases/2026-06-18-gartner-predicts-more-than-70-percent-of-mainframe-exit-projects-will-fail-due-to-overestimation-of-generative-ais-capabilities

[36] IBM. *Enterprise COBOL for z/OS 6.5 Language Reference*, "Reference format". https://www.ibm.com/docs/en/cobol-zos/6.5.0?topic=structure-reference-format

[37] IBM. *CICS Transaction Server for z/OS 6.x*, "Defining resources with DFHCSDUP" and "The DFHCSDUP DEFINE command". https://www.ibm.com/docs/en/cics-ts/6.x?topic=resources-defining-dfhcsdup · https://www.ibm.com/docs/en/cics-ts/6.x?topic=commands-dfhcsdup-define-command

[38] IBM. *CICS Transaction Server for z/OS 6.x*, "BMS macro DFHMSD" and "BMS macro DFHMDI". https://www.ibm.com/docs/en/SSJL4D_6.x/reference-applications/bms/dfhp47j.html · https://www.ibm.com/docs/en/SSGMCP_6.2.0/reference-applications/bms/dfhp47g.html

[39] Open Mainframe Project. *COBOL Check*. Apache License 2.0; o repositório está arquivado e o projeto está em status Emeritus. https://github.com/openmainframeproject/cobol-check

[40] IBM. *IBM Developer for z/OS 14.2*, "ZUnit overview". https://www.ibm.com/docs/en/SSQ2R2_14.2.0/com.ibm.etools.rdz.zunit.doc/topics/c_zunit_overview.html

[41] ISO/IEC JTC 1/SC 22. *ISO/IEC FCD 1989:2001, Information technology - Programming languages - COBOL.* Final Committee Draft, document N3204, January 15, 2001, Annex D.1. https://www.open-std.org/jtc1/sc22/open/n3204.pdf · IBM. *Enterprise COBOL for z/OS 6.5 Language Reference*, "ALTER statement". https://www.ibm.com/docs/en/cobol-zos/6.5.0?topic=statements-alter-statement

[42] Dijkstra, E. W. "Letters to the editor: go to statement considered harmful". *Communications of the ACM* 11(3), 1968, pp. 147-148. DOI: 10.1145/362929.362947. https://doi.org/10.1145/362929.362947

[43] Corpus de calibração, lido nas revisões listadas na seção 7.6 apenas para medição; nenhum fonte é redistribuído com este paper. Licença conforme declarada em cada repositório: aws-samples/aws-mainframe-modernization-carddemo (Apache-2.0), cicsdev/cics-genapp (EPL-2.0), IBM/Bank-of-Z (Apache-2.0), etalab/taxe-fonciere (CeCILL 2.1), walmartlabs/zECS (Apache-2.0), IBM/dbb (Apache-2.0), IBM/zOS-Client-Web-Enablement-Toolkit (Apache-2.0); uma demonstração de fornecedor sob licença restrita a demonstrações e um benchmark sintético publicado sem arquivo de licença, que não são nomeados. https://github.com/aws-samples/aws-mainframe-modernization-carddemo · https://github.com/cicsdev/cics-genapp · https://github.com/IBM/Bank-of-Z · https://github.com/etalab/taxe-fonciere · https://github.com/walmartlabs/zECS · https://github.com/IBM/dbb · https://github.com/IBM/zOS-Client-Web-Enablement-Toolkit

[44] Google. *OSV-Scanner* documentation, "License scanning". https://google.github.io/osv-scanner/usage/license-scanning/ · Open Source Insights. *deps.dev API v3.* https://docs.deps.dev/api/v3/

[45] Linux Foundation and its Contributors. *The Software Package Data Exchange (SPDX) Specification, Version 2.3.* 2022. https://spdx.github.io/spdx-spec/v2.3/

[46] Aqua Security. *Trivy* documentation, "Rootfs" and "License scanning". https://trivy.dev/docs/latest/guide/target/rootfs/ · https://trivy.dev/docs/latest/guide/scanner/license/

[47] SCANOSS. *scanoss.py*, MIT License. https://github.com/scanoss/scanoss.py · SCANOSS documentation, "Platform Architecture Overview". https://docs.scanoss.com/en/latest/introduction/platform-architecture-overview · Software Transparency Foundation. *OSSKB.* https://www.softwaretransparency.org/osskb

[48] ISO/IEC 5962:2021. *Information technology - SPDX® Specification V2.2.1.* International Organization for Standardization, 2021. https://www.iso.org/standard/81870.html

[49] ISO/IEC 5230:2020. *Information technology - OpenChain Specification.* International Organization for Standardization, 2020. https://www.iso.org/standard/81039.html · https://openchainproject.org/license-compliance

---

ScaleQuality Code Maturity Model v3.0 · © 2026 Erik Fernandes Amaral · Licença CC BY 4.0. Os pesos internos de pontuação do perfil padrão, as heurísticas de detecção e a especificação de taint continuam sendo propriedade da ScaleQuality. Os pesos e as faixas do perfil Mainframe são publicados na seção 7 porque também aparecem no produto. Este documento descreve a metodologia em nível conceitual.