Our Methodology

Learn how the Sigrid® engine works



Software Improvement Group

Our Methodology at a glance

Section	Brief explainer	More detail and links
Our Benchmark	Your code is scored against 40,000+ real systems, 600 billion+ lines of code, and 300+ technologies, recalibrated every year.	A yearly-recalibrated benchmark spanning old and new technologies alike, weighted toward the languages most common today, so your score reflects the current state of the market, not a fixed or outdated norm. Read more
Compliant with- and co-editor of-leading global standards.	Backed by ISO 25010, ISO 17025, ISO 27001, SOC2, and independent lab accreditation, not self-reported claims.	Covers system and software quality models (ISO 25010), an accredited testing laboratory (ISO 17025), information security management (ISO 27001), data handling controls (SOC2/ISAE 3000), product certification (ISO Guide 65), and standards contributions via CISQ and NEN. Read more here
Architectural context	Gives your AI agents the architectural context to build better software and prevent architecture drift	Prevent architecture drift before it merges: a new call across a directory boundary, a facade bypassed on the way to a database, a dependency cycle the change would close. The architecture-drift skill grounds that check in Sigrid's measured dependency graph, so the verdict comes from how your system is actually wired today and not from the few files the agent happened to open. Read more here
Star ratings	A 1–5 star scale shows where you sit against the market, and it predicts real differences in speed and cost.	Scores follow a 5-30-30-30-5 distribution, with 3 stars representing market average. The scale isn't just descriptive: 4-star systems cost half as much to maintain and ship up to four times faster than 2-star systems. Read more here
Architecture standards	How ready your system is to evolve as the business changes, not just how it runs today.	Our model assesses five sub-characteristics: structure, communication, data access, evolution, and knowledge, scored from nine measurable properties in the source code and version history, aggregated into a 1–5 star rating. Read more here
Maintainability standards	How easily your code can be changed, fixed, or extended without breaking things.	A 1–5 star model grounded in source-code facts, measuring how much of a team's time goes toward keeping existing systems running versus building new things. Higher ratings correlate with lower risk and cost, faster delivery, stronger security, and more innovation capacity. Read more here and here
Security standards	How well your code holds up against the OWASP Top 10, the industry's standard list of critical risks.	Static analysis of source code and infrastructure, scored 1–5 stars and mapped against the OWASP Top 10. Complements (not replaces) penetration testing, SCA, and DAST, each layer catches what the others miss. Read more here
Open source standards	How exposed your software is through the third-party libraries it depends on.	Rates the risk carried by third-party open-source dependencies across five areas: known vulnerabilities, dependency freshness, license risk, community activity, and dependency management, not just whether a library has a known flaw. Read more here

Table of contents

OUR METHODOLOGY AT A GLANCE 2

TABLE OF CONTENTS 3

CHAPTER 1. OUR APPROACH 4

The benchmark 4

The star-system explained 4

Trusted, renowned, certified 6

CHAPTER 2. OUR FOUR QUALITY MODELS 7

Maintainability 7

Security 8

Open source 9

Architecture 10

One engine, two products 10

CHAPTER 3. HOW DOES SIGRID DETECT AI GENERATED CODE? 11

GOVERN THE SOFTWARE YOUR BUSINESS RUNS ON 12

Chapter 1. Our approach

Finding an objective measure of software quality is tough.

High-quality software behaves the way you expect it to, can withstand unforeseen situations, is secure against abuse, and complies with requirements.
What all these have in common is the need for maintenance. This is the honorable task of developers and the roles surrounding them. Efficient, effective, workable code, easy to read even. That is, easy for the developers (and agents) that come after you.

Essentially, *bad code is bad for business*. The inverse is not necessarily always true, but it is certain that you need to have a grip on quality to deliver business value in a predictable and efficient way.

The benchmark

Generally, to assess the quality of something, you need a sense of context. Each piece of software may have a widely different context. To put objective thresholds on code quality, at SIG we use technology-independent code measurements and compare those to a benchmark. A benchmark on quality is meaningful because it provides an unbiased norm of how well you are doing. The context for this benchmark is "the current state of the software development market". This means that you can compare your source code to the code that others are maintaining.

To compare different programming technologies with each other, the metrics represent a type of abstractions that occur universally, like the volume of pieces of code and the complexity of decision paths within. In this way, system size can be normalized to "person-months" or "person-years", which indicate of amount of developer work done per time period.

Those numbers are again based on benchmarks.
Our Sigrid engine compares analysis results for your system against a benchmark of 40,000+ industry systems, spanning 600 billion+ lines of code and 300+ technologies. This benchmark set is selected and calibrated (rebalanced) yearly to keep up with the current state of software development. "Balanced" here can be understood as a representative spread of the "system population". This includes anything in between old and new technologies, from anything legacy to modern JavaScript frameworks. In terms of technologies this is skewed towards programming languages that are now most common, because that best represents this current state. The metrics underlying the benchmark approach a normal distribution. This offers a sanity check of being a fair representation and allows statistical inferences on "the population" of software systems.
This comparison runs across all four quality models — maintainability, security, open source health, and architecture quality — so every score you see is benchmarked, not just the maintainability rating.

The star-system explained

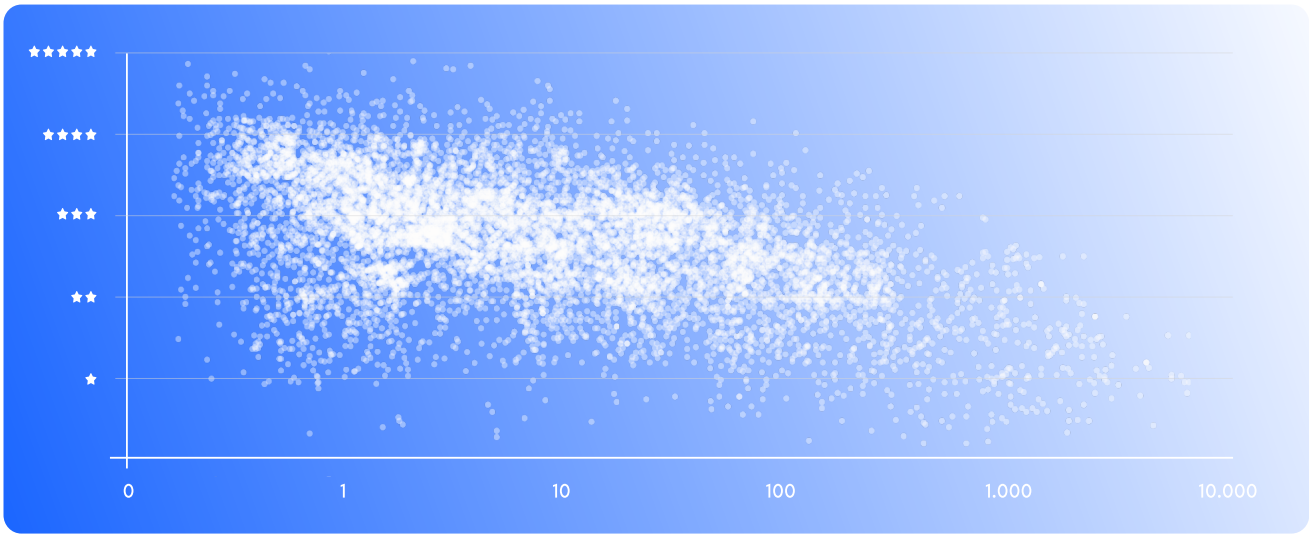
The code quality score compared to this benchmark is expressed in a star rating on a scale from 1 to 5 stars. It follows a 5%-30%-30%-30%-5% distribution. Technically, its metrics range from 0.5 to 5.5 stars. This is a matter of convention, but it also avoids a "0" rating score because 0 is not a meaningful end on a quality scale. The middle 30% exists between 2.5 and 3.5, with all scores within this range rated as 3 stars, representing the market average.

Even though 50% of systems necessarily score below average (3.0), 35% of systems will score below the 3-star threshold (below 2.5), and 35% will score above the 3-star threshold (above 3.5). To avoid a suggestion of extreme precision, it is

helpful to think about these stars as ranges, such that 3.4 star would be considered "within the expected range of market average, on the higher end". Note that calculation rounding tolerances are always downwards, with a maximum of 2 decimals of precision. So, a score of 1.49 stars will be rounded down to 1 star.

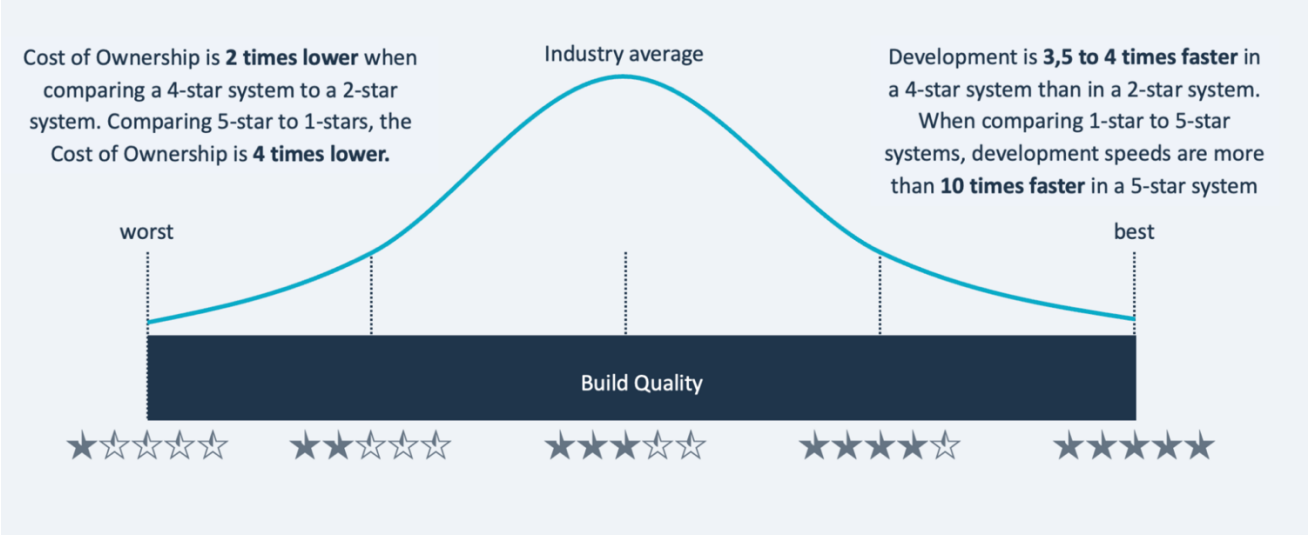
The system cloud

One of the findings that the SIG maintainability benchmark consistently confirms is that system volume correlates negatively with maintainability. This generally applies to all sectors and technologies: large systems tend to be harder to maintain than smaller systems. This holds even when we account for SIG model bias, where a higher system volume scores lower on maintainability because of e.g. increasing analysis/coordination efforts. From our experience we know that large systems can still score very well on a "code unit level" (the smallest unit of programming logic), but complexity increases quickly on an architectual level. As an illustration, a simplified visualization of what is commonly known as the "SIG cloud" is shown below. The X-axis is a logarithmic scale of system volume (in person-years) and the Y-axis represents the maintainability scale from 0.5 to 5.5 stars.



Star ratings are predictive of speed and costs

Based on experience with its datasets of systems, SIG has found a strong correlation and cause-effect relationship between star ratings and development speed/system maintenance costs. For example, maintenance/ownership costs for 4-star systems are two times lower than for 2-star systems, and their development speed is up to four times faster.
See the graphs below:



Trusted, renowned, certified

Software Improvement Group (SIG) operates the world's first and only laboratory accredited under ISO/IEC 17025 for software quality analysis, certified in collaboration with TÜViT.

Our lab meets the strict international standards for testing and calibration laboratories. We use standardized, repeatable measurements of source code based on globally recognized ISO/IEC standards, including ISO/IEC 25010 for software quality, ISO/IEC 5055 for code quality, and ISO/IEC 27001 for information security management. In addition, we co-developed ISO/IEC 5338, the new global standard for AI lifecycle management, and actively collaborate with NEN on developing and refining ISO and NPR guidance for software and systems engineering, as well as AI and big data standards.

Our controls are also independently attested under SOC2 (ISAE 3000) Type I and Type II, and our Software Product Certification service complies with ISO/IEC Guide 65 for organizations that offer certification services. As a member of CISQ's Governing Board, SIG also helps set the direction for automated software quality measurement standards.

Our quality models provide concrete, measurable guidance for improving software quality and security.

Chapter 2. Our four quality models

Maintainability

Maintainability is the ease with which a system can be changed, improved, or fixed. Just like a well-built house is easier to renovate, a well-structured codebase supports safe, efficient development over time.

How do we measure maintainability

At Software Improvement Group, we measure maintainability using a 1–5 star model grounded in sourcecode facts and validated across industries. Having a high maintainability rating has proven to lower risk and cost, speed up delivery, make your systems more secure, and increase innovation capacity.

Maintainability scores predict how much of a development team's time goes toward keeping existing systems working versus building something new. Systems below the threshold lock in engineering capacity that organizations need elsewhere. At a time when most senior leaders are trying to scale AI, that locked capacity is the hidden cost.

Resources

Maintainability Guidance for Producers

A practical companion to the Trusted Product Maintainability criteria, helping software producers understand the measurements and what to improve to reach 4-star certification.

Maintainability Evaluation Criteria

Reference document outlining the maintainability model, measurement areas, and thresholds used to evaluate and certify software products against the Trusted Product Maintainability standard.

Security

The same capabilities that help defenders move faster are available to attackers too, and the gap between what organizations can discover and what they can control is growing. This is the environment Software Improvement Group (SIG) is measuring against in 2026. And what our portfolio data continues to show is that the organizations weathering this environment best are not the ones with the most sophisticated security tooling. They are the ones with the clearest picture of what they are running, the strongest software foundations, and the discipline to treat security as a continuous practice rather than a periodic checkpoint.

Layers of defense: why one method is never enough

Most organizations understand, in principle, that security requires multiple layers.

- Penetration testing: finds what attackers would find.
- Static Application Security Testing (SAST): analyzes the source code to detect weaknesses before deployment.
- Software Composition Analysis (SCA): scans third-party open-source libraries and dependencies for known vulnerabilities. (See open source health model)
- Dynamic Application Security Testing (DAST): automatically scans your live system for weaknesses.

Each method covers ground the others miss. So, no single method is enough on its own. SAST, DAST and SCA are complemented by penetration testing to form a complete security assessment. Together, these techniques enable earlier detection, stronger compliance, and more secure software from the start. The reality is that many developers continue to release code that lacks proper input sanitization and access control. The OWASP Top 10, the industry's standard reference for critical web application risks, has seen broken access control and injection flaws at the top of the list for years. Attackers know this pattern as well, and they exploit it systematically.

How do we measure security?

Sigrid®'s SAST (Static Application Security Testing) security model ranks software systems from 1 to 5 stars. Sigrid® evaluates system properties through automated static analysis of the source code and infrastructure, deriving scores for various security characteristics. These scores are then mapped to the OWASP Top 10, the ten most critical risks in web application security. The star rating reflects a system's compliance against that benchmark: 1. Severely low degree of security controls, 2. Very low degree of security controls, 3. Low degree of security controls, 4. Moderate degree of security controls, 5. High degree of security controls. A 4- or 5-star rating does not guarantee full security. It indicates that security considerations have been factored into the design and implementation, making vulnerabilities less likely.

Resources

Application security model
Guidance for software producers on how SIG evaluates system security based on ISO 25010 and a star rating from one to five for implemented security controls.

OWASP top-10

We analyze system properties through in-depth reviews of source code, infrastructure, and supporting artifacts such as documentation. From these analyses, we derive scores for key system characteristics and map them to the OWASP Top 10, highlighting the most critical web application security risks and guiding targeted risk reduction over time.

CWE → CVSS benchmarking

We have built a benchmarking system that maps CWEs to CVSS scores using historical CVE data from the last five years. This effectively creates expert-judgment benchmarks, using real-world vulnerability information to prioritize and score potential weaknesses more accurately before they develop into exploitable security issues in production systems for clients.

NVD for CVSS scoring

We use the National Vulnerability Database (NVD) as our authoritative source for CVSS scoring. The NVD provides standardized severity ratings from 0.0 to 10.0, along with rich metadata, helping us interpret vulnerabilities consistently and prioritize remediation work based on their potential impact on systems, applications, and supporting infrastructure for clients

Open source

Open source software (OSS) is source code developed and maintained through open collaboration. Anyone can use, examine, alter, and redistribute OSS as they see fit, typically at no cost. Understandably so, it's (and has been) very popular: Our data shows that the average enterprise organization relies on 132 open-source libraries.

How do we measure open-source?

SIG's Open Source Health quality model rates the risk a system carries from its third-party open-source dependencies, benchmarked on a 1-to-5 star scale across five areas: vulnerabilities, dependency freshness, license risk, community activity, and dependency management.

Resources

Open Source Health Guidance for Producers

Guidance for software producers on SIG's Open Source Health Quality Model, explaining how open source dependencies are measured and what is needed to reach a 4-star risk profile for vulnerabilities, freshness, licenses, activity, and management.

CWE → CVSS benchmarking

We have built a benchmarking system that maps CWEs to CVSS scores using historical CVE data from the last five years. This effectively creates expert-judgment benchmarks, using real-world vulnerability information to prioritize and score potential weaknesses more accurately before they develop into exploitable security issues in production systems for clients.

NVD for CVSS scoring

We use the National Vulnerability Database (NVD) as our authoritative source for CVSS scoring. The NVD provides standardized severity ratings from 0.0 to 10.0, along with rich metadata, helping us interpret vulnerabilities consistently

and prioritize remediation work based on their potential impact on systems, applications, and supporting infrastructure for clients

Architecture

Software architecture is the structural design of a software system and refers to the way a given codebase is divided and grouped together based on responsibilities of the system, and the underlying dependencies between these responsibilities.

How do we measure Architecture?

The SIG Architecture Quality model measures how readily a software system can evolve as business needs change. It assesses five architecture sub-characteristics: Structure, Communication, Data Access, Evolution, and Knowledge, each reflecting how strongly that aspect affects the speed at which foundational changes can be made. SIG scores these from nine system properties measured in the source code and version-control history: code breakdown, component coupling, component adjacency, component cohesion, communication centralization, data coupling, bounded evolution, knowledge distribution, and component freshness. The property scores are then aggregated into an overall architecture rating from 1 to 5 stars.

Resources

Architecture Quality Guidance for Producers

Guidance for software producers on SIG's Architecture Quality Model, explaining how architecture flexibility is measured using code-based metrics, sub-characteristics, and 4-star thresholds.

One engine, two products

These same standards run inside Guardrails in real time, checking every change against your architecture, security, and maintainability rules as agents write code, and fixing what it finds before it ever lands in a commit. Auto-Fix Agents applies that same standard to what's already there. It's one engine, one standard, wherever code gets written.

Resources

Sigrid Axis documentation (coming Oct 1)
Installation (including an upgrade section from the Sigrid AI Toolkit), configuration, Guardrails, Auto-fix Agents, a skills reference, an MCP tools reference, and four guides.

Sigrid Core documentation
Introduction, Roles & Usage, How to set up, portfolio-level capabilities, system-level capabilities, External integrations and Sigrid CI pipeline integration guidelines.

Chapter 3. How does Sigrid detect AI generated code?

Sigrid uses a machine learning classification model that is trained on a set of stylometric features of AI generated and human-written code. Examples of stylometric features are line length, number of blank lines, indentation, casing, and similar source code characteristics. Sigrid takes into account over 90 different stylometric features and is able to identify AI generated code with an accuracy of 95–99%.

The model is self-trained and runs entirely on code metrics derived from your own source code. Sigrid does not send your code to a public AI service such as ChatGPT.

Detection happens at the level of units, meaning methods and functions. Units smaller than 5 lines of code are not evaluated, since they don't carry enough signal for reliable classification.

Sigrid detects AI-generated code in Java, Python, and C# today, trained on output from leading large language models and coding assistants such as Claude Code, Cursor, and GitHub Copilot. For the current list of supported languages, models, and coding assistants, see Technology Support and the AI code governance FAQ on docs.sigrid-says.com. Support for new models happens quarterly. Sigrid also runs over 300 checks to identify AI technologies in use across a codebase more broadly.

At 95–99% accuracy, roughly 19 out of 20 code fragments are classified correctly, which means a large codebase will still turn up a number of false positives. That's expected, not a flaw: the goal of AI-generated code detection isn't perfect accuracy on every snippet, it's giving you a reliable, high-level view of AI adoption across your portfolio, accurate enough to act on.

Govern the software your business runs on

Sigrid® gives you control over your entire software portfolio in the age of AI. Deterministic rules, powered by the largest software quality benchmark, the highest standards, and your full architectural context.

Many tools will tell you what "good" looks like. Only Sigrid can prove it. Sigrid assesses your code against deterministic rules, calibrated on 40,000+ real systems, in the context of your own architecture. It's the engine behind each Sigrid product: Core across your portfolio, Axis inside your coding agents.

One engine. Two ways to use it.

Axis and Core run on the same deterministic engine, the same benchmark, the same standard. That's what sets Sigrid apart from another AI opinion. Axis brings that standard into the agentic SDLC. Core is the control center for your entire software portfolio.

01
Deterministic, globally-recognized standards

Quality standards refined for 25+ years in the world's only ISO/IEC 17025-accredited software quality lab — not a prompt, not another model's opinion

02
Benchmarked against 40,000+ real systems
"Good" isn't abstract. It's what the largest software benchmark in the world says good looks like for a system like yours.

03
Architecture-aware: catches problems before merge
Every check understands what your system is meant to look like. It catches the drift and erosion that may be harmless in isolation but wrong in your codebase.

Ready to govern your software?

Book a demo and we'll walk through what continuous governance looks like for your portfolio.
Contact us