January 7, 2026 (Version 2, March 5, 2026)

Whitepaper

# AI-Ready Code: How Code Health Determines AI Performance

A Whitepaper on how AI fails, and why Code Health is the missing foundation of AI-assisted development

By Adam Tornhill

**Code health determines whether AI accelerates delivery or amplifies defects.** Large-scale studies show that AI-generated changes fail significantly more often in unhealthy code, with **defect risk rising by at least 60%**. In the AI era, healthy code is no longer optional.

CodeScene

# Executive summary

AI coding assistants promise faster delivery, but their performance depends heavily on code quality.

Large-scale studies show that AI-generated changes break significantly more often in unhealthy code, with **defect risk increasing by at least 60%** even within relatively maintainable systems.

**+60%**

**higher defect risk**

when AI works on unhealthy code

Because existing research excludes truly low-quality code, **real-world risk is likely much higher** and non-linear.

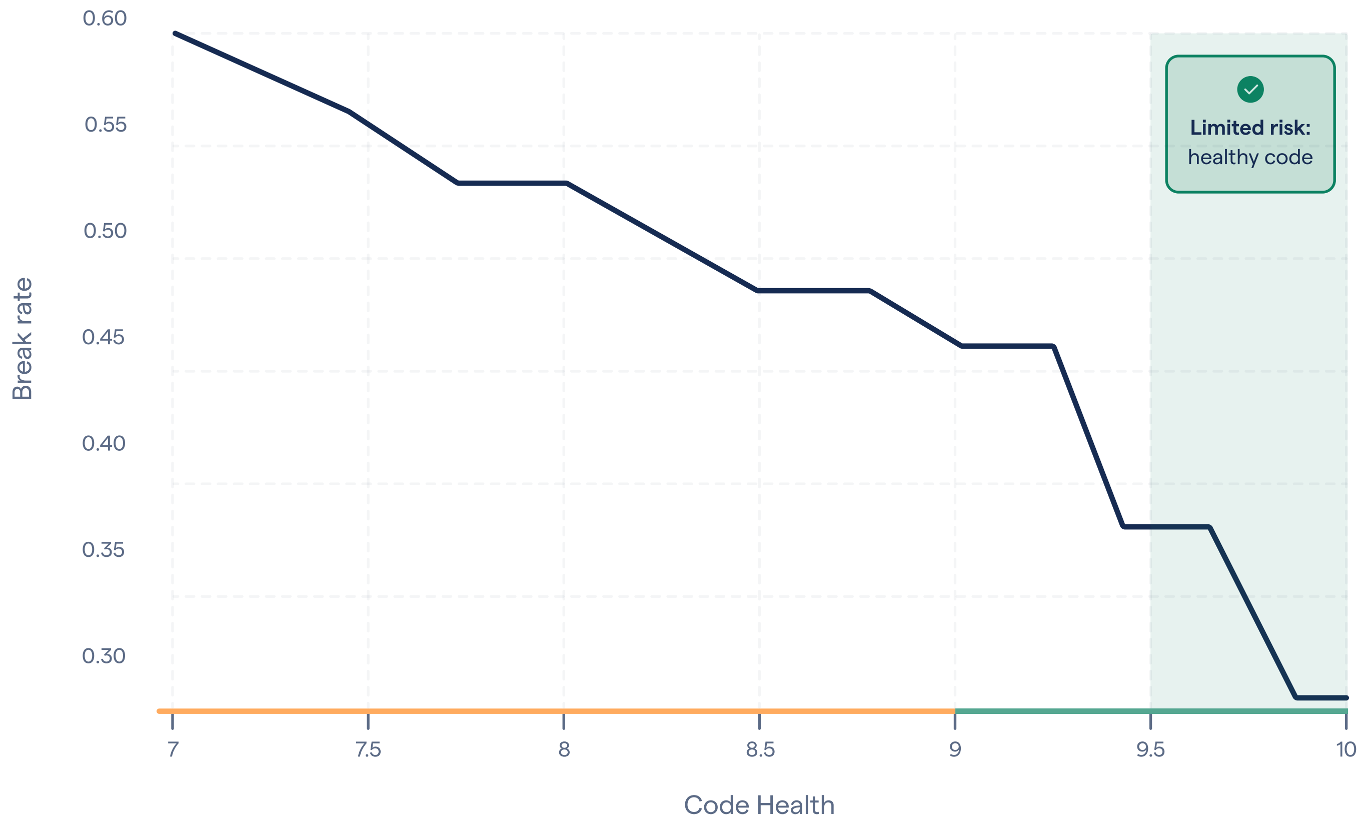
In the AI era, healthy code is no longer optional. It is a **prerequisite for safe, effective, and economically viable AI adoption**. This paper explains how to assess AI-readiness using Code Health indicators and how to uplift unhealthy code so AI can act as a reliable engineering partner.

# AI is now a software maintenance actor

For decades, the maxim has been that "programs must be written for people to read, and only incidentally for machines to execute" (SICP). Human-readable code is essential for maintaining secure, reliable, and efficient software development.

> But with the advent of AI coding, **textual source code now has a broader audience: machines need to understand it, too**. AI agents increasingly participate in **code modification**, **refactoring**, and **generation**.

In Code Red: The Business Impact of Code Quality, we showed that poor code quality correlates with **higher defect rates**, **slower delivery**, and **ballooning costs**. Now we're seeing the next evolution of that insight: **unhealthy code also makes AI behave badly**.



## AI-friendliness starts with Code Health

The study Code for Machines, Not Just Humans defines "AI-friendliness" as the probability that AI-generated refactorings preserve behavior and improve maintainability. It's a large-scale study of 5,000 real programs using six different LLMs to refactor code while keeping all tests passing. The results were stark:

LLMs consistently **perform better in healthy code**.

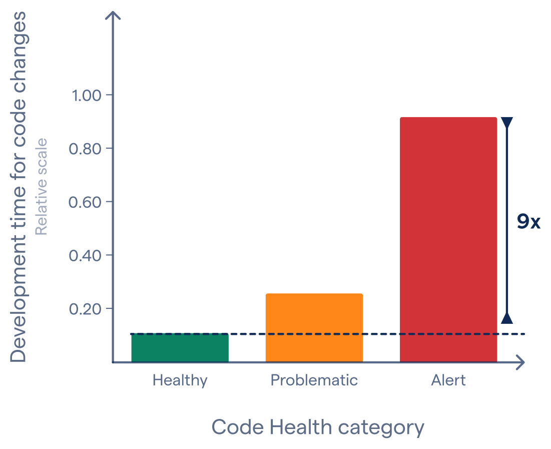
Unhealthy code triggered **a higher rate of AI-introduced defects** across the board.

There's a **60% higher defect risk** when applying AI to problematic code.

The study used the CodeHealth™ metric as a proxy for code quality. Code Health is known to correlate with lower defect density and increased speed. Code Health has also been used in recent industry-facing AI studies.

See the Code Health overview for more details, the multi-dimensional AI benchmarking for industry examples, and the paper on ACE: Automated Technical Debt Remediation for an applied AI study.

Development is **9x faster in healthy code**



### CodeHealth™ - the leading quality metric

CodeScene's CodeHealth™ is the only code-level metric with proven business impact (faster & better), backed by award winning, fact-based research.

CodeHealth™ is a ten point scale with three categories:

- **Green** (healthy code): **9.0+**
- **Yellow** (technical debt): **4.0 - 8.9**
- **Red** (severe technical debt): **1.0 - 3.9**

## The hidden limitation: Reality is worse. Much worse.

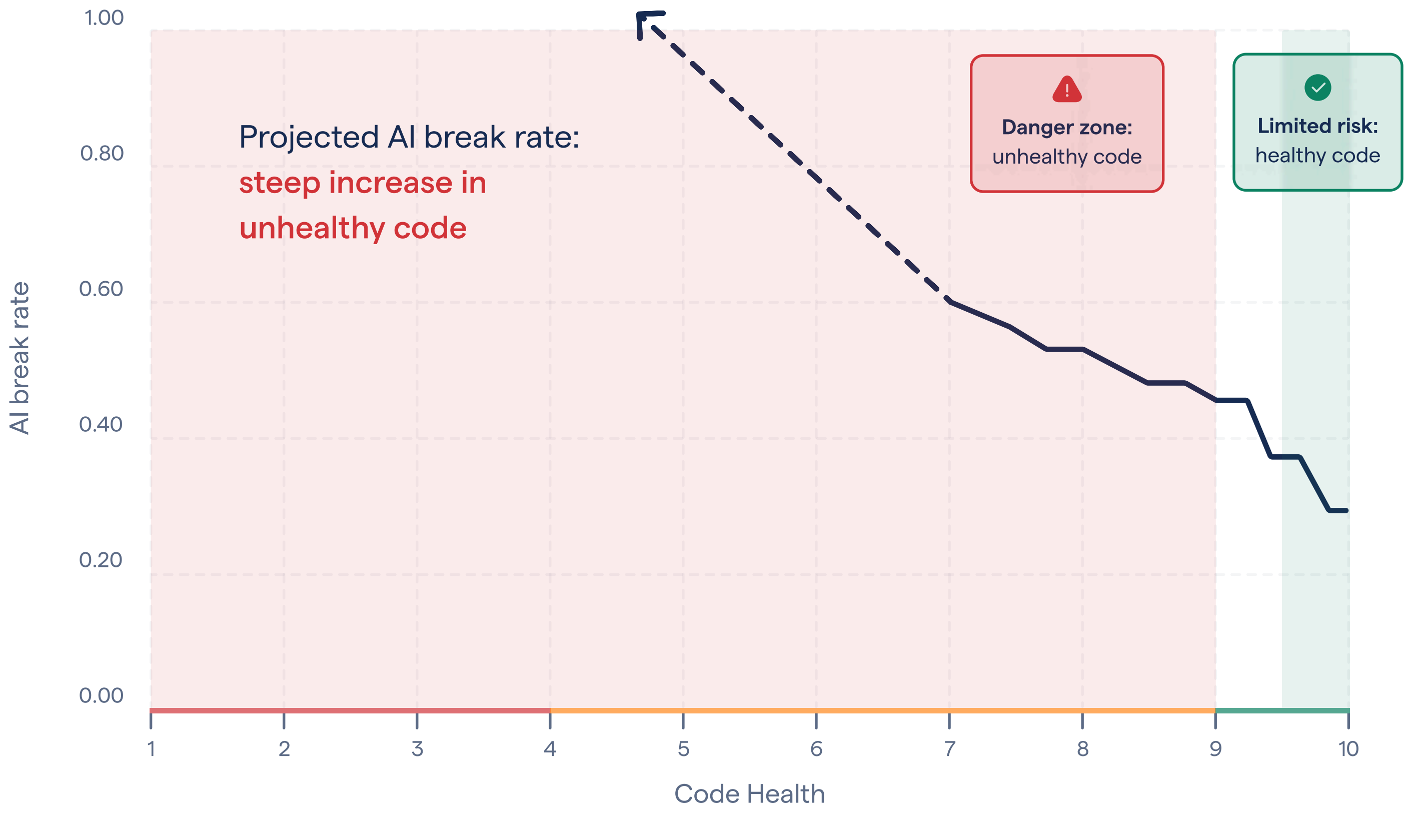
A 60% increase in AI-induced defects is severe. But the study only included code with **Code Health ≥ 7.0**. This means the research never touched the truly unhealthy code found in many legacy codebases: the modules scoring **4, 3, or even 1**.

**What would the AI error rate be on such code?** Based on patterns observed across all Code Health research, the relationship is almost certainly **non-linear**:

At Code Health **7**, AI breaks code frequently

At Code Health **3**, breakage may become **the default outcome**

This mirrors human data: **defect rates rise sharply in tangled, deeply unhealthy code**.



Projected AI break rate for truly unhealthy code below 7.0 in Code Health; the break rate is likely to increase steeply. The projection is based on the non-linearity of Code Health outcomes observed in both **Code Red** and the **Increasing, Not Diminishing Returns** papers.

This gap highlights a critical risk: Organizations with **very low-quality codebases** may experience **dramatically higher AI breakage rates than the study could measure.**

## The business implication of AI: Technical debt now has a multiplier

Naively used, AI agents will serve more as legacy code generators than genuine help. There's now clear evidence to make any AI adoption a serious concern:

**More bugs**

AI adoption leads to **41% more defects** without any increase in throughput.

**AI makes you slower**

This is a fascinating one: developers self-estimated that their AI reduced completion time by 20%, whereas in reality **they needed 19% longer** (!) than a control group of devs without AI.

**Initial AI velocity gains are cancelled out**

Novel research demonstrates how the initial promising velocity gains from AI adoption are **fully cancelled out** after just two months. The reason? A massive increase in code complexity.

So where is the good news? Well, there is hope. As the research shows, Code Health acts as a protective buffer. Healthy code **reduces error-generation risk** and **gives AI the structural clarity** it needs to act more predictably.

The CodeHealth™ metric plays the key role. It's the objective standard that turns AI from a fast code generator into **a quality-aware engineering partner**.

The first step is to use Code Health for **assessing AI-readiness**. That way, you avoid risks by putting AI to work only where the probability of success works in your favour.

Healthy and highly maintainable code comes with other benefits, too. Not only does it elevate AI performance, healthy code also **shortens development times** and **reduces defects up to 15x**. As a concrete example, the statistical model shows that improving Code Health from the industry average of 5.15 to the elite level of 9.1 gives the following benefits:

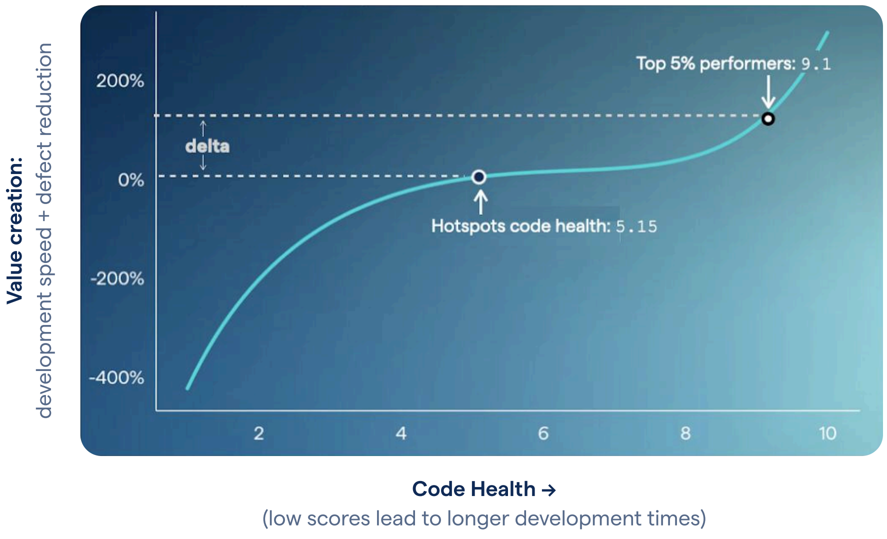
**~36% faster**

development speed

**~36% reduction**

in production defects

**This implies that you can have it all:** AI-friendly code means **increased speed** with **quality**.



The relationship between Code Health and business outcomes (faster & better) is **non-linear**; value creation accelerates in highly maintainable code (Data from M. Borg, et. al. 2024) **"Increasing, not Diminishing: Investigating the Returns of Highly Maintainable Code")**.

**⚠️ Caution: AI break rate is never zero**

Maintaining and enforcing (e.g. via **code health aware MCP** servers) strong code quality protects the shape and health of your codebase, but behavior still needs checks. That's where a strong test suite, good code coverage, and human review comes in; an AI cannot be trusted to put code into production.

With any technical debt removed, those checks are far easier to perform. Developers review healthy code more than **twice as fast** as unhealthy code, which dramatically lowers the overhead of keeping AI safe.

# Conclusion: Code for machines, not only humans

Machines get confused by the same patterns as humans. The evidence is clear: unhealthy code undermines AI-assisted development, increasing breakage rates and reducing the benefits of automation.

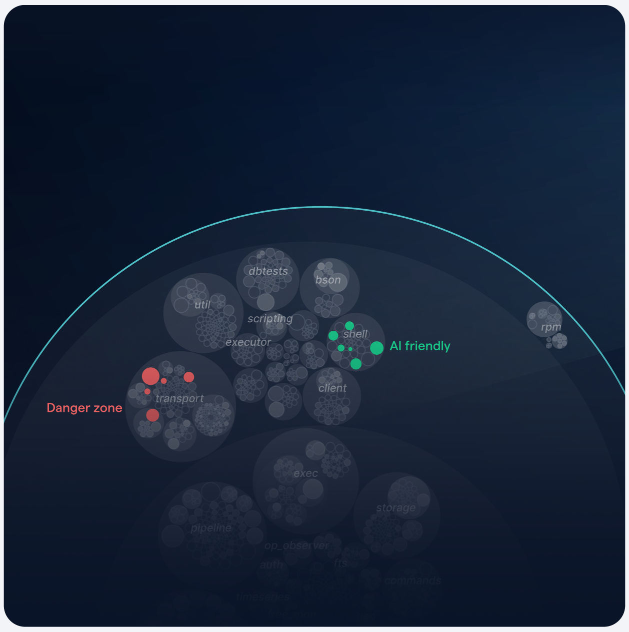
Organizations that want safe, reliable, and effective AI-assisted development must invest in Code Health as a foundational capability. Without it, AI will not accelerate delivery; it will accelerate defects and developer frustration.

## Safeguard healthy code, uplift unhealthy code

You have seen the research on AI readiness and why code health is essential for AI-assisted development. The CodeScene MCP Server turns those insights into action by exposing CodeScene’s Code Health analysis as local, AI-friendly tools.

The result is three high-impact use cases:

- Safeguard AI-Generated code, automatically
- Uplift unhealthy code: refactoring safely with CodeScene ACE + AI
- Understand existing code before acting



**Code Health visualisation**

Example on areas with red, unhealthy code where an AI agent will have a high break rate.

Read more and try out the MCP.

View the docs