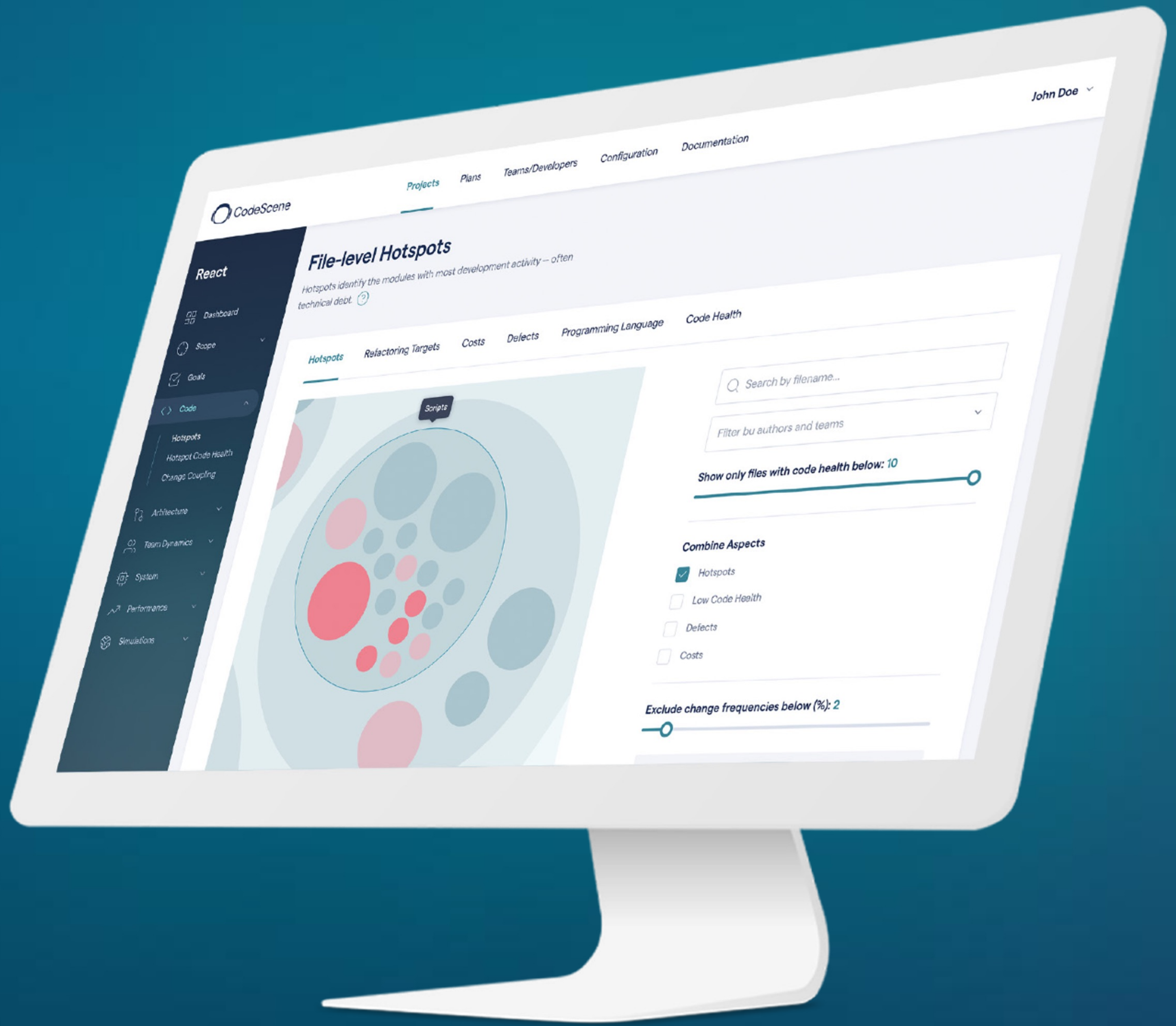
CodeScene

Scale AI coding safely without sacrificing quality

We give AI agents deterministic code-quality guidance, so they improve velocity without creating technical debt.



File-level Hotspots

Hotspots identify the modules with most development activity – often technical debt.

Show only files with code health below: 10

Combine Aspects

☑ Hotspots

☐ Low Code Health

☐ Defects

☐ Costs

Exclude change frequencies below (%): 2

© 2026 CodeScene AB, all rights reserved.

AI writes tomorrow’s legacy code, faster than teams can review it

40% more bugs

Uplevel

Velocity gains cancelled out due to **massive increase in complexity**

CARNEGIE MELLON UNIVERSITY PITTSBURGH PENNSYLVANIA 1900

Technical debt accelerator (75% tech leaders will face high or moderate technical debt by 2026)

FORRESTER®

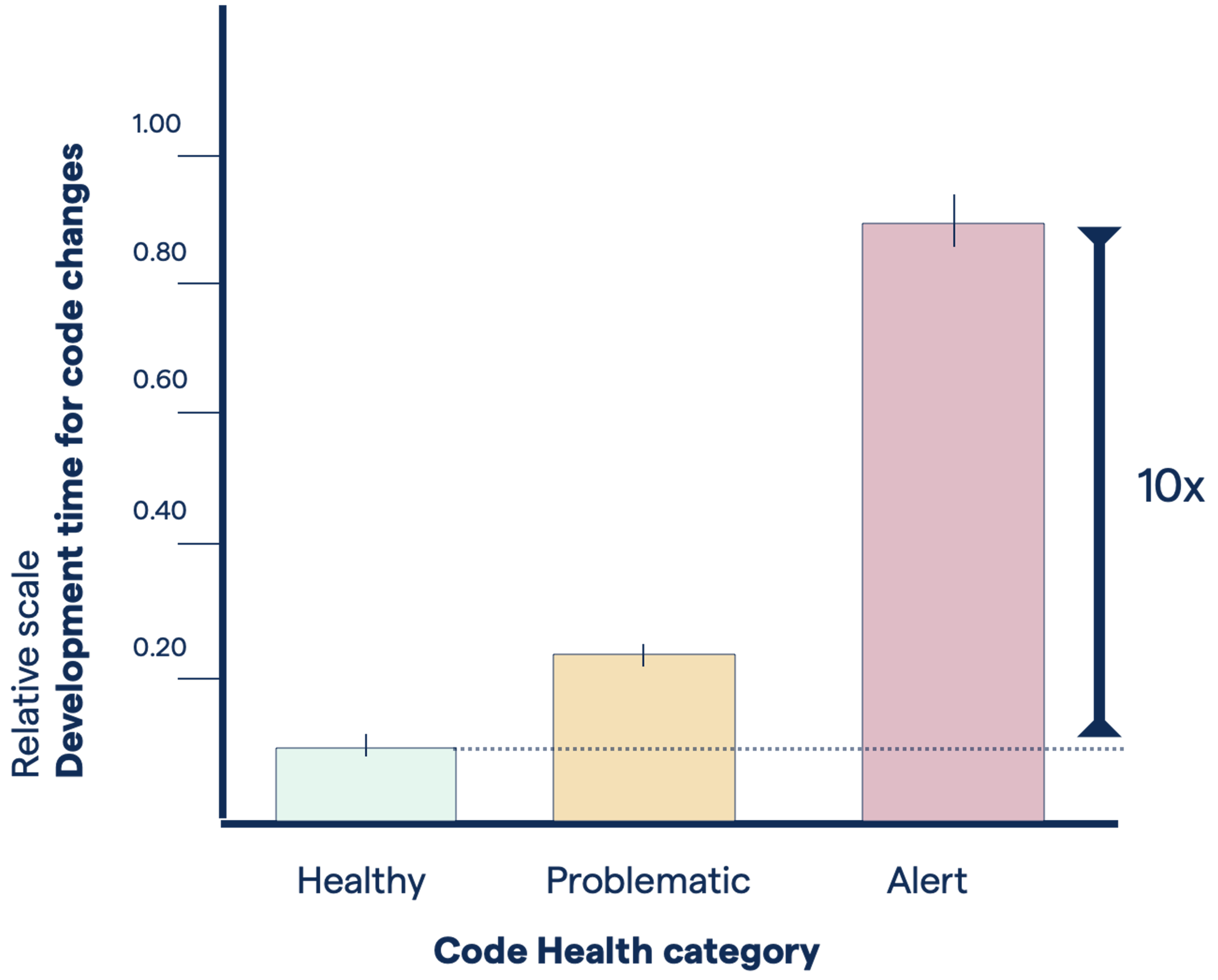
AI amplifies defects in unhealthy code **(+30% defect risk)**

CodeScene

© 2026 CodeScene AB, all rights reserved.

Code Health is the foundation – showing what ”good” and ”bad” code is

Task completions times in unhealthy code are up to 10x longer compared to green, healthy code

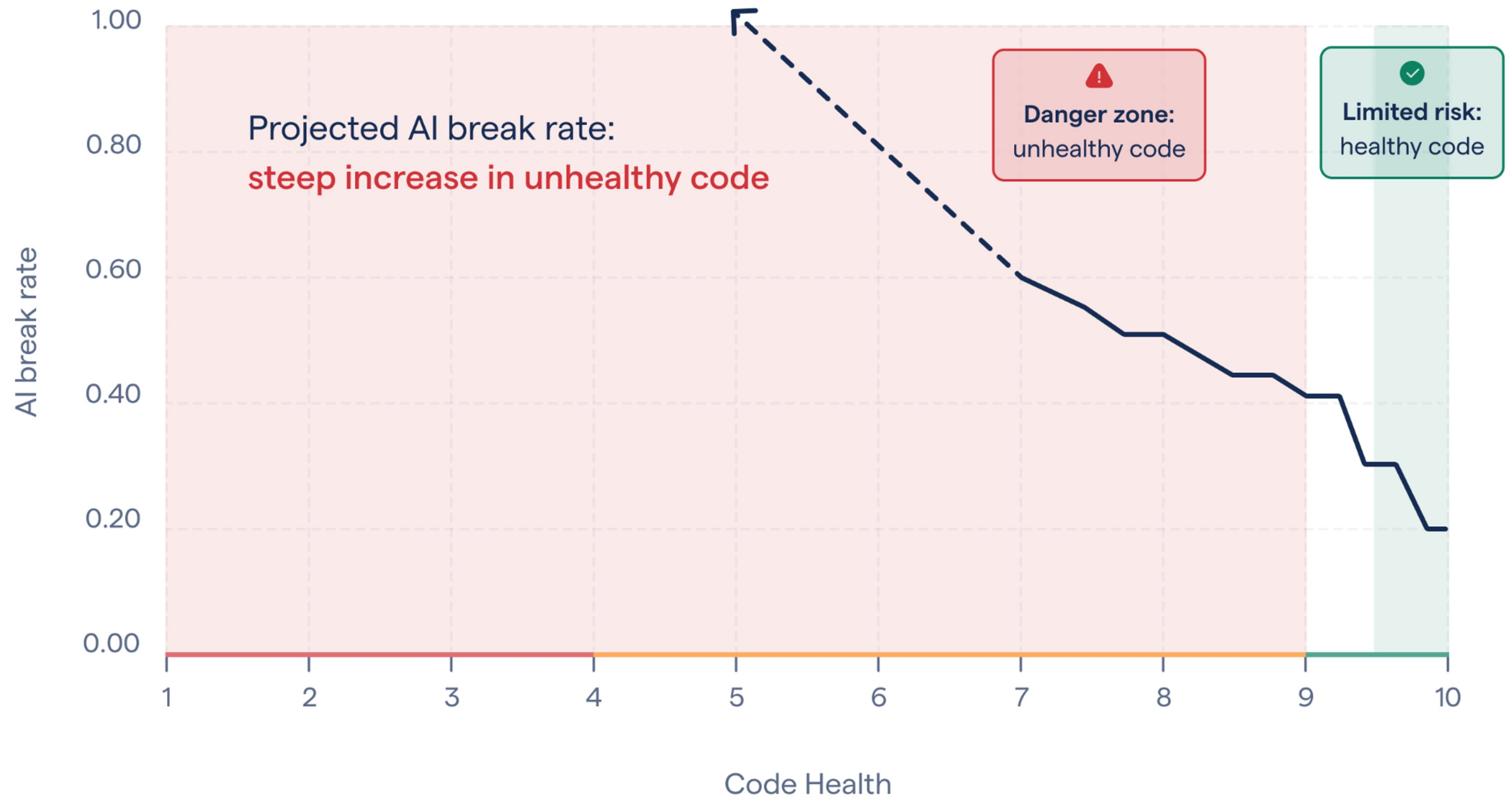


CodeHealth™ the leading quality metric

- Research-based, award-winning metric
- Low code quality correlates to higher defect rates, slower delivery and higher costs.
- Unhealthy code makes AI behave badly, and never before is more code produced than now
- Benchmark, objective and deterministic measure
- Protective buffer for AI-assisted development

© 2026 CodeScene AB, all rights reserved.

CodeHealth™ determines AI-performance



- New research, **Code for Machines, Not Just Humans**
- Large-scale study, 5000 real programs, 6 different LLMs, refactor code while keeping all tests passing
- LLMs perform better in **healthy code**
- **30% higher defect risk** when applying AI to **problematic code**
- Reality is worse, much worse

“In the AI era, healthy code is no longer optional. It’s a prerequisite for safe, effective, and economically viable AI adoption.”

Adam Tornhill, Founder and CTO of CodeScene

© 2026 CodeScene AB, all rights reserved.

A powerful automated AI framework:

a self-correcting, agentic workflow that makes AI-assisted development safe, predictable and measurable

CodeHealth™ metric

Validated code quality metric demonstrating proven business impact on delivery performance and defect risk

MCP server

Guides AI agents in real time so they produce maintainable, low-risk code

CodeScene ACE

Automates safe refactorings that help make risky code ready for AI acceleration

© 2026 CodeScene AB, all rights reserved.

Solving three core problems

Risk assessment & strategic view

CodeHealth™ analysis shows where AI-coding can be **safely applied** today.

AI safeguards for AI-ready areas

A code-health aware MCP server creates **a continuous feedback loop** for AI agents, enabling deterministic, real-time quality checks and **preventing AI from introducing technical debt**.

AI-powered uplift for not-yet-ready areas

Since AI works best on healthy code, using **CodeScene ACE** via **the MCP server**, teams can **automatically refactor problematic code**. Improvements are validated via **objective CodeHealth™ metric**.

Benchmarks shows 2-5x improvement over frontier model,
9x time savings on refactoring.

© 2026 CodeScene AB, all rights reserved.

Safe AI acceleration for modern engineering teams

Predictable and safe AI adoption

Leaders gain visibility, predictability, and measurable ROI, clear business KPIs around Code Health & delivery

Cleaner, safer AI-generated code

Developers get more reliable AI output and fewer defects

Code quality guidance for AI-agents

AI agents follow CodeHealth™ guidance for simple, clean, easy to evolve code

Faster AI acceleration

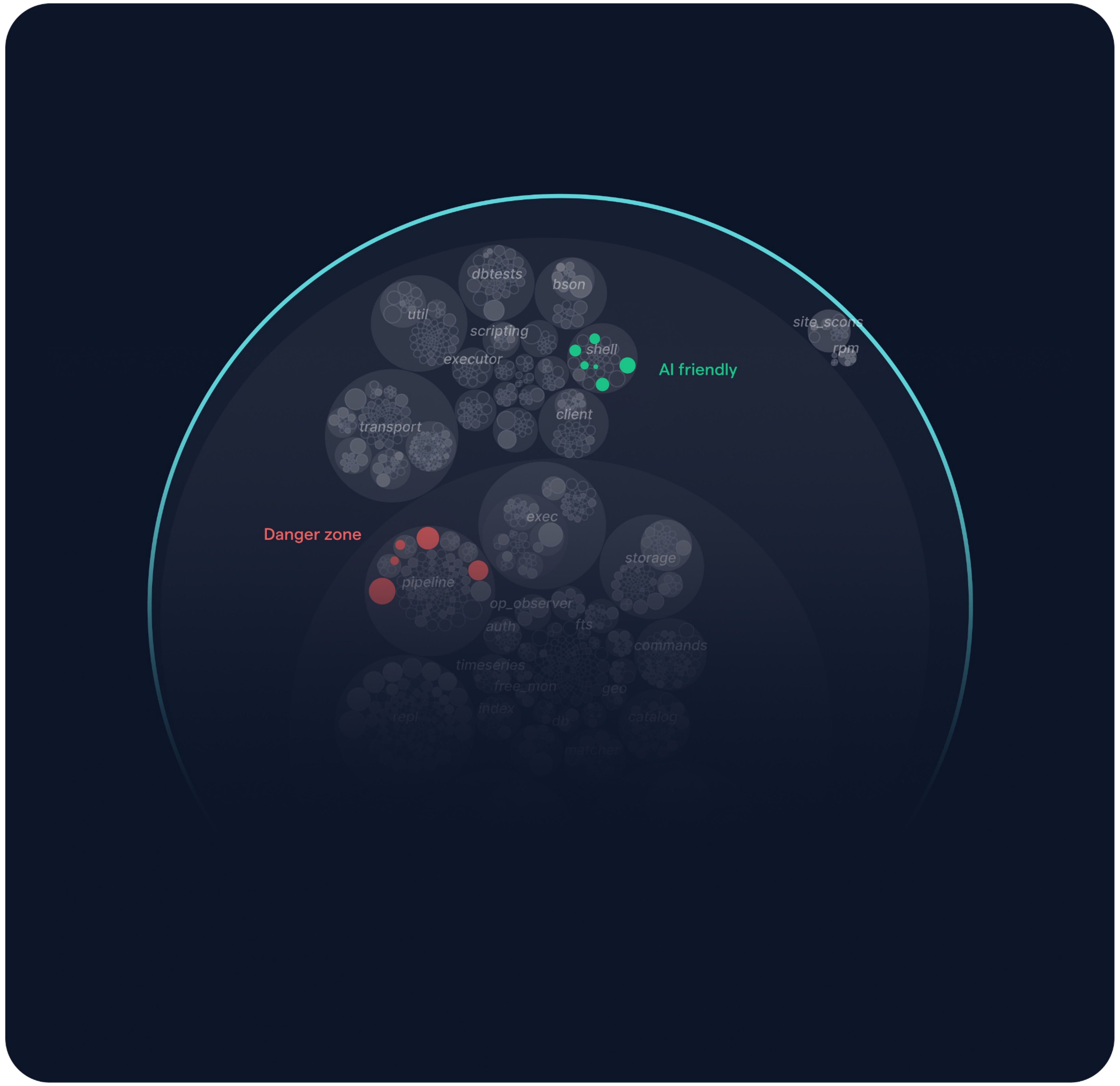
Teams modernize high risk areas first and expand AI readiness faster

© 2026 CodeScene AB, all rights reserved.

Making sure AI-coding assistants only work on healthy code enables higher ROI

CodeHealth™ visualisation

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



© 2026 CodeScene AB, all rights reserved.

How it works: Enabling AI-friendly code

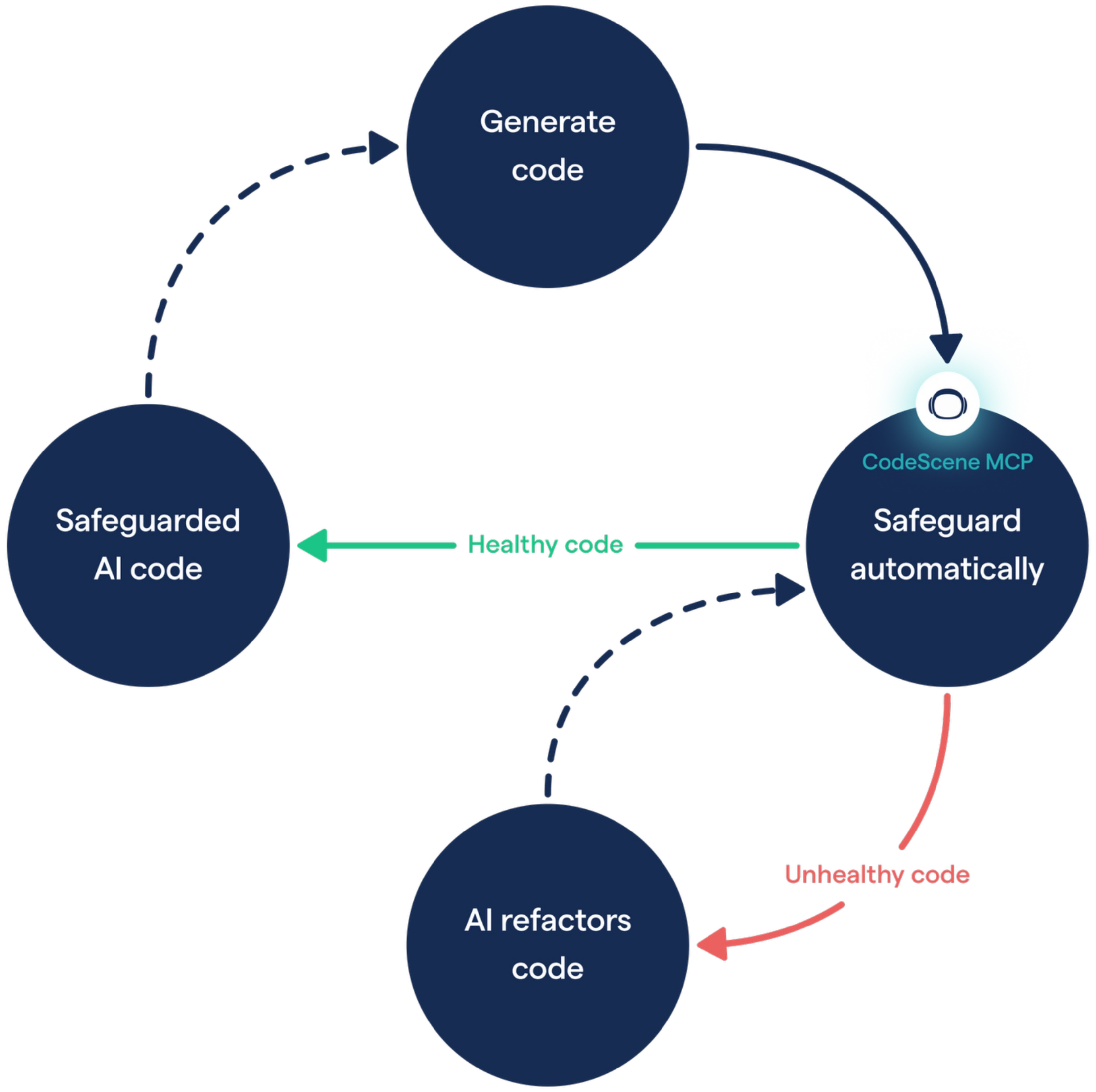
1 AI generates code, guided by CodeHealth™ via **MCP server**

2 **Code evaluated** before it becomes your problem

3 The agent refactors based on **objective metrics**

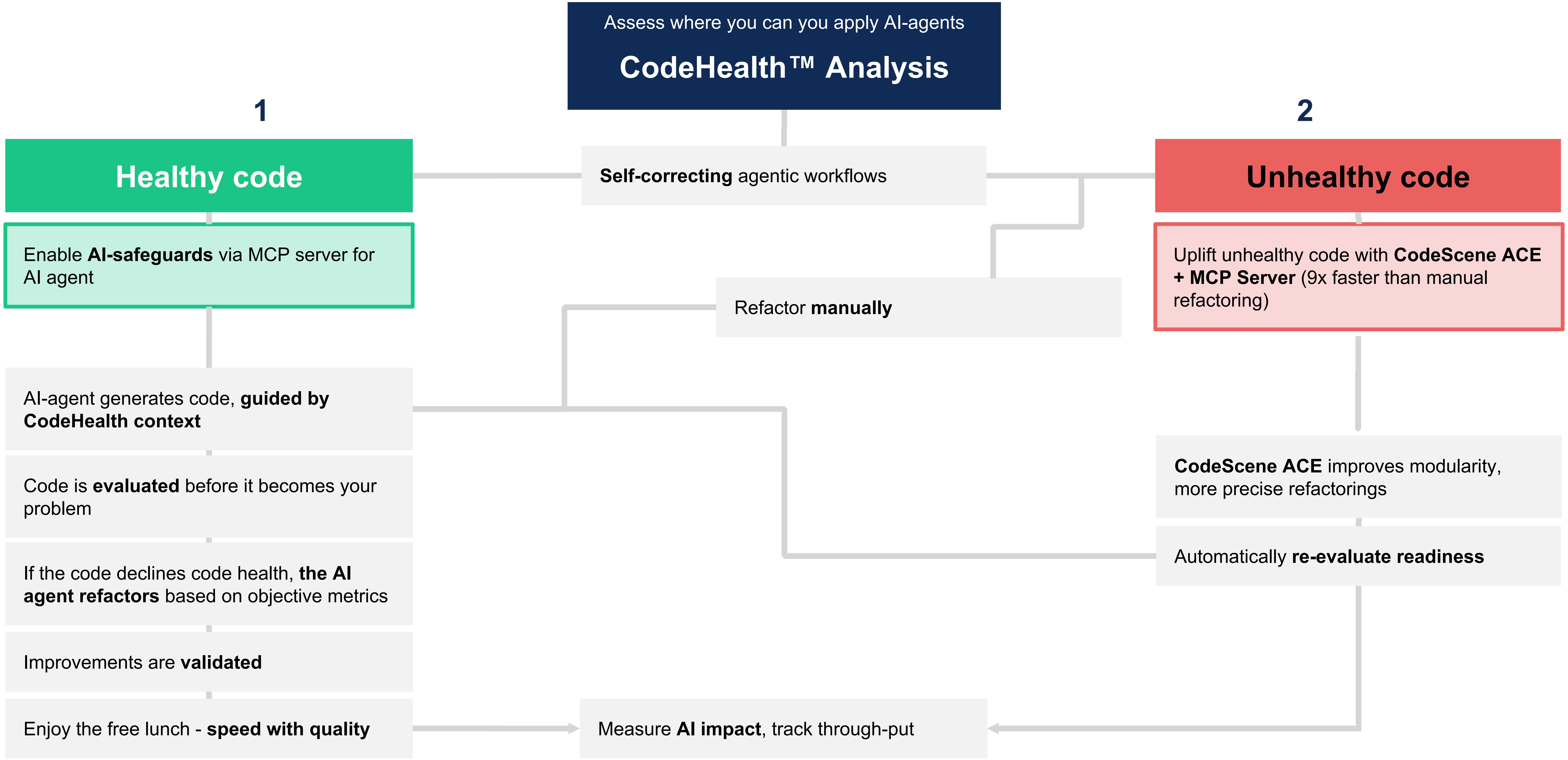
4 Improvements are **validated**

5 Enjoy the free lunch – **speed with quality**



© 2026 CodeScene AB, all rights reserved.

How do you get started?



Assess where you can you apply AI-agents

CodeHealth™ Analysis

1

Healthy code

Enable **AI-safeguards** via MCP server for AI agent

AI-agent generates code, **guided by CodeHealth context**

Code is **evaluated** before it becomes your problem

If the code declines code health, **the AI agent refactors** based on objective metrics

Improvements are **validated**

Enjoy the free lunch - **speed with quality**

Self-correcting agentic workflows

Refactor **manually**

Measure **AI impact**, track through-put

2

Unhealthy code

Uplift unhealthy code with **CodeScene ACE + MCP Server** (9x faster than manual refactoring)

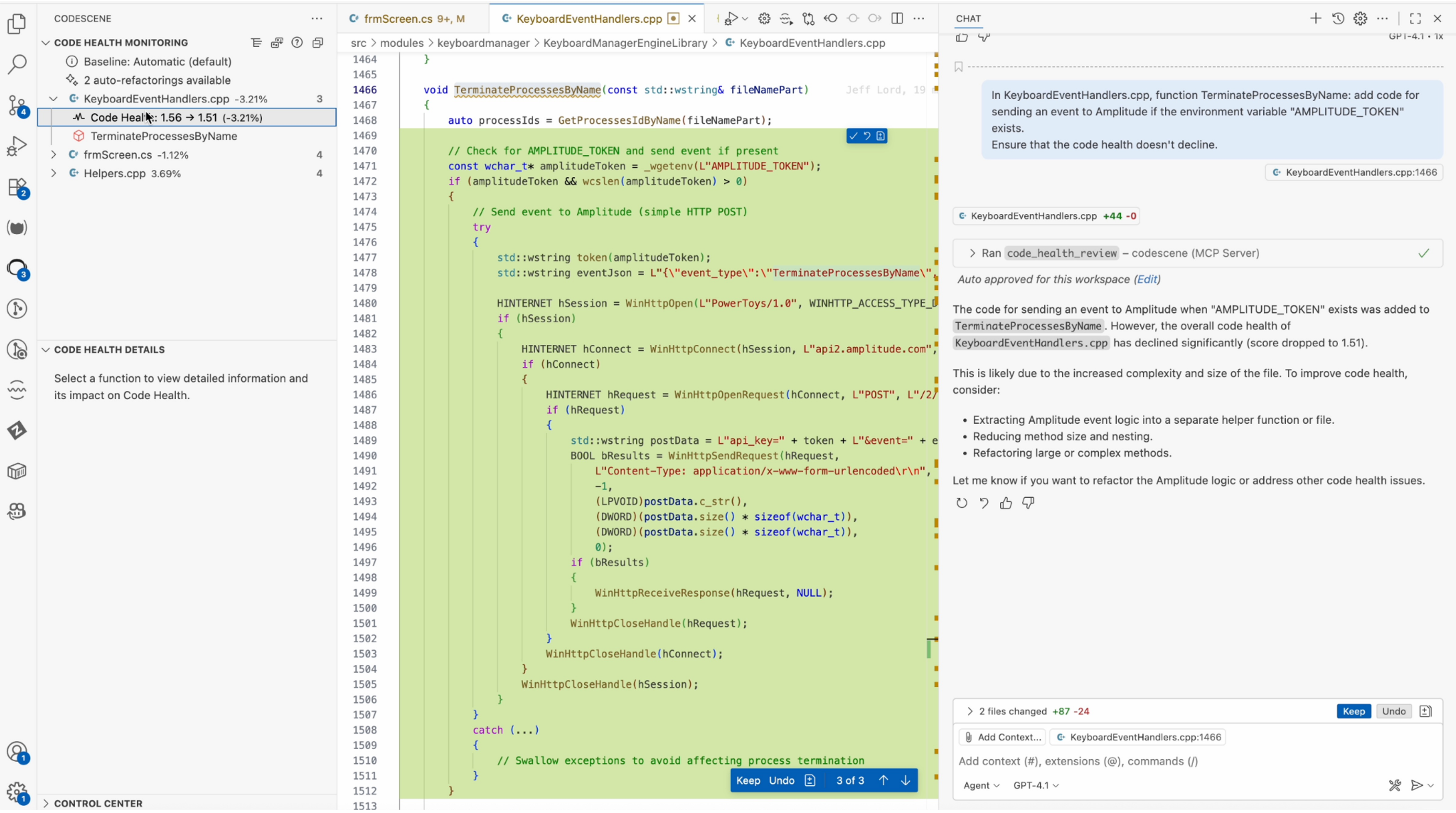
CodeScene ACE improves modularity, more precise refactorings

Automatically **re-evaluate readiness**

© 2026 CodeScene AB, all rights reserved.

Mission — improve AI performance

MCP: auto-inject Code Health insights into the AI


CODESCENE
CODE HEALTH MONITORING
Baseline: Automatic (default)
2 auto-refactorings available
KeyboardEventHandlers.cpp -3.21% 3
Code Health: 1.56 → 1.51 (-3.21%)
TerminateProcessesByName
frmScreen.cs -1.12% 4
Helpers.cpp 3.69% 4
CODE HEALTH DETAILS

Select a function to view detailed information and its impact on Code Health.

CONTROL CENTER
frmScreen.cs 9+, M
KeyboardEventHandlers.cpp
src > modules > keyboardmanager > KeyboardManagerEngineLibrary > KeyboardEventHandlers.cpp
Jeff Lord, 19

```
}

void TerminateProcessesByName(const std::wstring& fileNamePart)
{
    auto processIds = GetProcessesIdByName(fileNamePart);

    // Check for AMPLITUDE_TOKEN and send event if present
    const wchar_t* amplitudeToken = _wgetenv(L"AMPLITUDE_TOKEN");
    if (amplitudeToken && wcslen(amplitudeToken) > 0)
    {
        // Send event to Amplitude (simple HTTP POST)
        try
        {
            std::wstring token(amplitudeToken);
            std::wstring eventJson = L"{\"event_type\":\"TerminateProcessesByName\",

            HINTERNET hSession = WinHttpOpen(L"PowerToys/1.0", WINHTTP_ACCESS_TYPE_D
            if (hSession)
            {
                HINTERNET hConnect = WinHttpConnect(hSession, L"api2.amplitude.com",
                if (hConnect)
                {
                    HINTERNET hRequest = WinHttpOpenRequest(hConnect, L"POST", L"/2/
                    if (hRequest)
                    {
                        std::wstring postData = L"api_key=" + token + L"&event=" + e
                        BOOL bResults = WinHttpSendRequest(hRequest,
                            L"Content-Type: application/x-www-form-urlencoded\r\n",
                            -1,
                            (LPVOID)postData.c_str(),
                            (DWORD)(postData.size() * sizeof(wchar_t)),
                            (DWORD)(postData.size() * sizeof(wchar_t)),
                            0);
                        if (bResults)
                        {
                            WinHttpReceiveResponse(hRequest, NULL);
                        }
                        WinHttpCloseHandle(hRequest);
                    }
                    WinHttpCloseHandle(hConnect);
                }
                WinHttpCloseHandle(hSession);
            }
        }
        catch (...)
        {
            // Swallow exceptions to avoid affecting process termination
        }
}
```

In KeyboardEventHandlers.cpp, function TerminateProcessesByName: add code for sending an event to Amplitude if the environment variable "AMPLITUDE_TOKEN" exists.
Ensure that the code health doesn't decline.

KeyboardEventHandlers.cpp:1466
KeyboardEventHandlers.cpp +44 -0

Ran `code_health_review` – codescene (MCP Server) ✓

Auto approved for this workspace (Edit)

The code for sending an event to Amplitude when "AMPLITUDE_TOKEN" exists was added to `TerminateProcessesByName`. However, the overall code health of `KeyboardEventHandlers.cpp` has declined significantly (score dropped to 1.51).

This is likely due to the increased complexity and size of the file. To improve code health, consider:

- Extracting Amplitude event logic into a separate helper function or file.
- Reducing method size and nesting.
- Refactoring large or complex methods.

Let me know if you want to refactor the Amplitude logic or address other code health issues.

2 files changed +87 -24
Keep
Undo
Add Context...
KeyboardEventHandlers.cpp:1466
Add context (#), extensions (@), commands (/)
Agent
GPT-4.1

Provide instructions to the AI

The MCP server gets invoked automatically and does a Code Health review.

The Code Health insights let the AI discover the code quality problem and propose a plan 👍

💡 **Tip**: add the code health safeguard as a global custom instruction for your AI-agent.

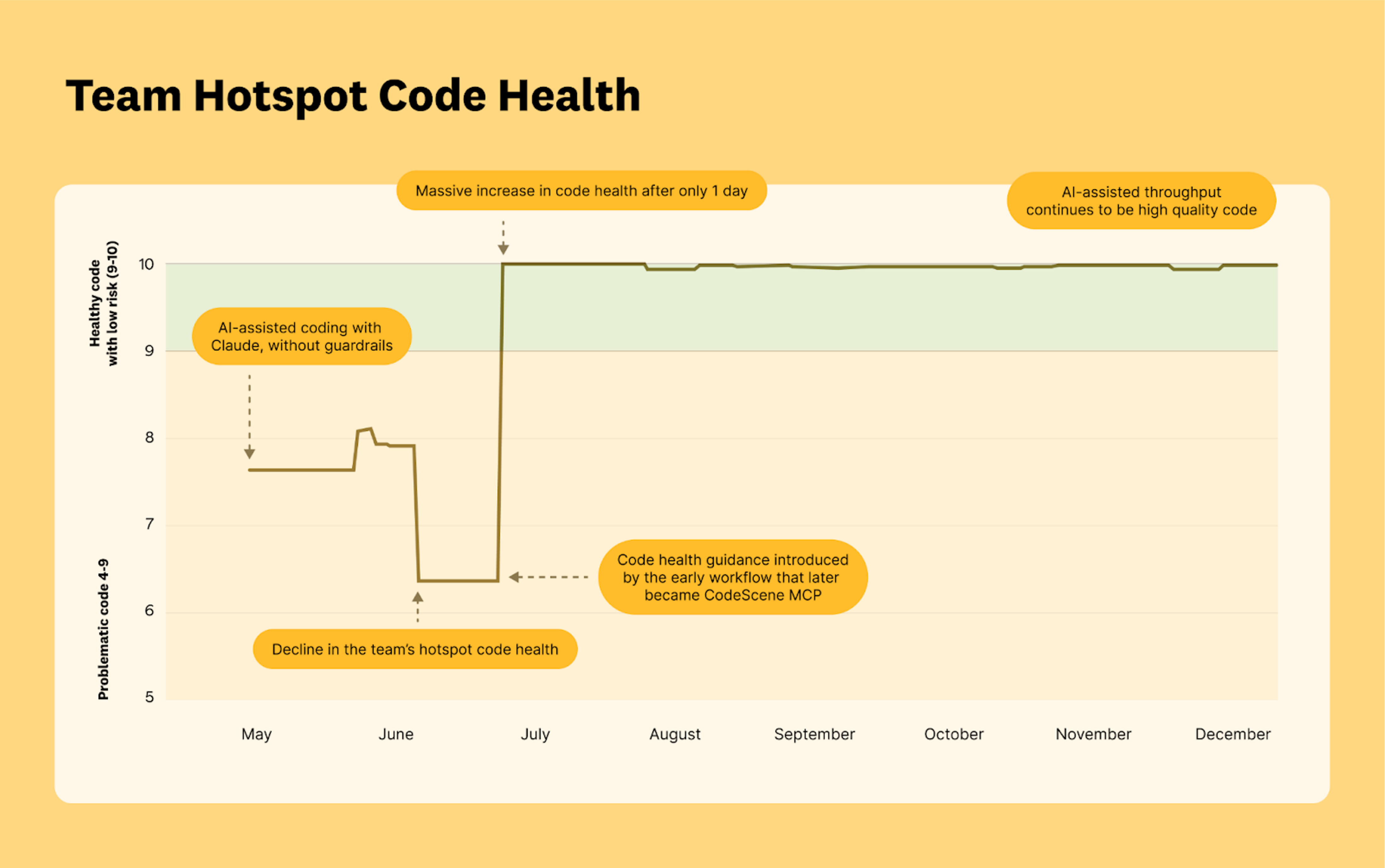
© 2026 CodeScene AB, all rights reserved.

loveholidays proving it in practice

loveholidays

A real-world example comes from **loveholidays**, where early agentic coding with Claude led to **declining code health**. Using an early prototype of what is now **the CodeScene MCP Server**, the team reversed the trend.

Within **5 months**, loveholidays scaled **from 0 to 40% AI-assisted code** while increasing **high throughput** and maintaining **high code quality**.



Massive increase in code health after only 1 day

AI-assisted throughput continues to be high quality code

AI-assisted coding with Claude, without guardrails

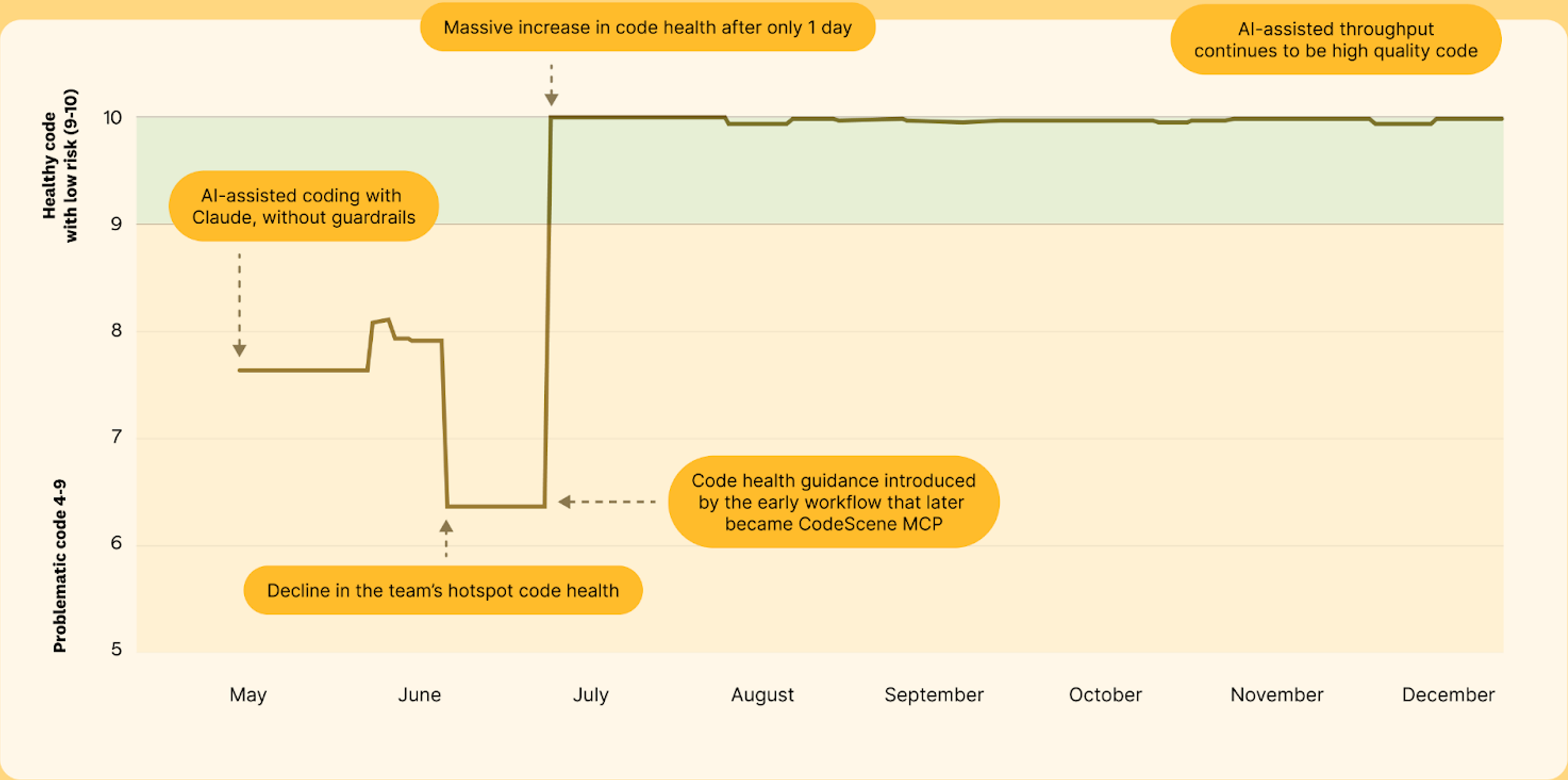
Code health guidance introduced by the early workflow that later became CodeScene MCP

Decline in the team's hotspot code health

> *"AI's promise is delivery efficiency, but efficiency without maintainability isn't progress. CodeScene ensures we know the difference."*
>
> **George Malamidis**, VP of Engineering

© 2026 CodeScene AB, all rights reserved.

Team Hotspot Code Health



Massive increase in code health after only 1 day

AI-assisted throughput continues to be high quality code

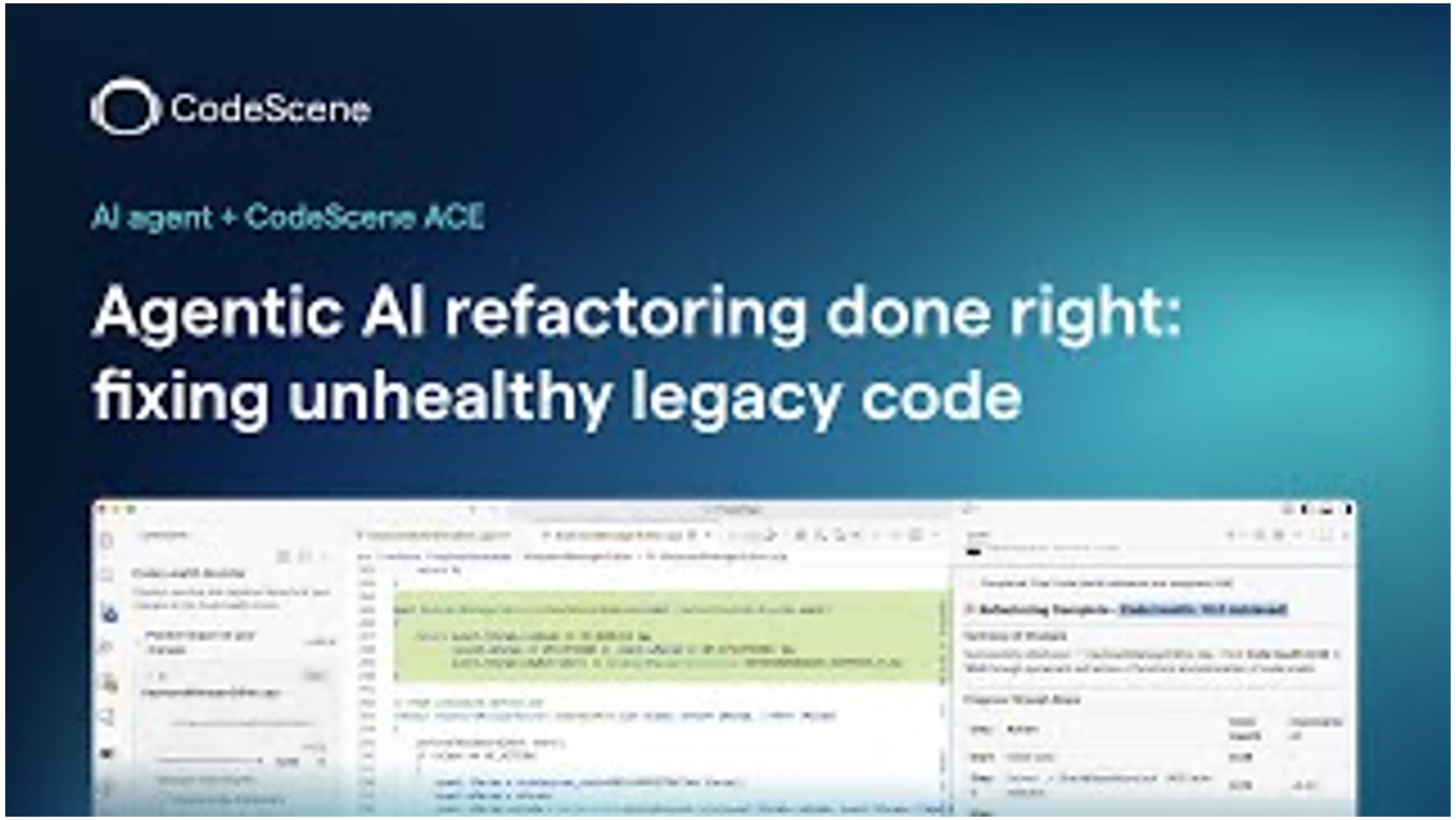
AI-assisted coding with Claude, without guardrails

Code health guidance introduced by the early workflow that later became CodeScene MCP

Decline in the team's hotspot code health

© 2026 CodeScene AB, all rights reserved.

Refactoring time reduced from from days to minutes.



CodeScene

AI agent + CodeScene ACE

Agentic AI refactoring done right: fixing unhealthy legacy code

[Watch the 2-min demo]

© 2026 CodeScene AB, all rights reserved.

Easy do get started (5 min implementation)

CURSOR Claude Windsurf OpenAI **…and more**

MCP server is already **included** in your license

→ **Get token** for the MCP server

→ **Install** MCP server using one of the methods

→ **Add** server to your **AI-coding assistant**

→ **Copy the Agent.md file**, guiding your AI agent how to use the MCP server including rules that safeguard AI coding

The MCP Server is designed to run in your local environment

© 2026 CodeScene AB, all rights reserved.

Your Go-To Guide

All the links you need to get started

Get started with MCP

- Easy Getting Started Guide with documentation to set up and use MCP quickly.
- CodeScene ACE via MCP server: Agentic AI refactoring done right: fixing unhealthy legacy code (2-min demo)
- MCP Server in Action: How CodeScene's MCP Server gives AI-coding Assistants Real Code Quality Insights (2-min demo)

Research

- Code for Machines, Not Just Humans: Quantifying AI-Friendliness with Code Health Metrics — research by Adam Tornhill and Markus Borg, CodeScene
- Whitepaper about the research AI-Ready Code: How Code Health Determines AI Performance
- Read the Report "Can GenAI Actually Improve Developer Productivity?"
- Research "Does AI-Assisted Coding Deliver? A Difference-in-Differences Study of Cursor's Impact on Software Projects"
- Forrester predicts rise in technical debt in 2026

Articles and blog post

- Rewriting the Rules of Code: loveholidays Validates that AI Scale Doesn't Have to Result in Technical Debt — International Business Times
- Read Strengthening the Inner Developer Loop: Turn AI Into a Reliable Engineering Partner — Adam Tornhill

© 2026 CodeScene AB, all rights reserved.