A FIELD REPORT

Lessons from AI-native engineering

What changed, and what you already know

Max Trense · Agentic Software Engineering Offenbach, 11.11.2026

THE PREMISE

The job changed. Most of the failures didn’t.

Small steps and running code we know from Agile. Test rigor and good code are no surprise to a seasoned engineer. Use what you already know.

THE ROUTE

Three acts, and one thing that is new

ACT 1

Small steps, running code

Low ceremony and a demo after every slice.

ACT 2

Don’t trust, verify

An agent’s output is a hypothesis until something checks it.

ACT 3

Context is the lever

When results are bad, fix what you feed the agent.

CLOSING

What’s actually new

The one change no earlier experience prepares you for.

ACT 1 · SMALL STEPS, RUNNING CODE

Heavy process produced documents, not product

Examples and running code outlast specs.

01

Spec-driven frameworks versus small slices

WHAT I TRIED

Spec-driven frameworks

BMAD, GSD and others. The more complicated the framework, the smaller the result.

WHAT WORKED

Small slices, a demo each

Build one slice, show it running, then decide what comes next.

WHY IT WORKS

Planning just in time

Decision records grow from real forks. The vision outlines the product, not its inner workings.

01 · YOU ALREADY KNOW THIS AS

Agile versus waterfall

TAKEAWAY

Plan just in time. Write things down only when a real fork forces you to.

ACT 1 · SMALL STEPS, RUNNING CODE

“Done” is a claim. Make it an observation.

Only running software that solves the actual problem provides value.

02

Every task landed. The server still didn’t serve.

THE CLAIM

“All tasks are done.”

Every task in the group was closed, and each one looked finished on its own.

THE OBSERVATION

Nothing ran

The outcome the tasks were meant to add up to did not work when started.

THE FIX

A check that runs it

A demo turns green only when a command drives the real entry point and passes.

02 · YOU ALREADY KNOW THIS AS

The definition of done and the sprint demo

TAKEAWAY

Stop at something you can demo, and watch it work.

ACT 2 · DON’T TRUST, VERIFY

Review loops often don’t converge

And an agent’s confidence carries no information.

03

Three ways a review loop fails to end

ONE

There is always another

Asked to find problems, the model finds one. Every round, however many came before.

TWO

Ping-pong

The session flips between the same two fixes, iteration after iteration.

THREE

Two runs, two answers

The same prompt twice gives two options, and each run argues strongly for a different one.

03 · YOU ALREADY KNOW THIS AS

Timeboxing and the spike

TAKEAWAY

Run an experiment instead of another review round.

ACT 2 · DON’T TRUST, VERIFY

What must hold has to be mechanical

A mechanical check is safer than a non-deterministic answer.

04

Where mechanical checks can live

TESTS

The test suite

The testing pyramid still applies, and the agent can run it as often as it likes.

LANGUAGE

Language and ecosystem

Choose them for what their compiler and tooling can verify for you.

DESIGN

Architecture and code design

Structure the code so that a rule is enforced, and doesn’t depend on being remembered.

ACT 2 · DON’T TRUST, VERIFY

Your codebase earns the agent’s autonomy

Code that is already in your project is a stronger example than any prompt.

05

What a ready codebase gives the agent

HARNESS

Fast, honest feedback

A type checker and a test suite tell the agent it is wrong before you have to.

EXAMPLES

Code worth imitating

The agent copies what it finds. Existing code beats instructions, for better or worse.

THEN

More autonomy

Widen what the agent may do alone once both are in place.

05 · YOU ALREADY KNOW THIS AS

Onboarding a junior engineer

TAKEAWAY

Build the harness and the examples first. Then widen autonomy.

ACT 2 · DON’T TRUST, VERIFY

Think about reviews before you become the bottleneck

AI generates code faster than any human can take responsibility for it.

06

Explain-back: a fresh agent guesses the task

STEP 1

Show only the diff

A fresh agent with no knowledge of the task reads the pull request’s diff.

STEP 2

Ask what the task was

It writes down what the change was apparently meant to achieve.

STEP 3

Compare with the real task

Every deviation is a hotspot, and that is where you look more closely.

ACT 3 · CONTEXT IS THE LEVER

Fix the prompt, not the conversation

Unlike a colleague, an agent can forget. Use that.

07

An escalation ladder for bad output

FIRST

Fix the prompt

Rewind to before the wrong turn and restate the request with better input.

IF IT RECURS

Fix the standing instructions

Change the rules file, skill or agent definition so the fix outlives the session.

IF IT MUST NEVER RECUR

Make it mechanical

Add a test, a lint rule or a type so the failure can’t get through.

07 · YOU ALREADY KNOW THIS AS

Fixing the root cause, not the symptom

TAKEAWAY

Bad output points at bad input. Go back and fix that.

ACT 3 · CONTEXT IS THE LEVER

The triplet of bad context

Contradiction, ambiguity and irrelevancy creep in faster than you think.

08

Three smells, one example each

CONTRADICTION

Two sources disagree

A global rule says never commit; a skill says commit after every task.

AMBIGUITY

One word, two meanings

“session” means a login in your docs and a chat to the agent.

IRRELEVANCY

Baggage nobody removed

Instructions on how to implement DB migrations when the agent works on the frontend

08 · YOU ALREADY KNOW THIS AS

Requirements hygiene

TAKEAWAY

Hunt for all three in everything the agent reads, including what you didn’t write.

ACT 3 · CONTEXT IS THE LEVER

Agents are users too

People will use your tool through an agent. Build for that caller.

09

From an MCP server to an agent-aware CLI

A START

An MCP server

Offer your product’s functions in a form agents can call directly.

GO FURTHER

`<cli> agent usage`

A command that returns a current description of your tool, in markdown, to the agent calling it.

THE PRINCIPLE

The whole interface

Output, errors and documentation all get read by an agent first.

09 · YOU ALREADY KNOW THIS AS

API design and developer experience

TAKEAWAY

Design your tools for an agent caller.

CLOSING · WHAT'S ACTUALLY NEW

There are no simple, meditative tasks anymore

The agent takes the restful work and leaves you the hard parts of your job.

10

What the agent took off your desk

BEFORE

Coding was the rest

An hour of plain implementation was how you recovered from meetings and hard decisions.

NOW

Only the hard parts

Deciding, designing and verifying remain, back to back.

ALSO GONE

Time to think twice

Walking through the code by foot left room to ask whether you needed the feature at all.

WRAP-UP

Most of this you already knew. One thing is new.

Bring your engineering experience with you into the new age, and pace yourself.

[Your name] · [Contact]