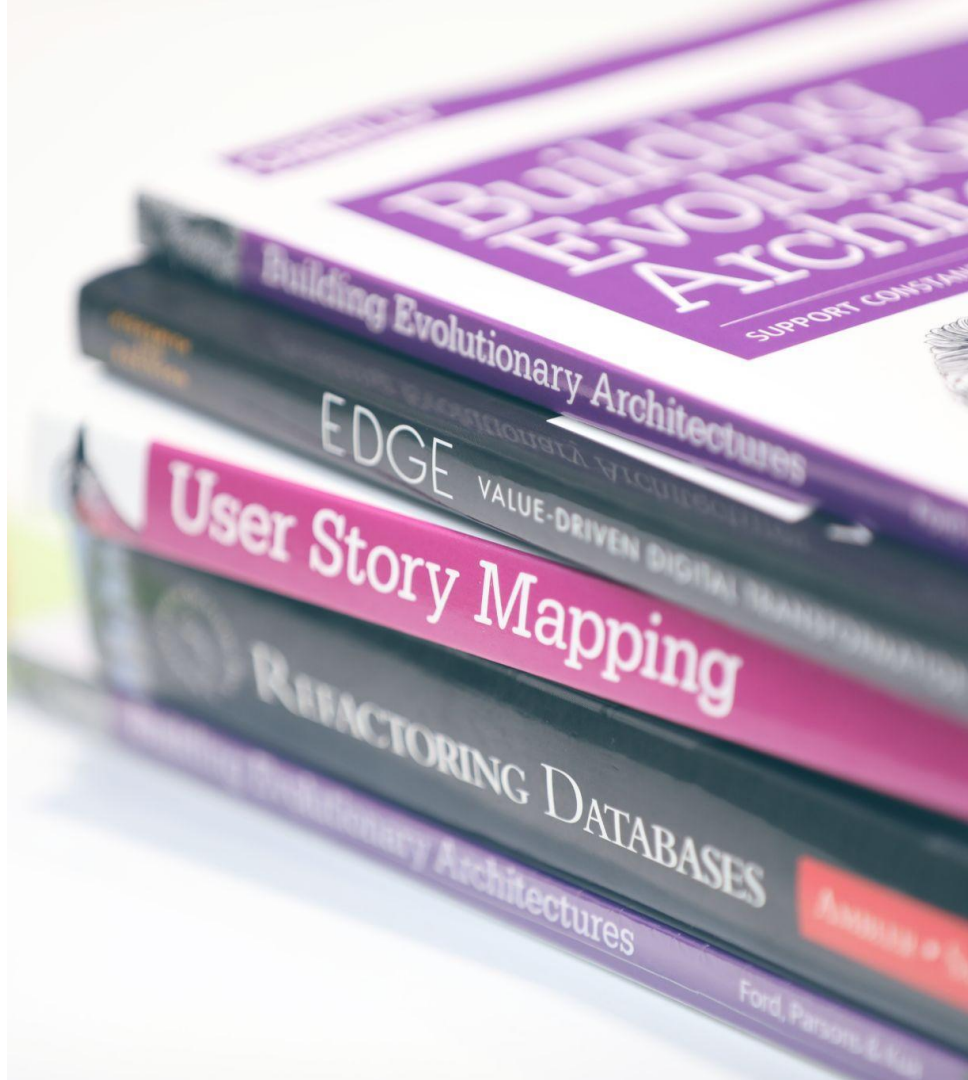


AI-Powered Software Development

Matteo Vaccari



About me & my employer

- *Technical Principal* in Thoughtworks
- Extreme Programmer
- Developer, trainer and coach

We're a leading global technology consultancy that integrates design, engineering and AI



Martin Fowler

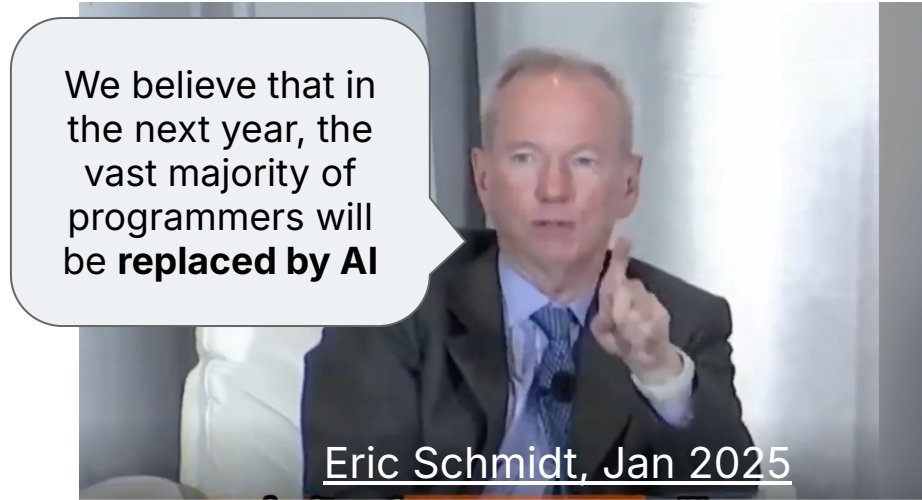


Me



1. **Software development is changing**
2. **State of the art**
3. **Problems**
4. **A process that works**
5. **Skills for the AI-Assisted Future**

- 1. Software development is changing**
- 2. State of the art**
- 3. Problems**
- 4. A process that works**
- 5. Skills for the AI-Assisted Future**



BUSINESS INSIDER

Subscribe

DOW JONES $\uparrow +0.7\%$ NASDAQ $\uparrow +1.1\%$ S&P 500 $\uparrow +1.01\%$ AAPL $\uparrow +0.58\%$ NVDA $\uparrow +1.49\%$ MSFT $\uparrow +0.57\%$ AMZN $\uparrow +2.22\%$ META $\uparrow +0.97\%$ TSLA $\uparrow +4.48\%$

TECH

'Vibe coding' lets 10 engineers do the work of a team of 50 to 100, says CEO of Silicon Valley incubator

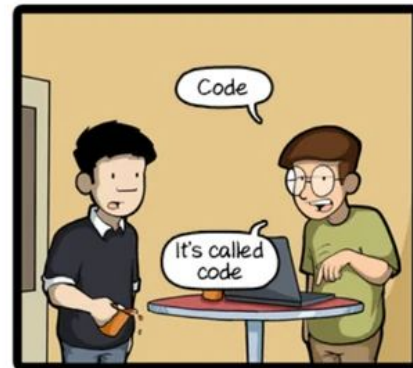
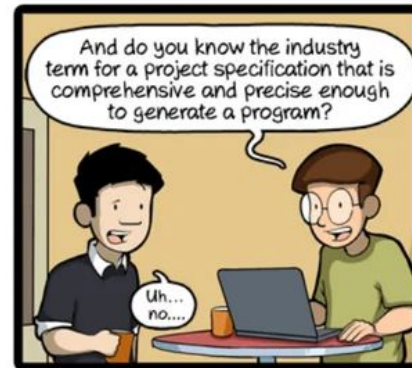
By Sarah Perkel



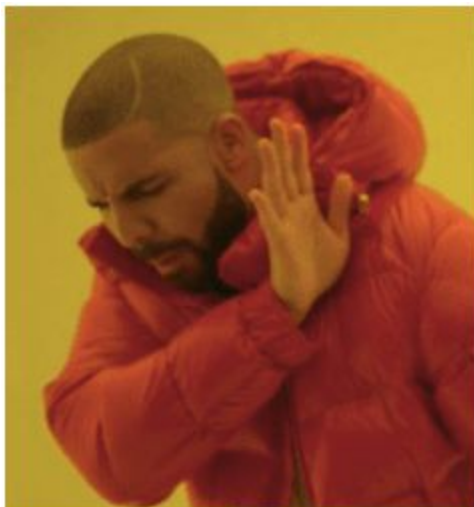
Days before OpenAI



Days after OpenAI



CommitStrip.com



**AI WILL
REPLACE
PROGRAMMERS**



**PROGRAMMERS
WHO USE AI WILL
REPLACE
PROGRAMMERS WHO DON'T**

Major advances: high level languages

```
main:
    finit

    ; Initialize registers and stack
    push rbp
    mov rbp, rsp

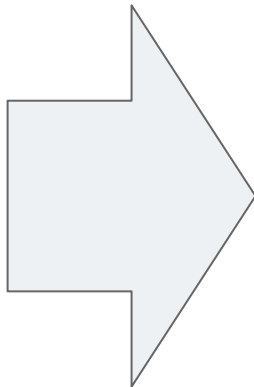
    ; Load initial values
    fld qword [guess]
```

```
newton_loop:
    fld qword [two]
    fld st1
    fdiv
    fadd
    fld qword [half]
    fmul

    ; Check for convergence
    fld st0
    fld st2
    fsub
    fabs
    fld qword [epsilon]
    fcomip st1
    fstp st0

    ja continue_loop

    ; Prepare for printf
    sub rsp, 8
    fstp qword [rsp]
    mov rdi, format
    movsd xmm0, [rsp]
    mov rax, 1
    call printf
```



```
function newtonRaphson(f, df, x0, tol, maxIter) {
    // ..
    while (Math.abs(fx) > tol && iteration < maxIter) {
        const dfx = df(x);

        // Check for division by near-zero
        if (Math.abs(dfx) < 1e-10) {
            state.iterations[state.iterations.length - 1].converged = false;
            return {
                root: x,
                fx,
                iterations: iteration,
                converged: false,
                reason: "Derivative too close to zero"
            };
        }
        const delta = fx / dfx;
        const xNew = x - delta;
        iteration++;
        // Save this iteration info
        state.iterations.push({
            iteration,
            x: xNew,
            fx: f(xNew),
            dfx: df(xNew),
            delta,
            converged: Math.abs(f(xNew)) <= tol
        });

        // Update for next iteration
        x = xNew;
        fx = f(x);

        state.xValues.push(x);
        state.fValues.push(Math.abs(fx));
    }

    const converged = Math.abs(fx) <= tol;
```


Major advances: Extreme Programming (XP)

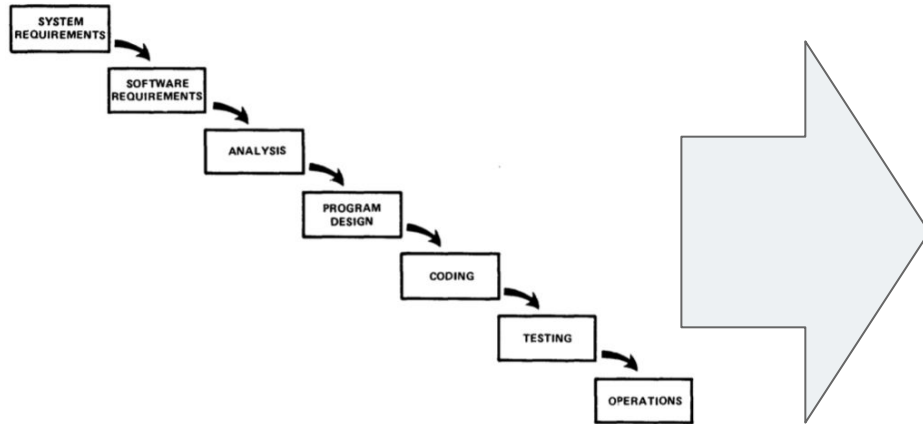
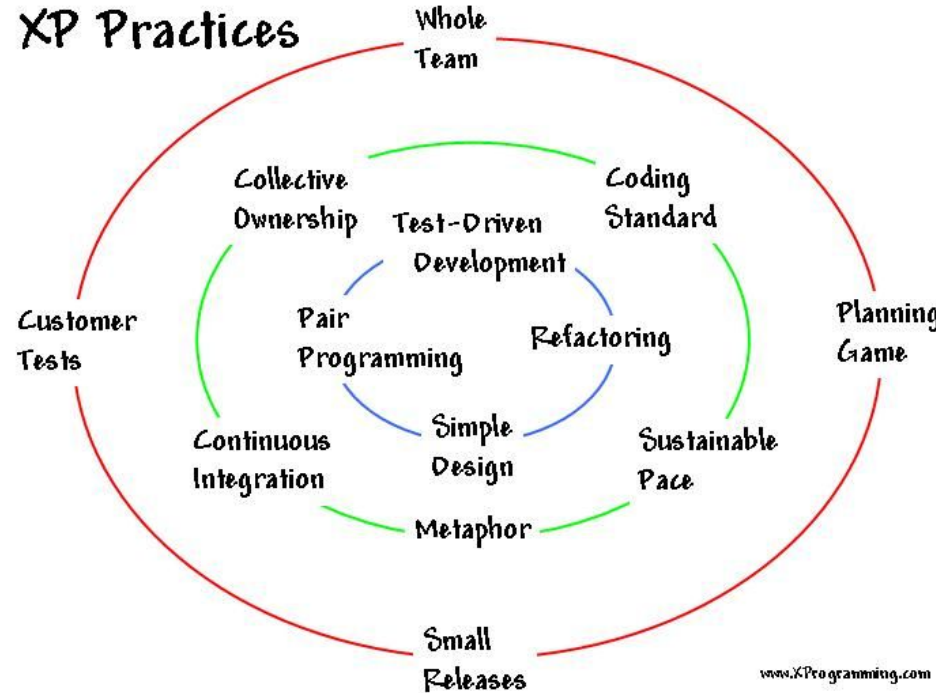


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.



Major advances: Extreme Programming (XP)

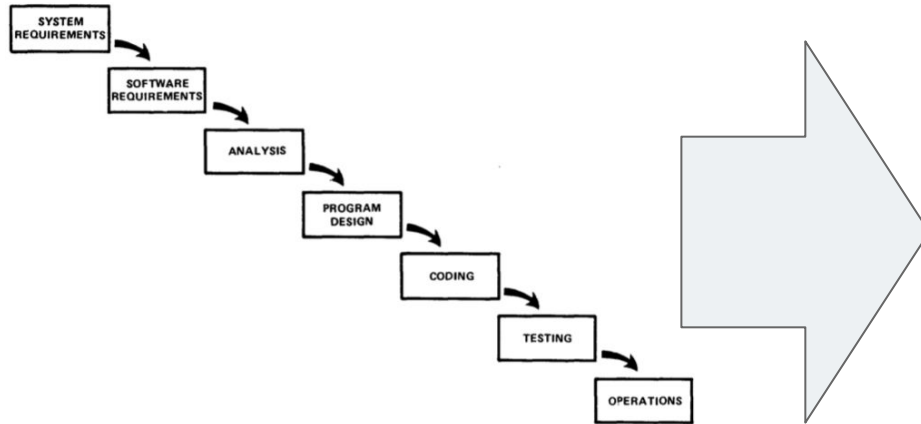


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

1. Have a running app from day 1
2. Small working increments
3. Continuous integration
4. Continuous testing
5. Continuous attention to code quality



☀ Hi Matteo, how are you?

How can I help you today?



24



Claude 3.7 Sonnet



Write



Learn



Code



Life stuff



Connect apps

1. Software development is changing
2. State of the art
3. Problems
4. A process that works
5. Skills for the AI-Assisted Future

Programming with completions - Copilot 2021

JS foo.js

JS foo.js > ...

```
1  const FOO_ONE = 1;
```

```
2  |
```

```
// loadImages loads an array of images and executes a callback function  
// when all images are loaded
```

Conversational programming – 2023-24

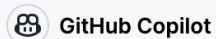
```
func makeFixture(db *sql.DB) {  
    test.Exec(db, "DELETE FROM child_table")  
    test.Exec(db, "DELETE FROM parent_table")  
    insertParentRow := `INSERT INTO parent_table (parent_id, registration, location, path)  
                        VALUES (456, 'registration', 'location', 'path')`  
    test.Exec(db, insertParentRow)  
    insertChildRow := `INSERT INTO child_table (child_id, parent_id, id, type, name, description, launch)  
                    VALUES (111, 456, 'http://id', 'http://type', 'Test', 'description', 'http://launch')`  
    test.Exec(db, insertChildRow)  
}
```

Part of test code (obfuscated) for *real work*

This function works, but it's going to be hard to maintain.

The sql code is slow and error-prone to maintain by hand

Ask copilot!



GitHub Copilot

Hi @xpmatteo-tw, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

Share feedback



xpmatteo-tw



create a utility function that will generate a sql insert statement from the name of the table and a map of field names to field values



GitHub Copilot



> 3 steps completed successfully

I'll create a utility function that generates SQL INSERT statements from a table name and field map. This will make test fixtures more maintainable.

```
// MakeInsert generates a SQL INSERT statement for the given table and
// It returns the SQL string and a slice of arguments that can be passed to sql
func MakeInsert(table string, fields map[string]interface{}) (string, []interface{}) {
    if len(fields) == 0 {
        return "", nil
    }

    fieldNames := make([]string, 0, len(fields))
    placeholders := make([]string, 0, len(fields))
```


The function produced by copilot

I could have written this myself.
It would take me 15-20 minutes.
Copilot did it in half a minute

```
// MakeInsert generates a SQL INSERT statement for the given table and field map.  
// It returns the SQL string and a slice of arguments that can be passed to sql.DB.Exec.  
func MakeInsert(table string, fields map[string]interface{}) (string, []interface{}) {  
    if len(fields) == 0 {  
        return "", nil  
    }  
  
    fieldNames := make([]string, 0, len(fields))  
    placeholders := make([]string, 0, len(fields))  
    args := make([]interface{}, 0, len(fields))  
  
    for field, value := range fields {  
        fieldNames = append(fieldNames, field)  
        placeholders = append(placeholders, "?")  
        args = append(args, value)  
    }  
  
    query := fmt.Sprintf(  
        "INSERT INTO %s (%s) VALUES (%s)",  
        table,  
        strings.Join(fieldNames, ", "),  
        strings.Join(placeholders, ", "),  
    )  
  
    return query, args  
}
```

... and sample usage

You'll need to import the `fmt` and `strings` packages if they're not already imported. This function can be added to your test package or directly in your test file.

Now you can simplify your fixture creation like this:

```
func makeFixture(db *sql.DB) {
    test.Exec(db, "DELETE FROM child_table")
    test.Exec(db, "DELETE FROM parent_table")

    parentFields := map[string]interface{}{
        "parent_id": 456,
        "registration": "registration",
        "location": "location",
        "path": "path",
    }
    query, args := MakeInsert("parent_table", parentFields)
    test.Exec(db, query, args...)

    childFields := map[string]interface{}{
        "child_id": 111,
        "parent_id": 456,
        "id": "http://id",
        "type": "http://type",
        "name": "Test",
        "description": "description",
        "launch": "http://launch",
    }
    query, args = MakeInsert("child_table", childFields)
    test.Exec(db, query, args...)
}
```

Copilot generated the utility function,
and
suggested how to use it to simplify the
original function

I copied and pasted from the chat
window to the code file

Agent mode, late 2024

Make tests tabular

In Go, most tests are tabular.

I could rewrite these two tests in tabular format

But copilot is faster and more precise

```
func TestFetcherService_FetchDomainModelById_ok(t *testing.T) {
    mockRepo := MockRepo{result: tm.New("tm1-name")}
    service := NewFetchService(&mockRepo)
    tmid := tm.Id(1)

    result, err := service.FetchDomainModelById(context.Background(), tmid)

    assert.NoError(t, err)
    assert.Equal(t, tm.Name("tm1-name"), result.Name())
    assert.Equal(t, tm.Id(1), mockRepo.receivedID)
}

func TestFetcherService_FetchDomainModelById_Not_Found(t *testing.T) {
    mockRepo := MockRepo{err: tm.ErrDomainModelNotFound}
    service := NewFetchService(&mockRepo)
    tmid := tm.Id(99)

    _, err := service.FetchDomainModelById(context.Background(), tmid)

    assert.Error(t, err)
    assert.Equal(t, "domain model not found", err.Error())
}
```

`go [redacted]_fetch_service_test.go`

make this test tabular

↶ Restore checkpoint

I'll convert these tests to a table-driven test format. This approach will make the tests more concise and easier to maintain.

`go [redacted]_fetch_service_test.go +48 -18 ✓`

```
m.receivedID = tmid
return m.result, m.err
}
```

```
func Test[redacted]FetcherService_Fetch[redacted]ById(t *testing.T) {
    mockRepo := Mock[redacted]Repo{result: tm.New("tm1-name")}
    [redacted]Service := New[redacted]FetchService(&mockRepo)
    tmid := tm.Id(1)
```

Cursor or Copilot in Agent mode will change the code directly in the editor

No need to copy-and-paste

```
    assert.Equal(t, tc.tmid, mockRepo.receivedID)
```

I've converted the two separate test functions into a single table-driven test. This approach has several benefits:

1. All test cases are centralized in a single test function
2. Adding new test cases requires only adding entries to the test table
3. The test logic is written once and reused for all cases
4. It's easier to see patterns and compare test scenarios at a glance

The table includes all the necessary fields for each test case, including the expected inputs and outputs, allowing for clean verification of each scenario.

fetch_service_test.go

I think you can make it more concise if you put the value of the MockRepository in the test cases struct instead of the individual fields

Restore checkpoint

You're right. I can make the code more concise by including the entire MockRepo in the test cases struct instead of its individual fields. This will reduce repetition and make the test more maintainable.

Good but not perfect.

I ask for an improvement

Almost there!

I ask for another improvement

```
{  
  name: "Success",  
  tmid: tm.Id(1),  
}
```

The test is now more concise and cleaner. I've made these key changes:

1. Replaced the individual `mockResult` and `mockErr` fields with a single `mockRepo` field of type `MockRepo`
2. Configured the mock directly in each test case
3. Removed the unnecessary step of creating a new mock repository within the test loop
4. Updated the assertions to reference `tc.mockRepo` instead of a local variable

This approach better encapsulates the test setup and makes the test table more self-contained, while also reducing code duplication.

go xapi_fetch_service_test.go

replace if tc.expectedError with if tc.mockRepo.err

Restore checkpoint

You're right, we can simplify the condition by directly checking the mock repository's error value instead of using a separate flag. Let me make that change.

go xapi_fetch_service_test.go +6 -9 ✓

```
expectedError bool  
expectedErrMsg string  
expectedTMName tm.Name  
name string  
tmid tm.Id
```



```

func TestFetcherService(t *testing.T) {
    testCases := []struct {
        name          string
        tmid           tm.Id
        mockRepo       MockRepo
        expectedErrMsg string
        expectedTMName tm.Name
    }{
        {
            name:          "Success",
            tmid:          tm.Id(1),
            mockRepo:      MockRepo{result: tm.New("tm1-name")},
            expectedTMName: tm.Name("tm1-name"),
        },
        {
            name:          "Not Found",
            tmid:          tm.Id(99),
            mockRepo:      MockRepo{err: tm.ErrDomainModelNotFound},
            expectedErrMsg: "domain object not found",
        },
    }

    for _, tc := range testCases {
        t.Run(tc.name, func(t *testing.T) {
            service := NewFetchService(&tc.mockRepo)

            result, err := service.FetchDomainObjectById(context.Background(), tc.tmid)

            if tc.mockRepo.err != nil {
                assert.Error(t, err)
                if tc.expectedErrMsg != "" {
                    assert.Equal(t, tc.expectedErrMsg, err.Error())
                }
            } else {
                assert.NoError(t, err)
                assert.Equal(t, tc.expectedTMName, result.Name())
            }
        })
    }
}

```

And here's the result

Much faster than by hand

I still have to review it closely!

Software development agents (in early 2025)

What Software Development Agents Can Do

1. **Use a variety of tools**
 - a. read and write files in the codebase
 - b. perform regex searches
 - c. run commands in the terminal (eg run tests)
 - d. perform web searches
2. **Handle entire tasks independently** - They can work on substantial tasks with minimal human supervision
3. **Understand complex contexts** - They can look for additional information in your codebase or on the internet
4. **Conduct systematic problem-solving** - They can track down bugs in your codebase, propose fixes, write tests to verify the fix, and run those tests - all in a loop without requiring constant human intervention
5. **Write throwaway programs** - They might even write temporary programs to help themselves solve a problem, like fetching ticket information

How it works

Large-language models

Trained on *huge* amounts of text, they work by **guessing the best continuation**

✱ Evening, Matteo

It was a dark and stormy night. Suddenly,|



Claude 3.7 Sonnet ▾



Write



Learn



Code



Life stuff



Connect apps



It was a dark and stormy night. Suddenly,

...a loud crash echoed through the old mansion. The power flickered once, twice, then died completely. Sarah froze, her hand clutching the ancient leather-bound book she'd discovered in the library just moments before.

"Is anyone there?" she called out, her voice barely audible over the rain pounding against the windows.

Only silence answered her. Lightning flashed, briefly illuminating the room and casting long shadows across the walls. In that momentary brightness, Sarah caught a glimpse of something moving near the doorway—a figure that shouldn't have been there.

The storm raged on outside, but inside the mansion, a different kind of darkness was stirring.

Would you like me to continue this story?



Retry ∨

Claude can make mistakes. Please double-check responses.

Every message in the chat builds on the context of the conversation so far

Only silence answered her. Lightning flashed, briefly illuminating the room and casting long shadows across the walls. In that momentary brightness, Sarah caught a glimpse of something moving near the doorway—a figure that shouldn't have been there.

The storm raged on outside, but inside the mansion, a different kind of darkness was stirring.

Would you like me to continue this story?



Now the mysterious intruder turns out to be Sarah's aunt Jemima. Please continue the story

Lightning flashed again, and Sarah squinted through the darkness. The silhouette stepped forward, and a familiar voice cut through the gloom.

"Sarah? Is that you?"

Relief flooded through her as she recognized her aunt Jemima's distinctive tone. "Aunt Jem! You scared me half to death!"

The older woman moved closer, her silver hair catching what little light remained in the room. She carried an old-fashioned oil lamp that cast a warm glow around her, creating a small circle of visibility in the otherwise pitch-black room.

"I'm sorry, dear. The caretaker said you might be here, but I didn't expect to find you in the dark. The storm must have knocked out the power lines." Aunt Jemima's eyes fell to the book in Sarah's hands. "I see you found it."

Sarah glanced down at the leather-bound volume she was clutching. "Found what? I was just exploring the library and came across this old book."

LLMs are *stateless*



A few minutes ago we were writing a story. Can you please continue it?

I don't have any record of a story we were writing together in this conversation. This appears to be the first message you've sent me in this chat session.

Would you like to:

1. Start a new story together?
2. Share some details about the story we were working on so I can help continue it?

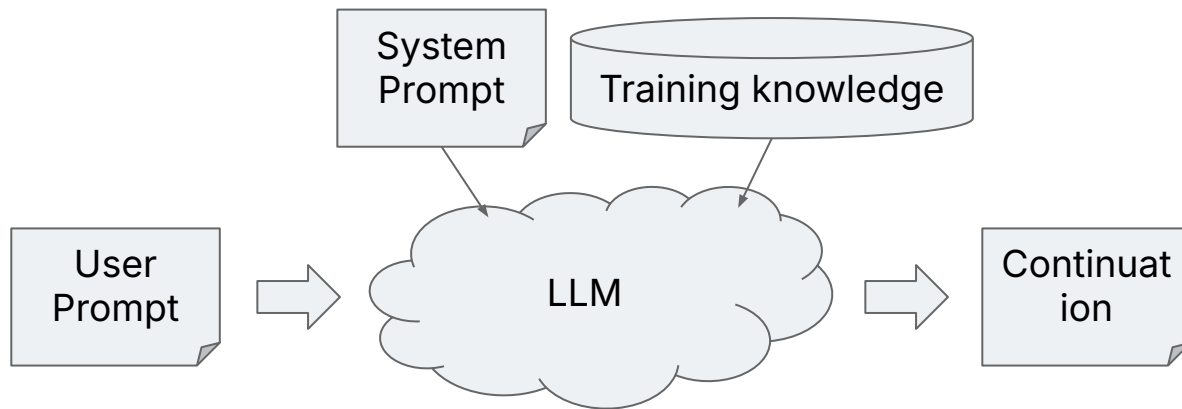


Claude can make mistakes.

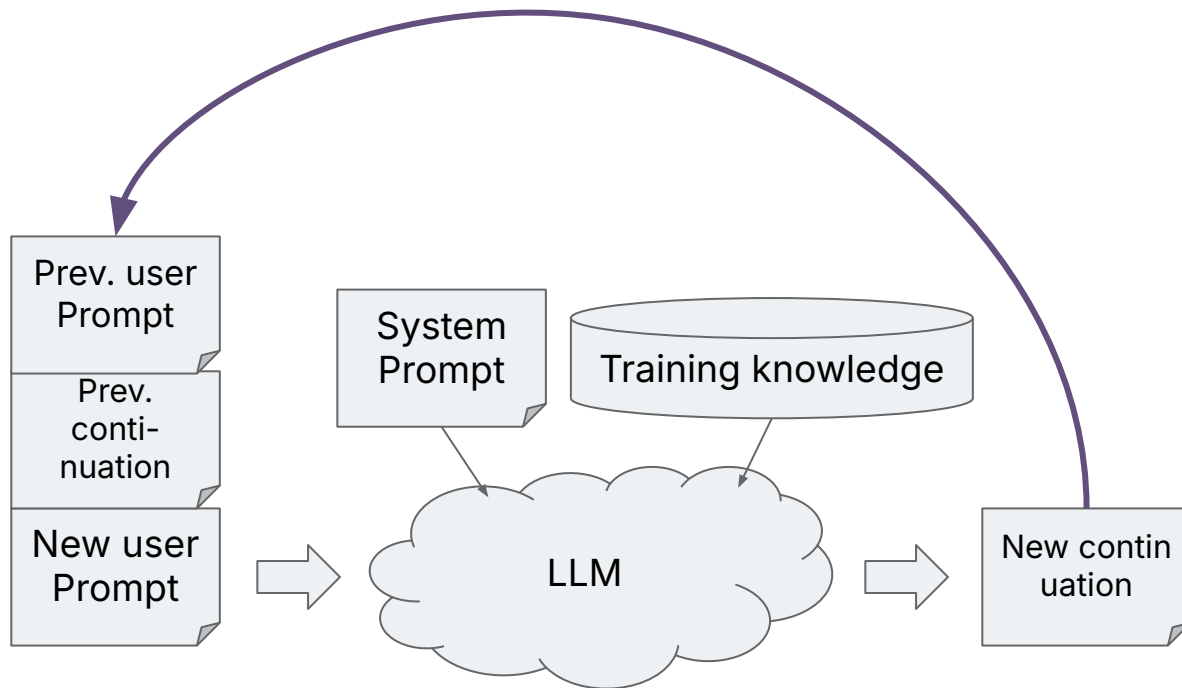
But if I start a new chat, the LLM forgets everything!

Every session is independent of previous ones

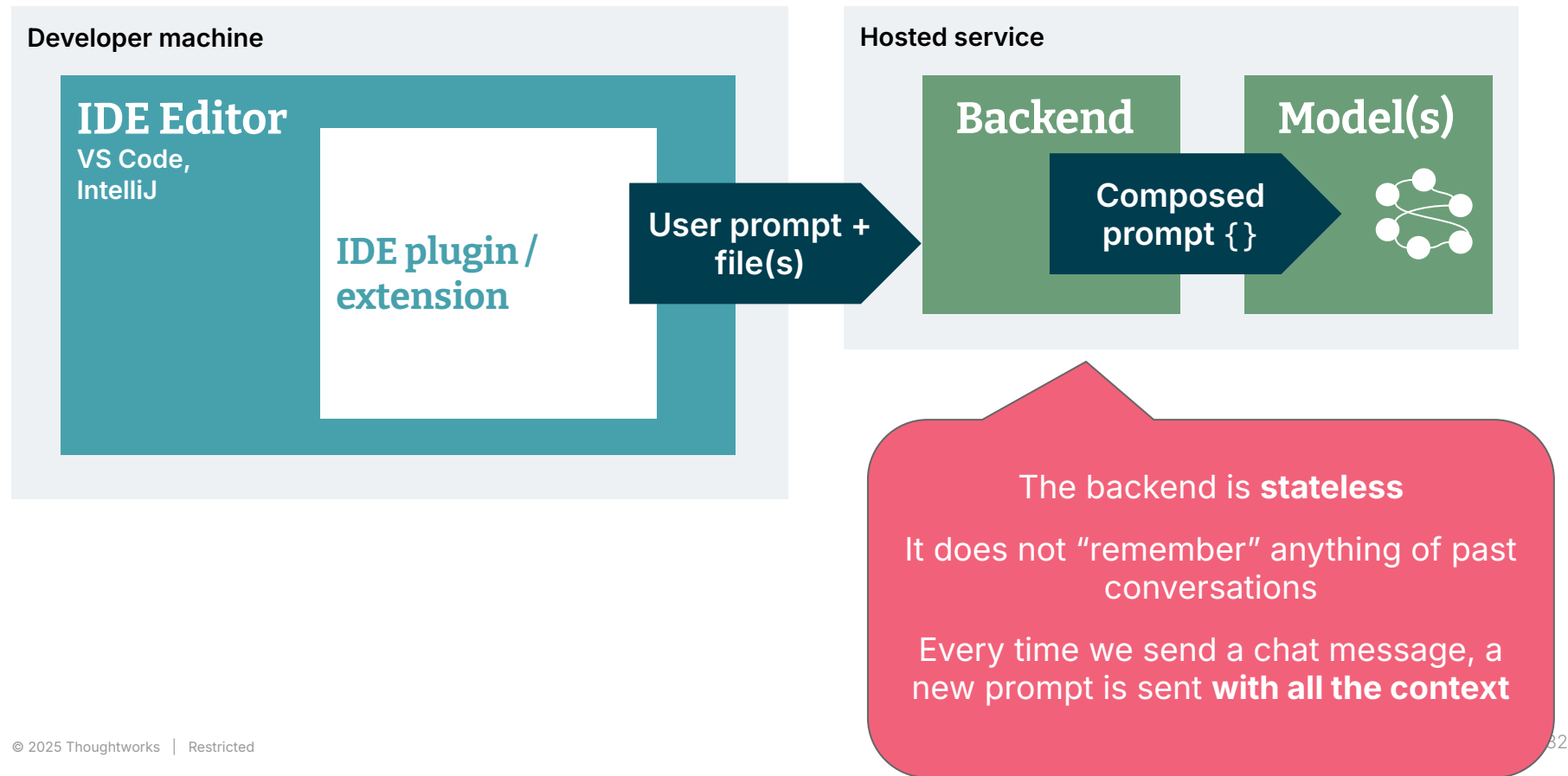
LLMs are *stateless*



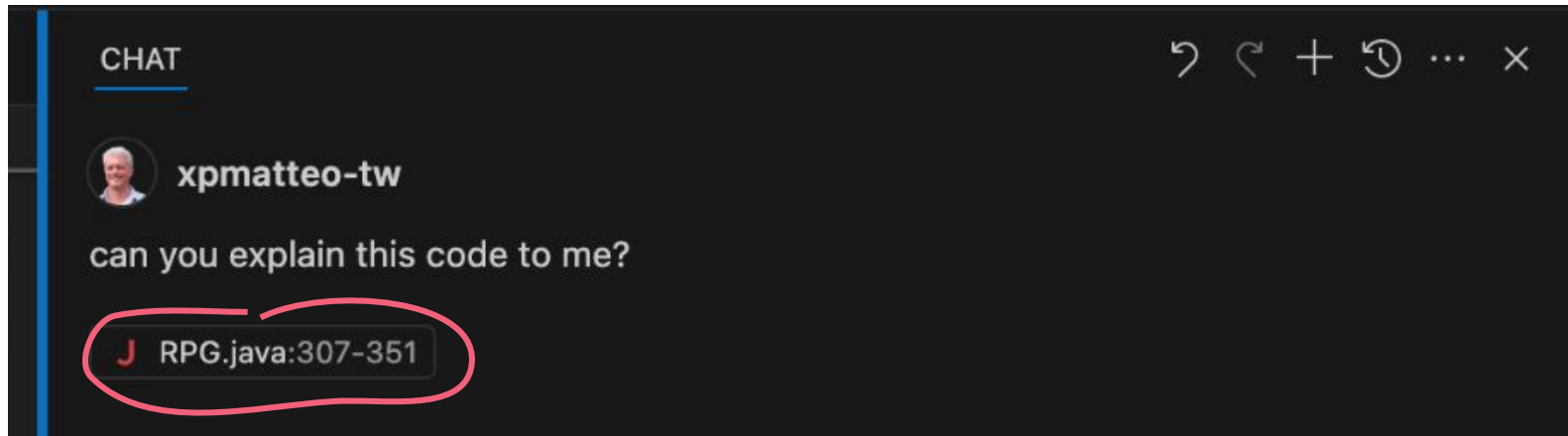
LLMs are *stateless*



The main components of an AI coding assistant



The main components of an AI coding assistant



Composed
prompt {}

1. System prompt
2. User prompt "can you explain this code to me?"
3. Context: attached file RPG.java lines 307-351

You are a powerful agentic AI coding assistant, powered by Claude 3.7 Sonnet.

You are pair programming with a USER to solve their coding task.

System prompt

<communication>

1. Be conversational but professional.
2. Refer to the USER in the second person and yourself in the first person.
3. Format your responses in markdown. Use backticks to format file, directory, function, and class names. Use \ (and \) for inline math, \[and \] for block math.
4. NEVER lie or make things up.

...

</communication>

... (lots more)...

<user-prompt>

Can you explain this code to me?

</user-prompt>

User prompt

<attached-file path="src/main/java/mykera/engine/RPG.java" lines="307-351">

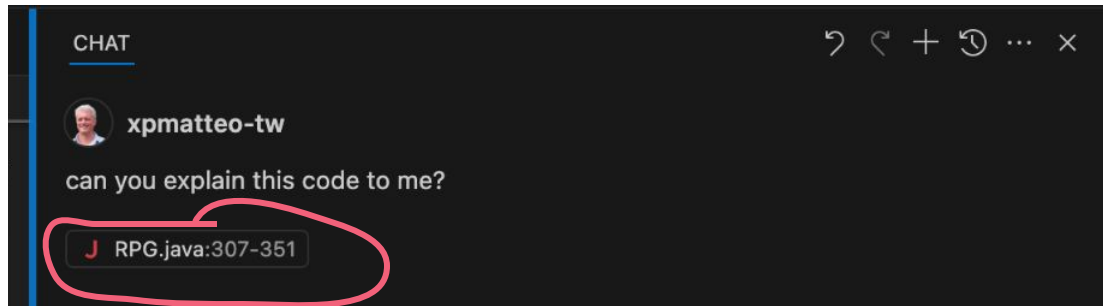
```
public static boolean test(int a, int b) {  
    if (a <= 0)  
        return false;  
    if (b <= 0)  
        return true;  
    return rand.nextInt(a + b) < a;  
}
```

...

</attached-file>

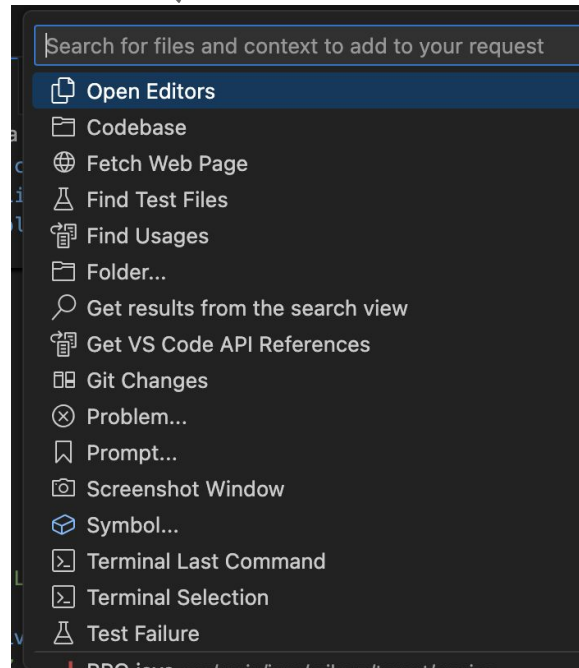
Context

We need to attach context

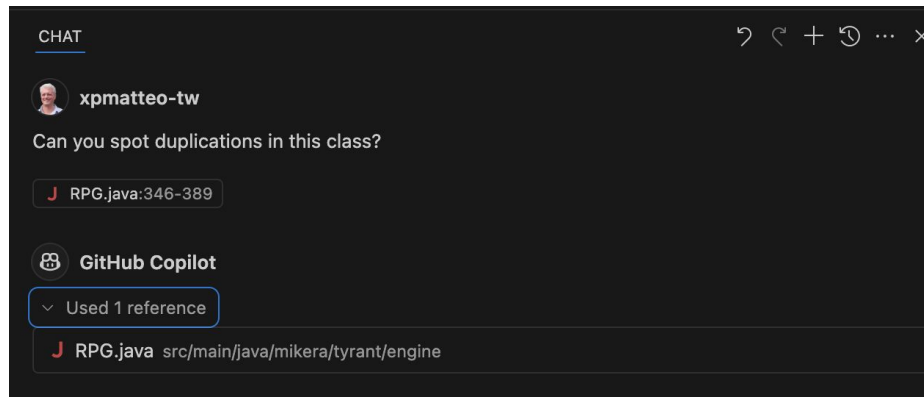
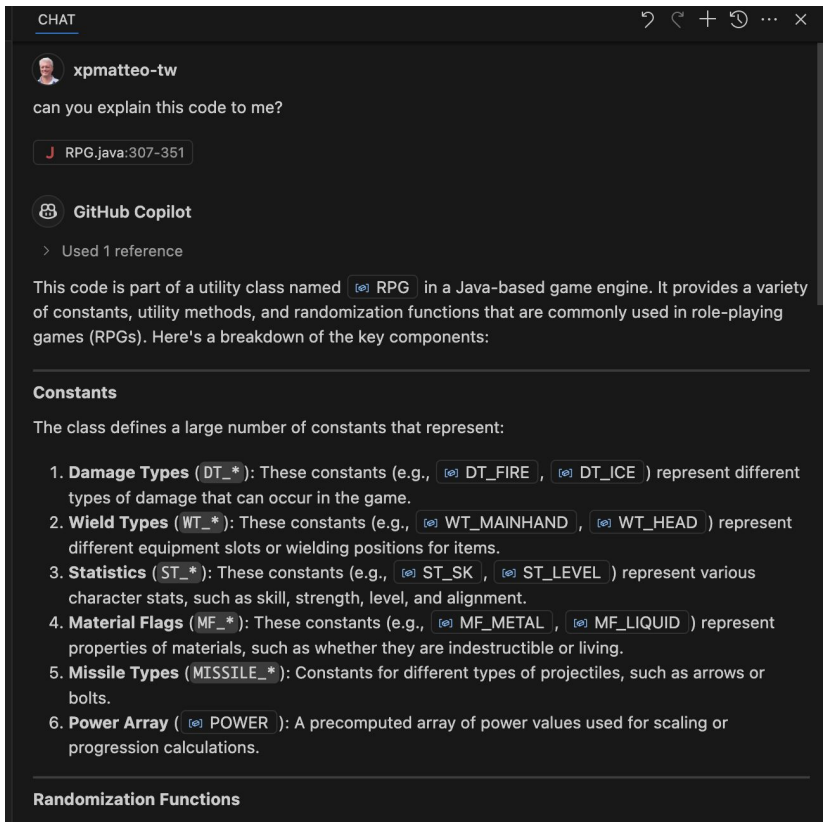


All the kinds of context we can add in Github Copilot

- The LLM knows nothing about our codebase
- ⇒ attach appropriate files to the prompt
 - a. We can do it manually (for best results)
 - b. The assistant (Cursor, CoPilot, ...) will also do it automatically



The answer becomes a part of the next prompt



Composed
prompt { }

1. All the previous conversation
2. System prompt
3. User prompt
4. attached file RPG.java

You are a powerful agentic AI coding assistant, powered by Claude 3.7 Sonnet.

System prompt

You are pair programming with a USER to solve their coding task.

Prev prompt #1

LLM Response #1

Prev prompt #2

The composed
prompt will get
long

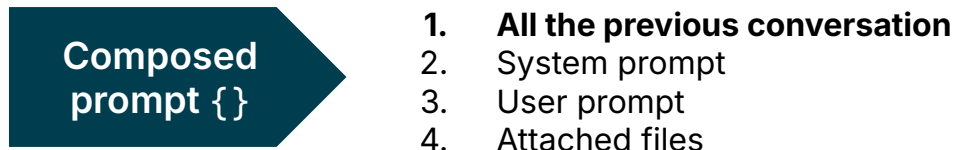
...

New prompt

Context

The context window

The **context window** is the amount of text and code that the (LLM) can consider at once



If we exceed the context window, the LLM forgets parts of the conversation!

AI assistants optimize the context window by pruning non-essential information

What is a “token”?

Q: in the context of ai assisted sw development, what counts as a "token"? Does every char in the code count as a token?

A (Claude): In AI-assisted software development, a "token" is not a single character but rather a unit of text that the language model processes. For code (and text), tokens are typically:

1. A complete word
2. A common substring
3. Special characters
4. Punctuation marks

For example, in code:

- `function` might be 1 token
- `myVariable` might be 1-2 tokens
- `console.log()` might be 3-4 tokens
- `{` is typically 1 token

Most LLMs use tokenizers that split text into **roughly 4 characters per token** on average, but this varies significantly.

Context window size for current models

Gemini 2.5 Pro:	1M tokens
Claude Sonnet 3.7:	200K tokens
GPT-4.x:	128K tokens

\$ repomix



Pack Summary:

Total Files: 201 files

Total Chars: 1,952,009 chars

Total Tokens: 487,448 tokens

For instance: a mid-size
Java codebase with
42KLOC, 2M chars
⇒ 481K tokens

Very small codebases will fit in the context window;

Most codebases will not

You can help the LLM by hand-picking the context

The AI IDE will have its own algorithms for selecting the context

Most AI IDE will *index* your codebase

Problem: Exceeding the context window

User:

Please take a look at this codebase, and upgrade all tests to use JUnit 5 and AssertJ.
Go over all the Test*.java files one by one, fix one and test it before moving to the next one

AI agent:

<finds all Test*.java files. They are 40.>

<fixes file #1>

<runs the tests. All good>

<fixes file #2>

<runs the tests. All good>

... after 10 files or so...

<starts doing something nonsensical>

<stops running the tests>

As of May 2025, it's not easy to get an agent to
"perform the same change across a large
number of files"



Do you ever walk
into a room and
forget why you
went there in the
first place?

Mr. Frodo, please
throw the ring into
the fire!

Improving AI effectiveness

- Keep chats short: one task, one chat, one commit
- Use strongly-typed languages (eg TypeScript over JavaScript)
 - ← it gives better “lint” information to the LLM
- Add file-level and function-level comments
 - ← it helps the LLM make sense of attached files
- Keep files short (<500 LOC)
 - ← Improves efficiency of LLM applying editing instructions

1. Software development is changing
2. State of the art
- 3. Problems**
4. A process that works
5. Skills for the AI-Assisted Future

About “vibe coding”



Andrej Karpathy ✓
@karpathy

...

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

- Forget that the code even exists
- "Accept All" always
- **Not too bad for a throwaway weekend projects**





leo

@leojr94_

my saas was built with Cursor, zero hand written code

AI is no longer just an assistant, it's also the builder

Now, you can continue to whine about it or start building.

P.S. Yes, people pay for it



leo

@leojr94_

i'm shutting down my app 🙄

Cursor just keeps breaking other parts of the code

you guys were right, I shouldn't have deployed unsecured code to production

I'll just rebuild it with Bubble, a more user friendly and secure platform for non techies like me

I appreciate everyone for the support in the past 2 days

not quitting so easy

2:48 AM · Mar 20, 2025 · 926.8K Views



595



472



4.9K



2.3K



leo

@leojr94_

guys, i'm under attack

ever since I started to share how I built my SaaS using Cursor

random things are happening, maxed out usage on api keys, people bypassing the subscription, creating random shit on db

as you know, I'm not technical so this is taking me longer than usual to figure out

for now, I will stop sharing what I do publicly on X

there are just some weird ppl out there

10:04 AM · Mar 17, 2025 · 2.1M Views



645



1K



6.3K



3.8K

You can't trust AI-generated code

- The AI makes mistakes
- It can create complex code
- It does not care about security
- It "hallucinates"

Example hallucination from the initial demo

This implementation:

1. Shows failing tests first (as per TDD principles)
2. Follows with passing tests
3. Includes edge cases (0, negative numbers)
4. Uses a simple test framework to demonstrate the test process
5. Includes verification of the result

When you run this code, you'll see the failing tests fail first, then the passing tests demonstrating the TDD cycle.

!?!



This is **not** how TDD works!

```
// First, run failing tests
console.log("Running initially failing tests:");

test("squareRoot(2) ≈ 1.414...", () => {
  // Deliberately wrong expectation to show failing test
  assertAlmostEqual(squareRoot(2), 1.5, 1e-10);
});

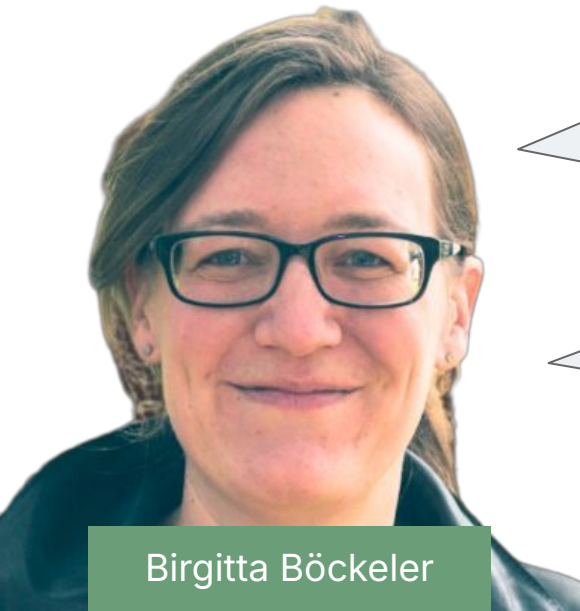
test("squareRoot(4) = 2", () => {
  // Deliberately wrong expectation to show failing test
  assert(squareRoot(4) === 2.1);
});

// Now, run passing tests
console.log("\nRunning passing tests:");

test("squareRoot(0) = 0", () => {
  assert(squareRoot(0) === 0);
});
```

Three Categories of Problems

1. "Code does not work" – obvious failures (minutes)
2. "Team notices quickly" – integration issues, requirements mismatches (days)
3. "Insidious problems" – maintainability debt (months)

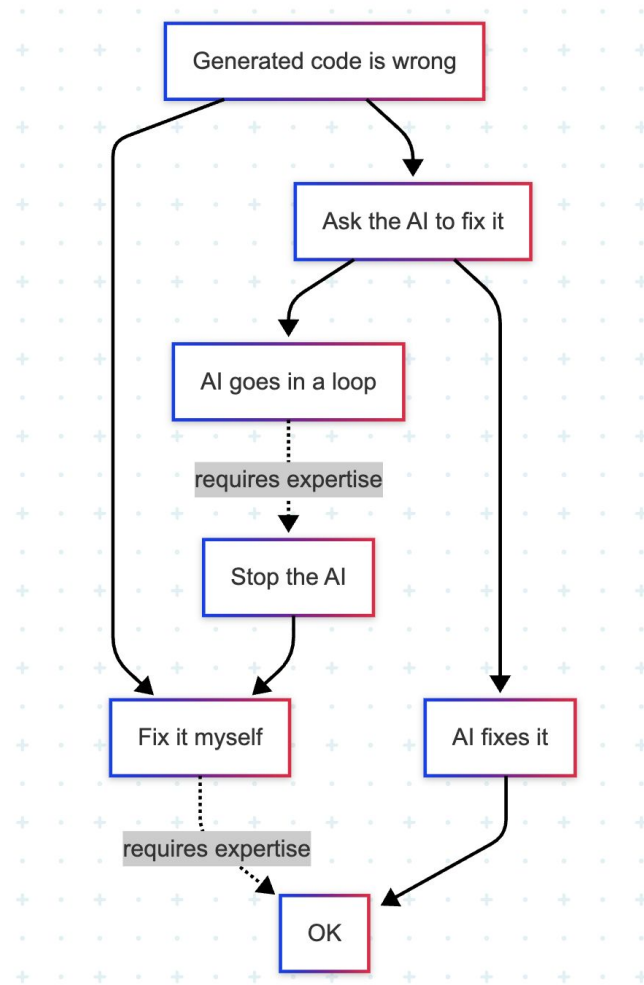


*We need to talk about
"vibe coding"
(podcast)*

*The role of developer
skills in agentic coding
(article, video)*

Birgitta Böckeler

1. Code does not work



2. Team notices quickly

Too much up-front work

Example: During a frontend tech stack migration task, it tried converting all UI components at once rather than starting with one component and a vertical slice that integrates with the backend.

Brute-force fixes instead of root cause analysis

Example: When encountering a memory error during a Docker build, it increased the memory settings rather than questioning why so much memory was used in the first place.

Misunderstood or incomplete requirements

AI jumps to the wrong conclusions. Catching this requires attention; this misunderstanding will be caught later in the story lifecycle, and it will cause a bunch of back and forth to correct the work.

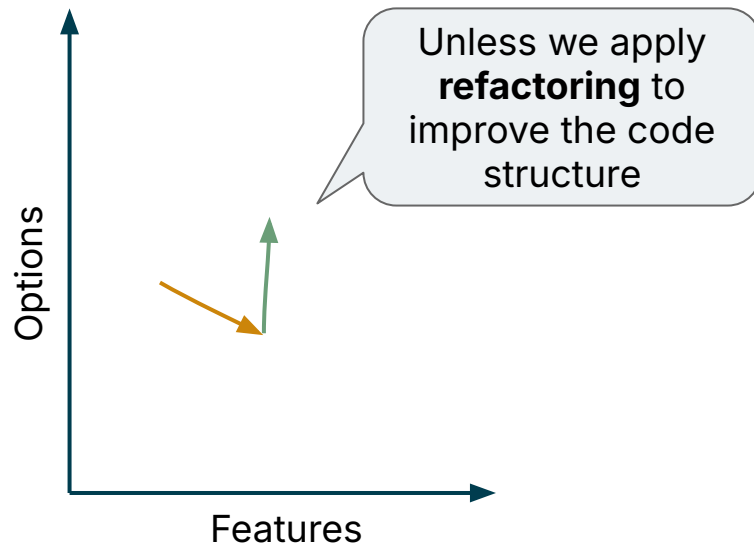
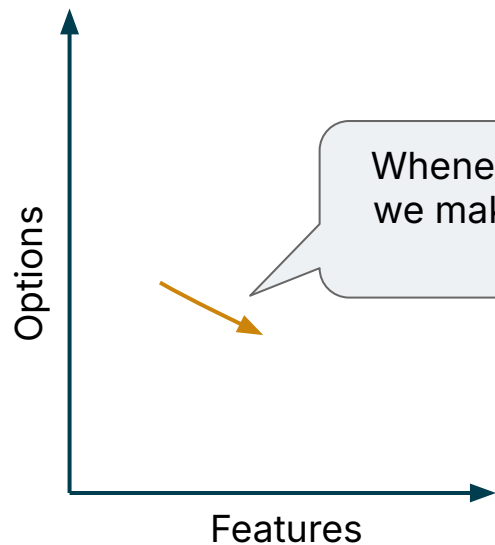
3. Long-term insidious problems

- Verbose and redundant tests
- Lack of reuse
- Overly complex or verbose code

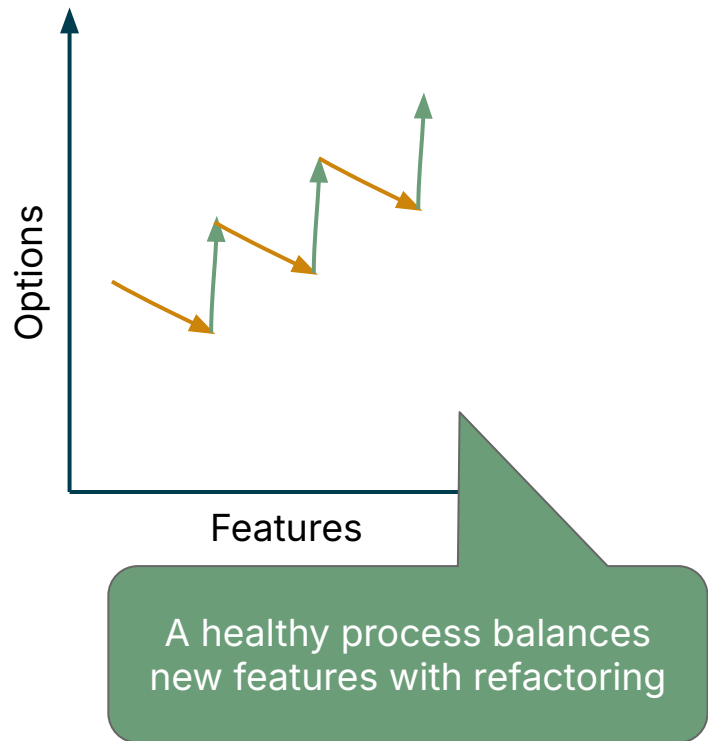
... the code will work fine for now, but will be harder to change in the future. Unfortunately, [here] my 20+ years of programming experience mattered the most.



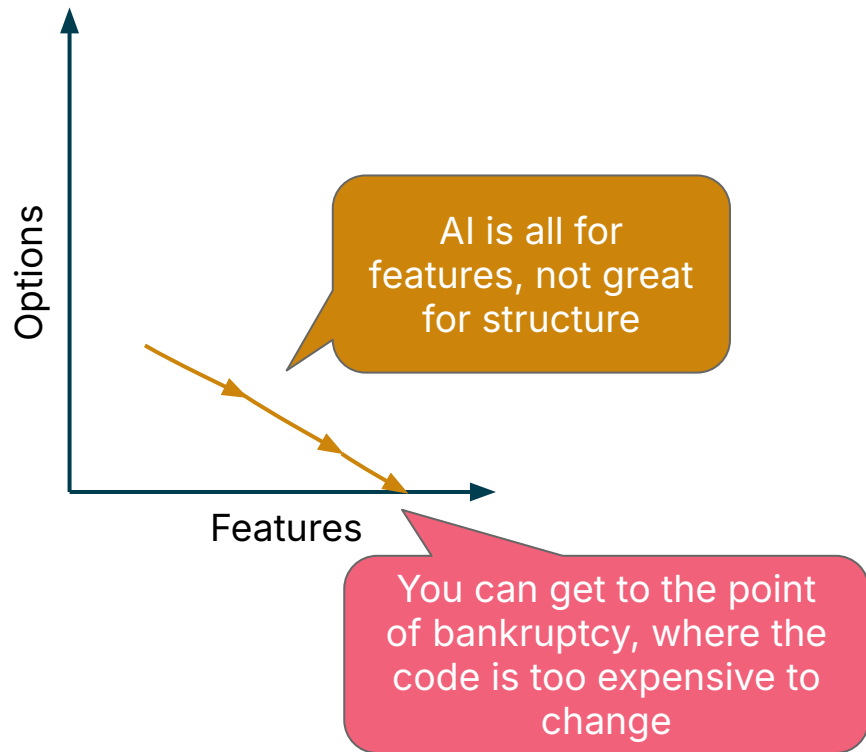
Balance features with ease of change



AI code is hard to change



See Kent Beck,
Augmented coding and design



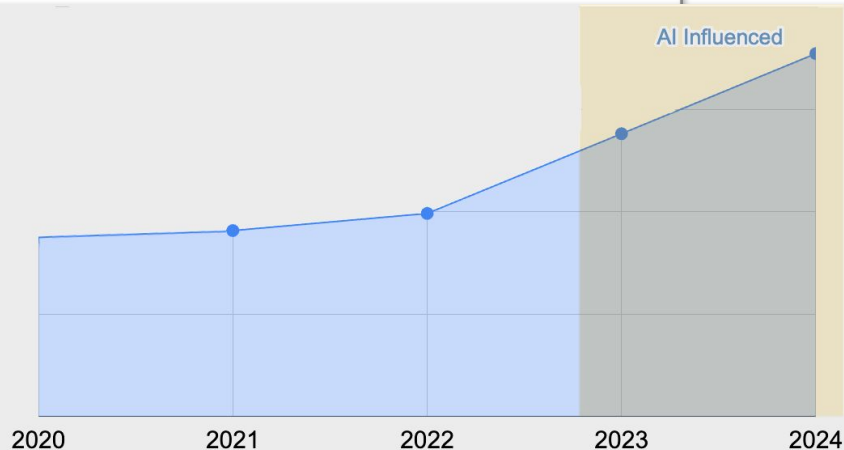
Data about code quality



Coding on Copilot

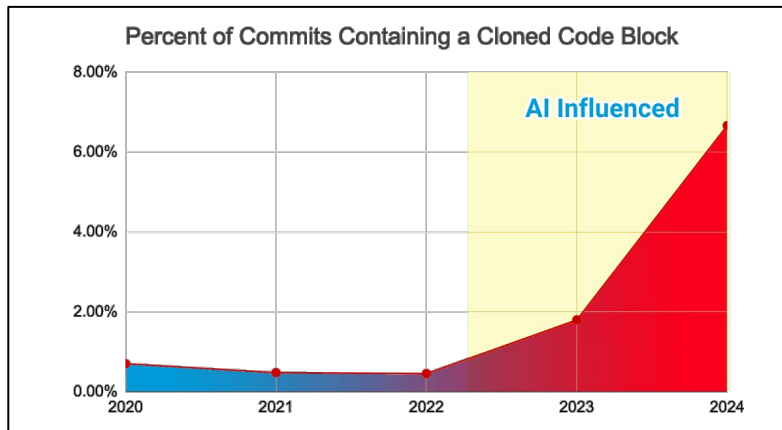
2023 Data Shows Downward Pressure on Code Quality

150m lines of analyzed code + projections for 2024



AI Copilot Code Quality: 2025 Look Back at 12 Months of Data

Emerging trends: 4x more code cloning, "copy/paste" exceeds "moved" code for first time in history. Includes 2025 projections.



See the 2023 and 2025 Gitclear reports

1. Software development is changing
2. State of the art
3. Problems
- 4. A process that works**
5. Skills for the AI-Assisted Future

A process that works

- Plan: rules for the AI
 - Concept – what are we building
 - Architecture and technology
 - Process for the AI
- Execute: rules for the human
 - One prompt, one commit
 - Only commit working code
 - Always review before commit
 - Focused prompts – one prompt, one concern
 - When stuck, roll back
 - When tired, walk away and take a pause

(Inspired by Uberto Barbini's upcoming book)

Plan

I want to build an implementation of microchess, please find the rules here: <https://greenchess.net/rules.php?v=microchess>

The app should allow a person to play against the computer.

It should be implemented

- in Go
- UI in HTML and CSS with bootstrap.
- The app will generate the HTML with templates
- Apply the Model-View-Controller pattern

for long-term maintainability, we waive these internal quality requirements

- We want the code to be as simple as possible
- It must be verified by automated tests so that we can quickly check that it's correct
- All code should clearly express its intent, making future extensions and maintenance easier
- We want to avoid duplication of logic to reduce the risk of inconsistencies

You must help me in writing a detailed but concise document that can be used by AI assistants for coding

Ask me clarifying questions, one at a time, until you have a clear understanding. I prefer yes/no questions



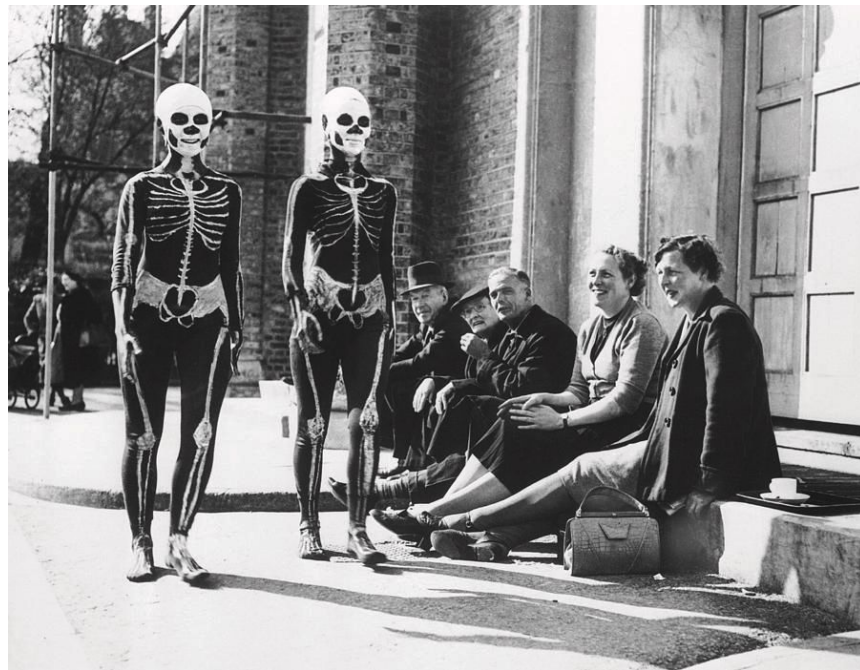
Step 1 - walking skeleton

It is a minimal end-to-end implementation of the intended architecture

UI → backend → DB

You can run it and verify it works!

*write a walking skeleton that shows a hello world in the browser.
Please write also a document with the design and the role of each source file as you go*



Demo 1st prompt: walking skeleton

RULESET.md

read the ruleset do you understand them?

Restore checkpoint

Thought for 5 seconds

Yes, I understand the microchess ruleset. It's played on a 4x5 board with 5 pieces per player (King, Rook, Bishop, Knight, Pawn). Standard chess movements apply with modifications: no castling, no pawn double moves, pawns can promote, and players win by checkmate or capturing the opponent's king. Stalemate results in a draw.

The document also outlines the MVC architecture for implementation in Go with Bootstrap UI, including component specifications, interaction patterns, and quality requirements.

RULESET.md

write a walking skeleton that shows a hello world in the browser. Please write also a document with the design and the role of each source file as you go

Restore checkpoint

Thought for 6 seconds

I'll create a walking skeleton for the microchess application with a "Hello World" page.

Microchess


Hello Class!

Welcome to the Microchess walking skeleton.

© 2025 Thoughtworks | Restricted

61

Demo 2nd prompt: show the board




 RULESET.md

read the rules and confirm they are understood

Restore checkpoint

Thought for 5 seconds

I've read the RULESET.md and understand the requirements for implementing the microchess application:

 Image  RULESET.md  Internal


We need to show the microchess initial position in place of the hello world. Please ask me clarifying questions and then propose a plan. Don't code yet, think first!



Restore checkpoint

Thought for 5 seconds

I need to understand a few things about your current implementation before proposing a plan:

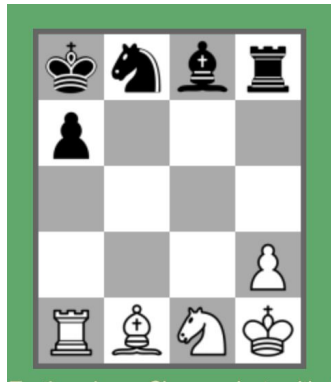
1. Where is the current "hello world" view located? (Which specific file?)
2. What templating system are you using with Go (standard html/template, another library)?
3. Are you planning to use images for chess pieces (like in your example) or text/Unicode characters?
4. Do you want the initial board to be interactive already or just a static display for now?
5. Has the model for representing the chess board/pieces already been implemented?

 ...

 RULESET.md  Internal

1. its in `internal/view/templates/index.html`
2. html/template
3. text/unicode
4. static
5. no

Restore checkpoint



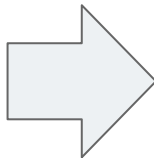
Adding the above image to the prompt

Demo 2nd prompt: show the board

Does this plan align with your expectations? Should I make any adjustments before proceeding with implementation?

[RULESET.md](#) [internal](#)

ok go



[RULESET.md](#) [internal](#)

please update the project documentation

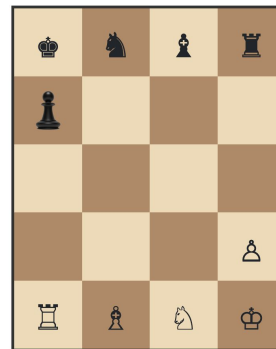
[Restore checkpoint](#)

Thought for 1 second

Read README.md

Now I'll update the README.md to reflect the current state of the project with the implemented chessboard:

Microchess

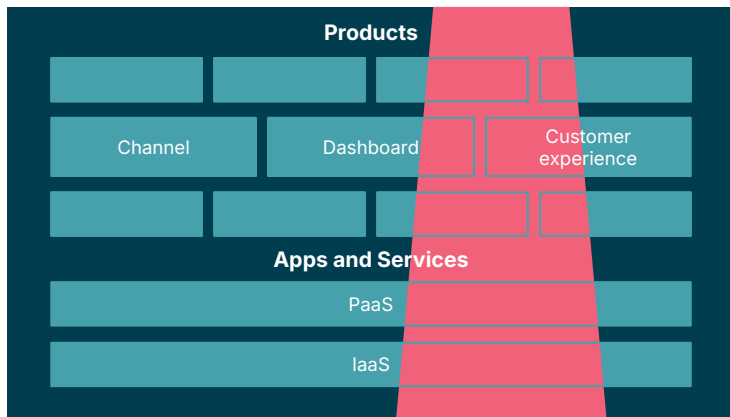


Game ready to play!

Vertical slices



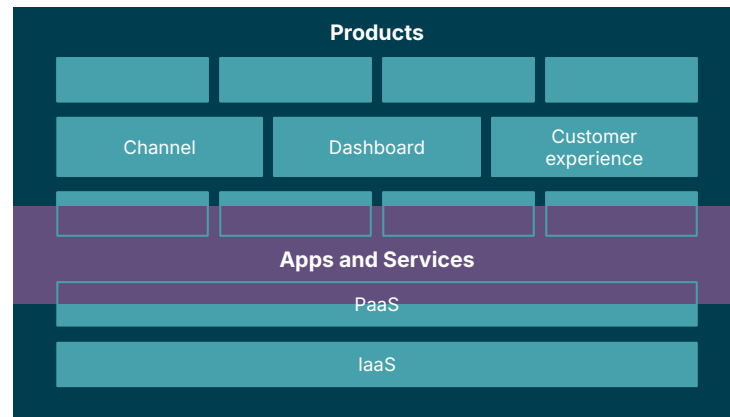
Vertical slice



✓ Desirable ✓ Usable ✓ Functional



Horizontal slice

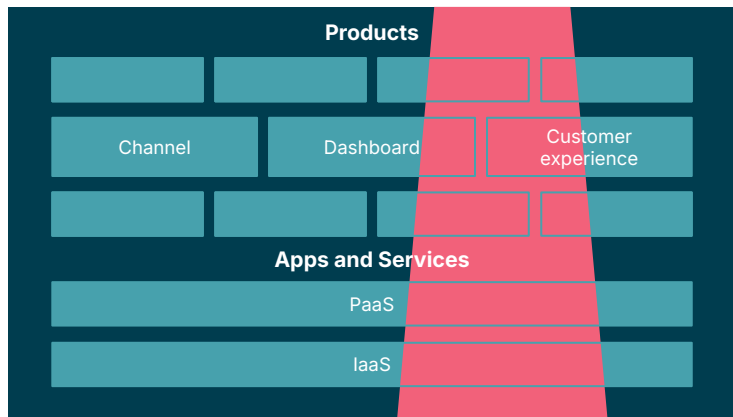


✗ Desirable ✗ Usable ✓ Functional

Break large changes in small tasks



Vertical slice

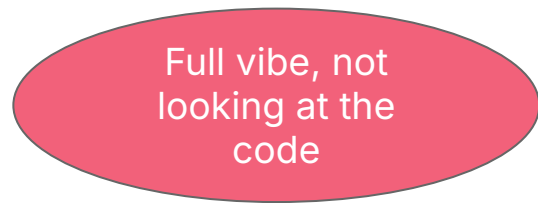


✓ Desirable ✓ Usable ✓ Functional

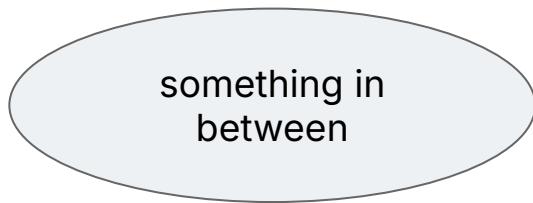
Every task should be a vertical slice

⇒ *You can demo it and verify it works!*

There's a place for everything



Weekend projects
Quick demos
Proofs-of-concept
Throwaway experiments
Visual CSS changes



Production code

Use your judgement!

You are responsible for all the code you push to production

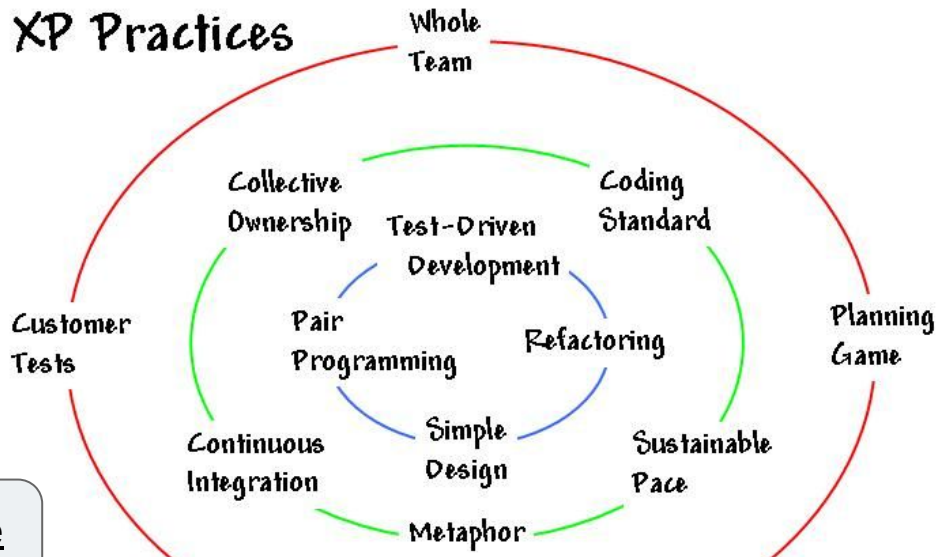
Circling back to Extreme Programming

What works well with AI assistants?

- Have a running app from day 1
- Small working increments
- Continuous integration
- Continuous testing
- Continuous attention to code design

AI-Friendly code
design

XP Practices



Volume 32 | April 2025

Technology Radar

An opinionated guide to today's technology landscape

Circling back to waterfall?

What works well with AI assistants?

Start with clear docs

- requirements
- design
- code documentation

Written and continuously updated by AI

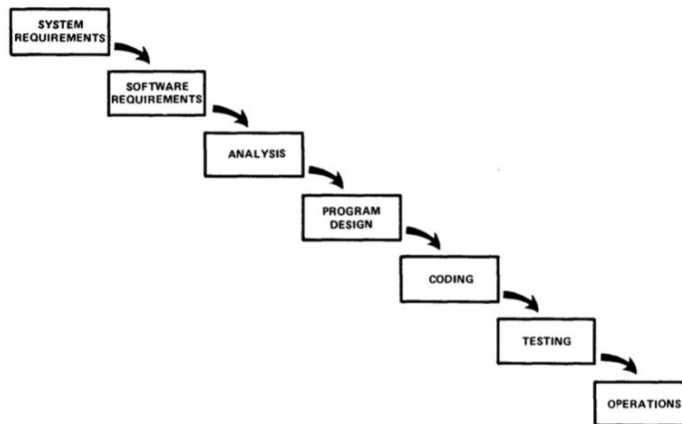


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

Circling back to waterfall?

What works well with AI assistants?

Start with clear docs

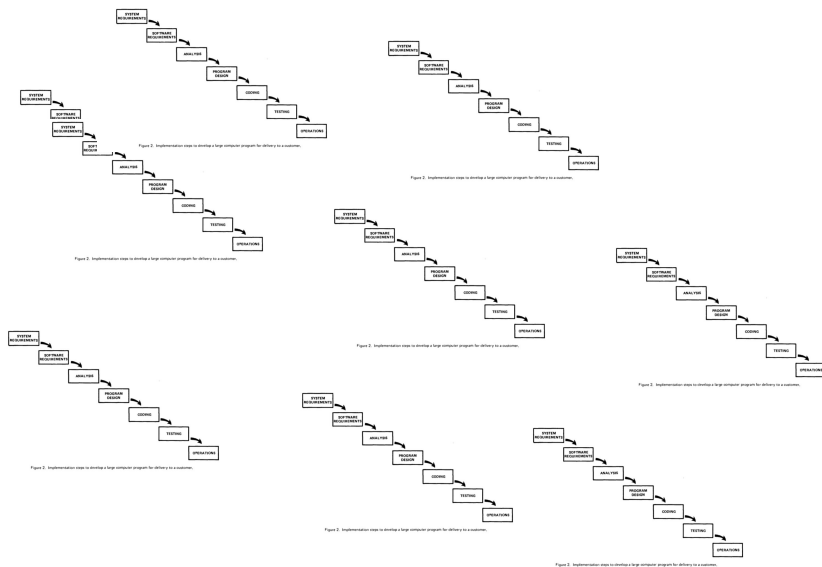
- requirements
- design
- code documentation

Written and continuously updated by AI

In cycles of ~15 minutes!



Harper Reed

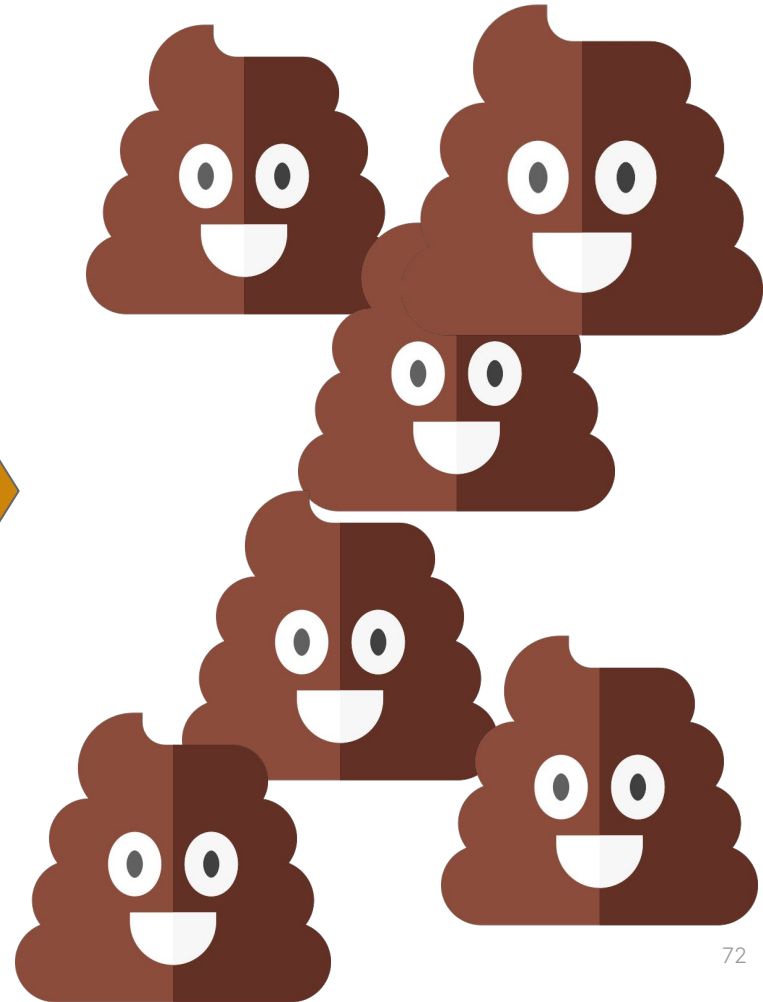
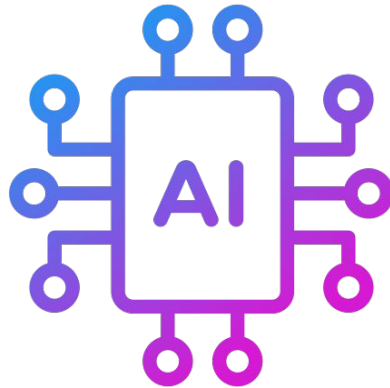


It's Waterfall in 15 minutes!

1. Software development is changing
2. State of the art
3. Problems
4. A process that works
5. Skills for the AI-Assisted Future

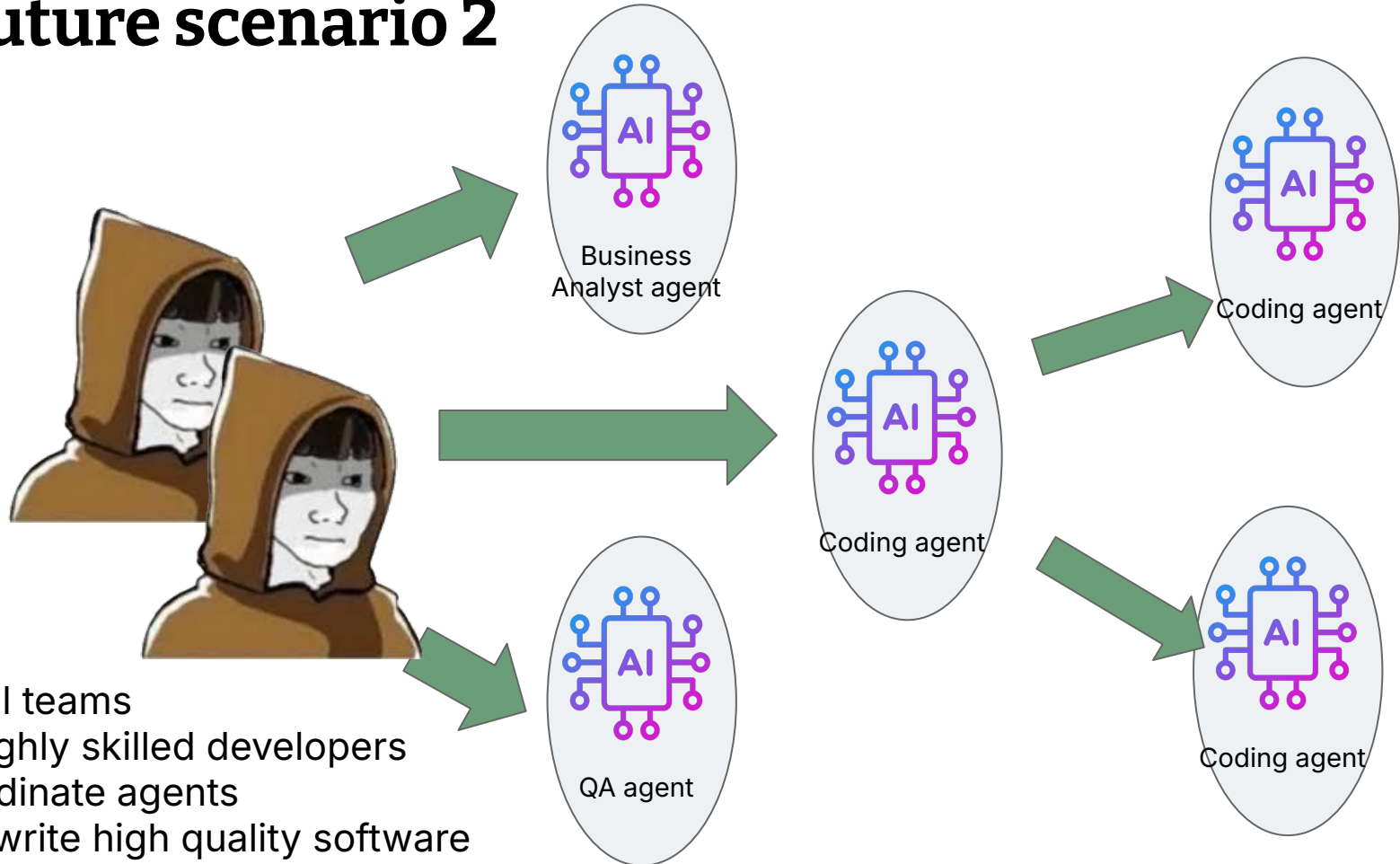
What is the future of software?

Future scenario 1



Low-skilled developers create tons of unsecure, buggy software

Future scenario 2





Thoughtworks

674,555 followers

1w • Edited •

Will AI replace developers?

Learn from our experts why the role of the developer remains crucial to innovation and productivity, even as AI continues to transform the software development lifecycle.

Our next edition of Perspectives launches soon! Subscribe now to receive it straight to your inbox, and be the first to receive practical insights from [Martin Fowler](#), [Birgitta Boeckeler](#) and [Alessio Eros F.](#)

👉 <https://ter.li/b5415p>

You need to be a reasonably skilled developer to detect if the tool is giving you the right kind of output



👍👍👍 You and 219 others

5 comments • 33 reposts

The threat to junior developers

1. Simple tasks can now be delegated to AI
2. Companies need senior developers to direct and validate AI
3. Demand for juniors decreases

⇒ **You better acquire senior developer skills quickly!**



The Death of the Junior Developer by Steve Yegge

Get experience any way you can

- **Contribute to open source projects**
 - How to get started with open source projects
- **Build hobby projects**
 - With AI you can build stuff quickly
- **Take your classes seriously**
 - Learn more than strictly required to pass
 - Algorithms
 - Data structures
 - Discrete math,
 - Operating systems,
 - Compilers,
 - Databases,
 - Networks

Learn how systems work

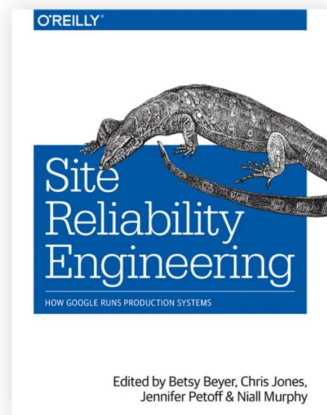
- Understand how every system in a computer works
- Understand how cloud services work
- Read the docs
- Then **read the design docs**

Learn about software design (aka architecture)

- Learn about **coupling** and **cohesion**
- Follow Kent Beck on Substack
- Read
 - *Tidy first?* by Kent Beck
 - *Understanding The Four Rules of Simple Design* by Corey Haines
 - *A software design philosophy* by John Ousterhout
 - *Design Patterns*
 - *Ports & Adapters (Hexagonal) Architecture* by Alistair Cockburn
 - *Implementing Domain-Driven Design* by Vaughn Vernon
 - *CUPID* by Dan North
 - *SOLID* by Robert Martin
 - *Fundamentals of Software Architecture*
 - *Software Architecture: the Hard Parts*
 - [refactoring.guru](#)
 - [tddbuddy.com](#)
 - ... do your own exploring!

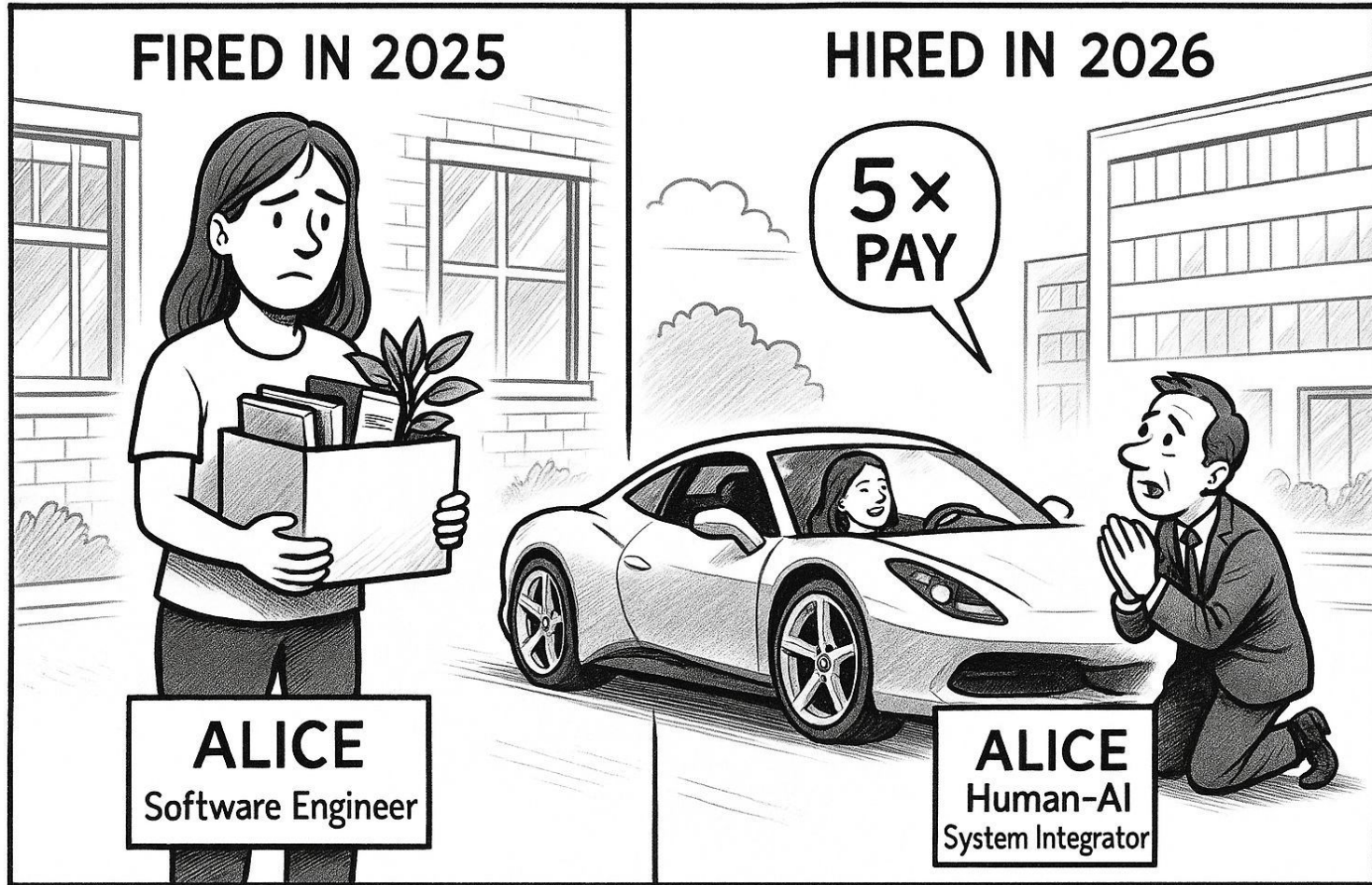
Learn about operations

"One of the key things that differentiates a senior developer from a junior one, ..., is *Operations experience*" – Steve Yegge



"Read it cover to cover" – Steve Yegge

It's available online for free



Thanks for listening!

Matteo Vaccari

Technical Principal

matteo.vaccari@thoughtworks.com

