

THE NEW ERA OF SOFTWARE DEVELOPMENT

Coding with AI

New Software Development Workflow

Luong NGUYEN
luong.nguyen@montimage.eu

78.5% of developers now use AI tools

December 2025 • Stack Overflow Developer Survey

History of AI Coding Assistants

From Tab Autocomplete to Conversational AI (2018–2022)

2018

Tab Autocomplete Era Begins

Tabnine launches — first AI code completion using ML for single-line suggestions

Impact: Developers get syntax-aware predictions based on code patterns

2021

GitHub Copilot Technical Preview (June 29)

Powered by **OpenAI Codex** (GPT-3 variant trained on code). Multi-line completions.

Impact: AI understands intent, not just syntax. 30-40% of code from AI suggestions.

2022

The ChatGPT Inflection Point (Nov 30)

Copilot goes GA (\$10/mo). ChatGPT launches — developers start asking for code snippets conversationally.

New workflow: Copy-paste from chat → IDE. Trade integration for sophisticated generation.

The Rise of AI-Native IDEs & Agents

From Single Files to Full Codebase Autonomy (2023–2025)

2023

⚡ Cursor Launches — AI-Native IDE Revolution

4 MIT students rebuild VS Code around AI. **Composer** enables multi-file editing.

Also: GPT-4 (Mar), Claude 2 with 100K context (Jul), Aider CLI tool launches

2024

🧠 Claude 3.5 Sonnet Becomes Coding Champion (Jun 20)

Claude 3 family (Mar) → 3.5 Sonnet leads all benchmarks. **Windsurf** launches (Nov) — first "agentic IDE"

MCP Protocol announced (Nov). Cursor reaches \$9.9B valuation, 1M+ users

2025

🏠 The Agentic Era + Model Wars

Claude Code (Feb) — terminal-based agentic coding. **Gemini 3** (Nov), **Opus 4.5** (Nov), **GPT-5.2** (Dec)

Opus 4.5: 80.9% SWE-bench | GPT-5.2: 80% SWE, 93% GPQA | Gemini 3: 1501 Elo, 1M context

💡 The IDE→CLI Shift: Agents orchestrate git, tests, builds — terminals > text editors for automation

Sources: Anthropic, GitHub, TechCrunch

LLM Model Releases That Shaped Coding

Each generation brought new capabilities to AI-assisted development

AUG 2021

OpenAI Codex

Powers GitHub Copilot launch. First production code model.

NOV 2022

GPT-3.5 / ChatGPT

Conversational coding begins. Copy-paste workflow emerges.

MAR 2023

GPT-4

Major reasoning leap. Complex architecture decisions possible.

JUL 2023

Claude 2

100K token context. Analyze entire codebases at once.

JUN 2024

Claude 3.5 Sonnet

Beats Opus at lower cost. Becomes top choice for coding.

NOV 2025

Gemini 3 Pro

1501 Elo LMArena. 76.2% SWE-bench. 1M token context.

NOV 2025

Claude Opus 4.5 ★

80.9% SWE-bench. Best for coding, agents, computer use.

DEC 2025

GPT-5.2 🔥

80% SWE-bench. 93.2% GPQA. Best abstract reasoning.

ONGOING

DeepSeek V3.2

Open-source competitor. 10x cheaper than proprietary.

Key pattern: Context windows grew (4K→200K), reasoning improved • Sources: Anthropic, OpenAI, Google

How AI Transforms Development Methodologies

From Waterfall (1970s) → Agile (2001) → AI-Native (2024+)

● Waterfall Era

Sequential phases: Requirements → Design → Code → Test → Deploy
Rigid, plan-driven, change-resistant

● Agile Era (2001)

Iterative sprints, working software, responding to change
TDD emerges: Write tests first, then code

● AI-Assisted (2022–2024)

Same workflow + AI writes small chunks of code
Copilot for completions, ChatGPT for snippets

● AI-Native (2025+)

AI supports entire lifecycle: plan → architect → code → test → deploy
CLI-first, specification-driven, agentic execution

THE PARADIGM SHIFT

TDD → Specification-Driven Development

AI works best with clear upfront specifications. Detailed requirements → better generated code.

Traditional TDD

1. Write failing test
2. Write code to pass
3. Refactor

⚠️ <10% adoption

Spec-Driven + AI

1. Define specification
2. AI generates tests
3. AI generates code

✓ *Tests become free*

Kent Beck (TDD creator): "TDD is a superpower when working with AI agents"

Sources: Pragmatic Engineer, InfoWorld, Tabnine

Emerging AI-Native Methodologies

Specification-first frameworks designed for AI collaboration

GITHUB

Spec Kit

Open-source toolkit for specification-driven development (Sep 2025)

- 1 **Specify** — Define what & why
- 2 **Plan** — AI generates tech plan
- 3 **Tasks** — Break into chunks
- 4 **Implement** — Execute & review

💬 MY EXP

Quite slow for simple tasks

FISSION

OpenSpec ★

Brownfield-first approach for existing codebases (Nov 2025)

LIFECYCLE

Proposal → Review & Apply → Archive

TWO-FOLDER MODEL

openspec/specs/ — current truth
openspec/changes/ — proposals

💬 MY EXP — TOP CHOICE

More natural flow: new feature → proposal → plan → implement. Less documentation overhead.

24K

BMAD Method

★
Multi-agent framework with 19+ AI personas

SPECIALIZED AGENTS

Analyst, PM, Architect, Dev, QA, Scrum Master...

TWO PHASES

1. Agentic planning with personas
2. Context-engineered development

💬 MY EXP

Over-engineered for small tasks. Super slow — took 8h to complete a landing page!



OpenSpec is my current top choice, but it can change depending on the mission • Sources: GitHub, EPAM, Fission Labs

AI Coding Tools: IDE vs CLI

Choose your interface based on workflow and preferences



IDE-Based Tools



Cursor

\$20-200/mo • AI-native IDE, multi-file editing



Windsurf

Free-\$15/mo • Cascade agent, autonomous tasks



Google Antigravity

Free • Gemini 3 powered, agentic dev platform



VS Code Plugins

Copilot, Cline, Continue Dev, Kilo Code, Claude, Codex, Gemini, Codeium, Tabnine



CLI-Based Tools



Claude Code ★

\$20-200/mo • Best agentic coding, 50%+ share



OpenAI Codex CLI

API pricing • GPT-5.2 powered, 24hr autonomous



Warp

Free-\$22/mo • AI terminal with Planning Mode



Aider / Gemini CLI

Free/OSS • Model flexibility, Git-native workflow

IDE: visual editing + context • CLI: git/CI integration + automation • Sources: Anthropic, OpenAI, Google, Cursor, Warp

Choosing Your Tools & Models

Match the right tool to your use case

TOOL BY SCENARIO

Daily coding (VS Code)	→ Cursor or Copilot
Learning to code	→ Copilot Free / Codeium
Complex refactoring	→ Claude Code / Cursor Agent
Budget-conscious	→ Aider + API / Gemini CLI
Enterprise security	→ Tabnine / Windsurf Ent.
AWS development	→ Amazon Q Developer

MODEL BY TASK (Dec 2025)

Claude Opus 4.5 ★ Best coding, agents, computer use. \$5/\$25 per M tokens

80.9% SWE-bench

GPT-5.2 🔥 Best abstract reasoning (53% ARC-AGI-2), 400K context

80% SWE / 93% GPQA

Gemini 3 Pro Best multimodal, 1M context, long-horizon planning

76% SWE / 1501 Elo

Claude Sonnet 4.5 Best cost/quality ratio. \$3/\$15 per M tokens

77% SWE-bench

DeepSeek V3.2 10x cheaper than proprietary, high-volume tasks

Open-source

Opus 4.5 leads coding, GPT-5.2 leads reasoning, Gemini 3 leads multimodal • Sources: Anthropic, OpenAI, Google (Dec 2025)

Setup Your Environment for AI Coding

Essential configuration files that supercharge AI assistance

CLAUDE CLAUDE.md

Auto-read by Claude Code. Place at repo root.

```
# Project Context
Prioritize readability.

## Tech Stack
FastAPI + PostgreSQL

## Commands
Dev: uvicorn app:main
Test: pytest -v
```

Keep under 500 lines

UNIVERSAL AGENTS.md

Vendor-neutral. Works with Codex, Jules, Cursor...

```
# Agent Instructions

## Permissions
Can run tests
Cannot deploy

## Conventions
Type hints required
```

Linux Foundation standard

IDE IDE Rules

Tool-specific configuration files

Cursor Rules

`.cursor/rules/*.mdc`

Copilot Instructions

`.github/copilot-instructions.md`

Pattern-specific rules

Config files = persistent context for better AI suggestions • Sources: Anthropic, GitHub, Cursor

Advanced Setup: MCP & Tool Integration

Connect AI to your entire development ecosystem

MCP Model Context Protocol

"USB-C for AI applications" — standardizes connections between AI and tools

KEY PRIMITIVES

Tools

Functions AI executes

Resources

Data sources to read

Prompts

Reusable templates

Anthropic (Nov 2024) → OpenAI (Mar 2025) → Linux Foundation (Dec 2025)

100+ MCP SERVERS AVAILABLE

GitHub Slack Google Drive PostgreSQL Stripe Puppeteer

FREE

Context7

Fetches current, version-specific documentation into prompts

```
# Just add to any prompt:  
use context7
```

Solves outdated training data problem

💡 For Brainstorming

Use chat interfaces for exploration:

ChatGPT — broad knowledge, plugins

Gemini — Google ecosystem, search

Claude — long context, artifacts

Sources: Anthropic, Upstash, Linux Foundation

Best Practices for AI-Assisted Coding

Maximize productivity while maintaining quality and security

Task Definition

Break large tasks into subtasks with explicit milestones

✗ "Write a sorting function"

✓ "Python quicksort for int lists up to 1GB with error handling, PEP 8"

Planning First

Create execution plans before implementation

"Analyze codebase, create plan.md, ask for approval before each step"

Context Management

Treat config files as living documentation

Use @codebase, @files directives. Include example I/O.

Security Critical

45-48% of AI code contains security flaws

- Never trust AI code blindly — always review
- Verify suggested dependencies exist
- Run SAST/DAST before merging

Quality Workflow

AI generates → Static analysis → Manual review → Tests → Security scan → Merge

Validate test quality — AI can create passing tests that test nothing

Documentation

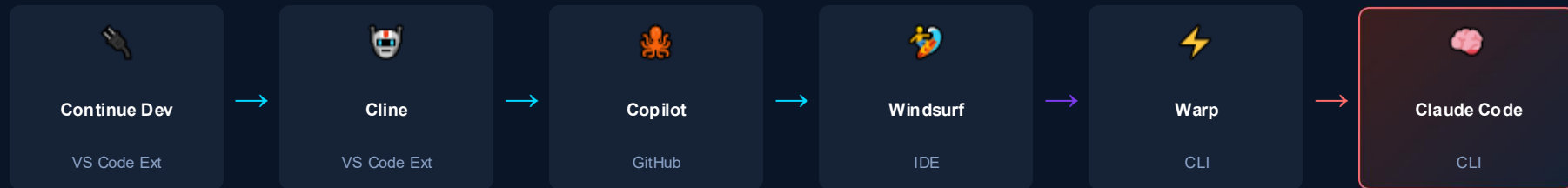
Generate docs from same spec source as code

Living docs that evolve with implementation

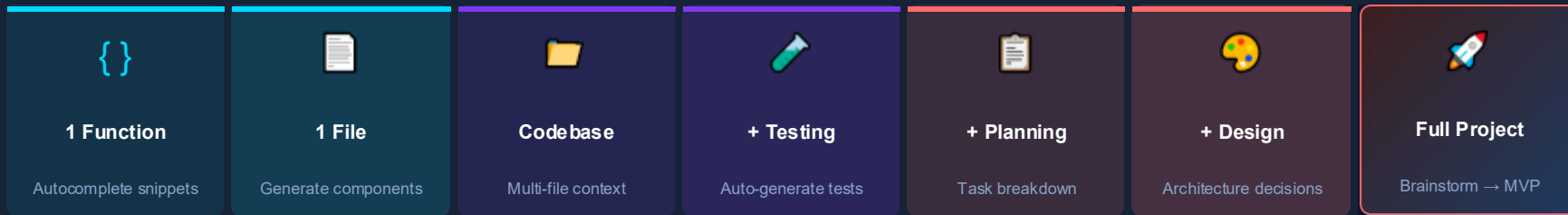
#1 frustration: "Almost right, but not quite" (66%) — review everything • Sources: Codacy, Veracode, SO Survey

My AI Coding Journey

From single functions to full project delivery



How AI Involvement Grew in My Workflow



💡 **Key Insight:** The shift from IDE extensions to CLI tools enabled true agentic workflows — AI moved from assistant to collaborator

Personal experience • Each tool transition expanded what AI could help accomplish

My Current AI Workflow

Structured approach using OpenSpec methodology

🔑 NEW PROJECT (Greenfield)

🏠 EXISTING PROJECT (Brownfield)

📄 OpenSpec Method — For Heavy Tasks (Features / Bug Fixes)

1 PROPOSE

`openspec:proposal`

Describe the feature to implement



2 REVIEW

Review & Update

Manual edit or chat with AI



3 APPLY

`openspec:apply [name]`

Accept plan → AI generates code



4 VERIFY

Review Changes

Code + feature, request mods



5 COMPLETE

`openspec:archive`

Feature done ✓

✅ Why OpenSpec Works

Human reviews plan before code generation, catches mistakes early, maintains control

🔄 Iterative Loop

Steps 2-4 can repeat until satisfied with the implementation quality

📁 Spec-Driven

Proposals stored in specs/ folder, changes tracked, audit trail maintained

OpenSpec: Brownfield-first methodology • Human-in-the-loop at every decision point

What Else Can AI Coding Assistants Do?

Tackling the tasks that no developer wants to do 🤖



Write Documentation

Well-structured & comprehensive

- ✓ README files with badges
- ✓ API documentation
- ✓ Architecture diagrams (Mermaid)
- ✓ Flow charts & sequences
- ✓ User guides & tutorials

🖼️ Screenshot placeholder



Setup CI/CD Workflows

Automation made easy

- ✓ GitHub Actions workflows
- ✓ GitLab CI pipelines
- ✓ Docker configurations
- ✓ Deployment scripts
- ✓ Environment setup

🖼️ GitHub Actions screenshot



Write Unit Tests

Improve code coverage

- ✓ Generate test cases
- ✓ Edge case coverage
- ✓ Mock & stub setup
- ✓ Integration tests
- ✓ Test data generation

🖼️ Screenshot placeholder

💡 **Pro tip:** AI excels at tedious but important tasks — let it handle the boilerplate while you focus on architecture & logic

THE TIME IS NOW

Why You Should Start Today

78.5%

of developers use AI tools monthly

55%

faster task completion with AI

41%

of code is now AI-assisted

11wk

to fully realize AI benefits



The tools are improving faster than adoption

Market: \$4.9B (2024) → \$30.1B (2032)



AI mastery is a skill that compounds

Prompting, workflow adaptation, judgment — all improve with practice