SIILI®

How Generative AI will change the software development lifecycle

12.9.2024

Marko Jaanu

Marko Jaanu works at Siili Solutions Oyj as a Head of Technology and Advisor.

He has worked with software development, architecture, test automation, quality and product management themes for over 25 years consulting various Finnish companies, in various roles.

In Siili, Marko has a responsibility on technology vision, competence development and technology community activities. He also drives the software development excellency for Siili in the areas of AI assisted development, open source, security, architecture and offerings.

Marko has been honored with several excellence awards during his career from project and service delivery as well as consultancy excellence.

https://www.linkedin.com/in/markojaanu/

SIILI®

Siili Solutions

Siili is an **AI-powered development partner** for advanced digital solutions

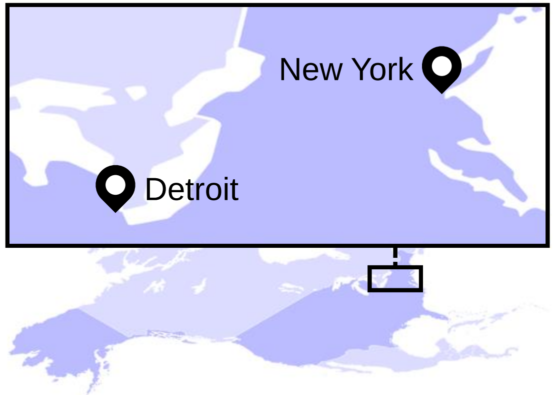
Siili is an international force to be reckoned with

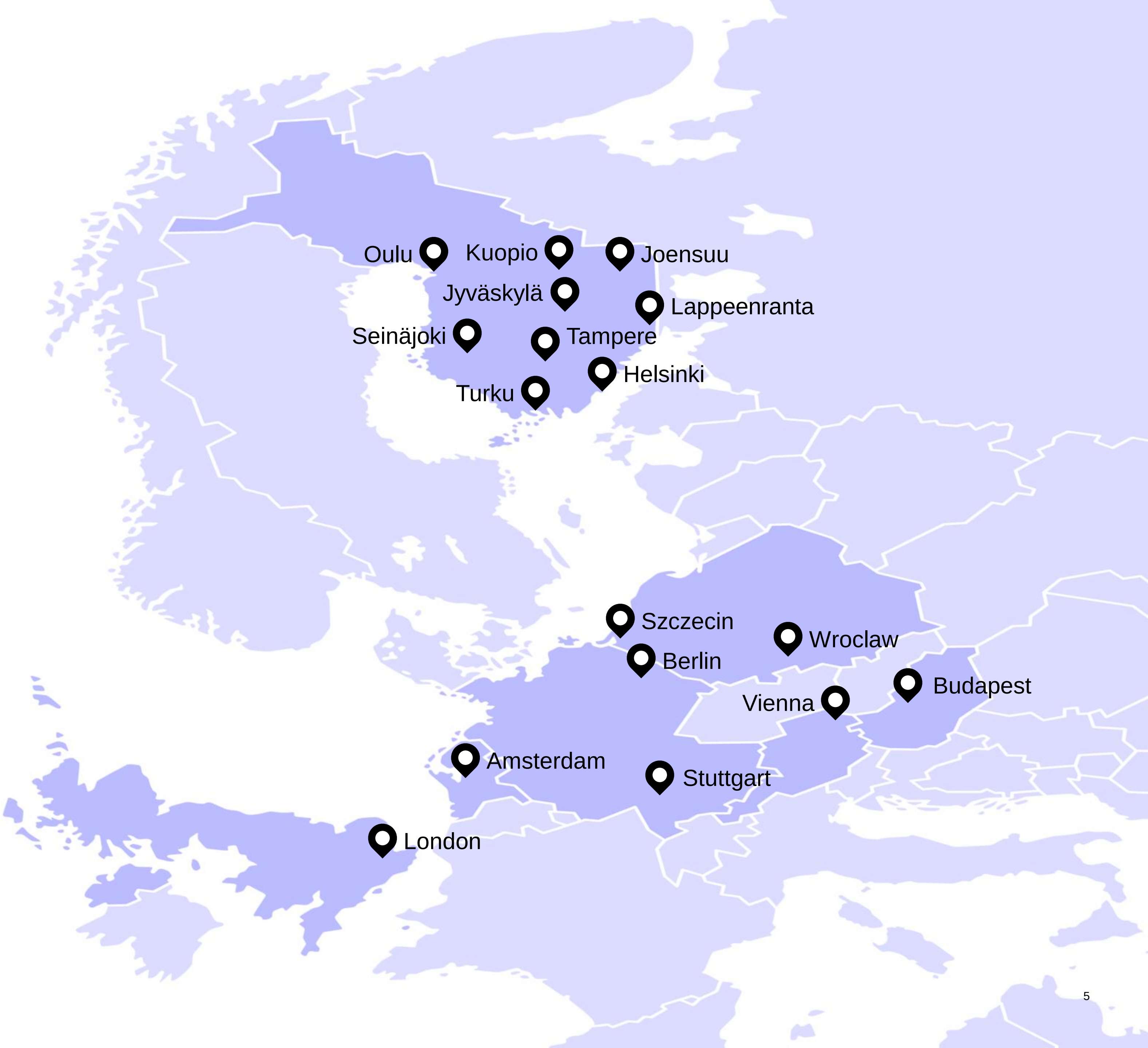
Locations

19

Countries

8





Revenue and profitability

Revenue

122.7 M€ (2023)

118.3 M€ (2022)

Revenue growth

3.7 % (2023)

19.2 % (2022)

Organic revenue growth

0.1 % (2023)

15.2 % (2022)

EBITA

8.4 M€ (2023)

11.6 M€ (2022)

Adj. EBITA %

6.9 % (2023)

9.8 % (2022)

Share of international revenue

26.7 % (2023)

25.2 % (2022)

Explore

What to make real

Design

How to make it real

Data + AI

///

Engineering

Make it real

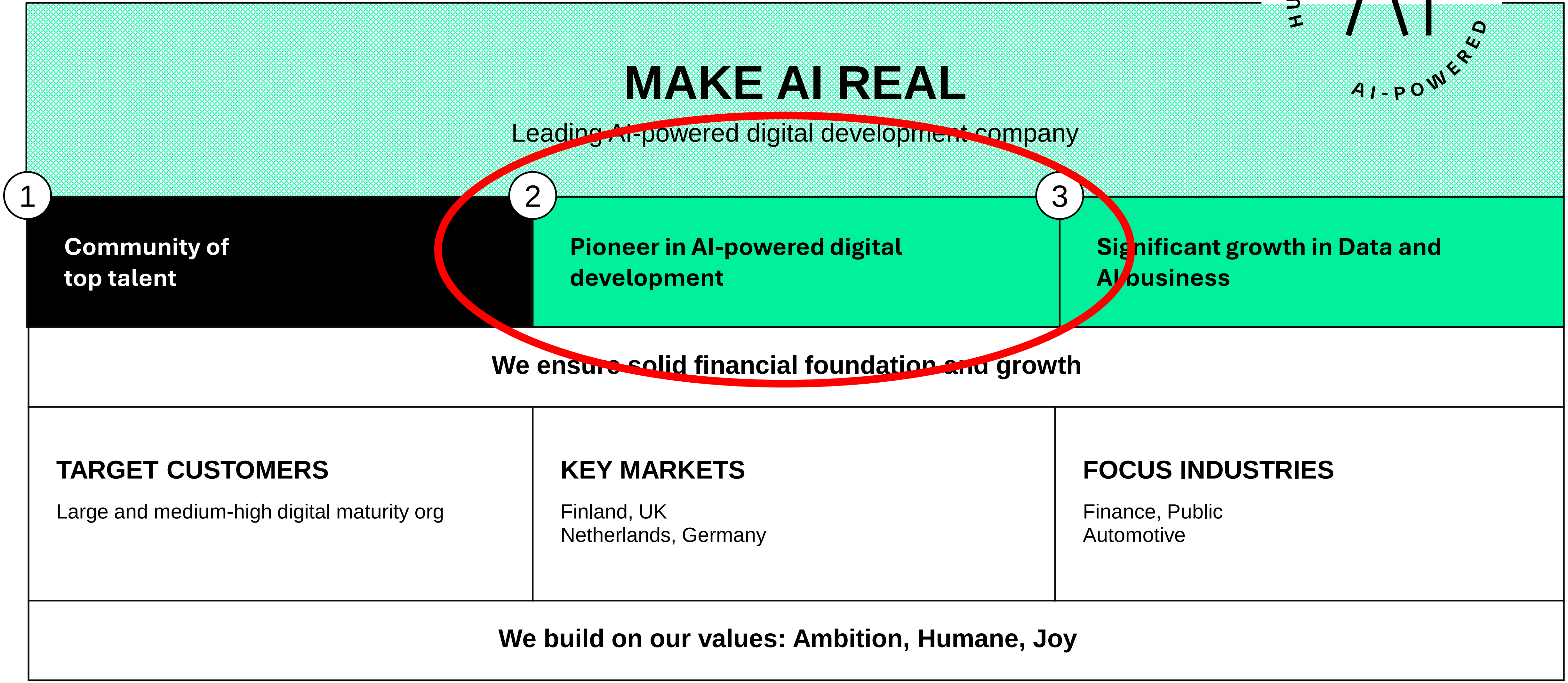
Maintenance

Keep it real

MAKE **IT** REAL → MAKE **AI** REAL

Key building blocks of our strategy

HUMAN CREATIVITY AI-POWERED



MAKE AI REAL

Leading AI-powered digital development company

1 **Community of top talent**

2 **Pioneer in AI-powered digital development**

3 **Significant growth in Data and AI business**

We ensure solid financial foundation and growth

TARGET CUSTOMERS

Large and medium-high digital maturity org

KEY MARKETS

Finland, UK
Netherlands, Germany

FOCUS INDUSTRIES

Finance, Public
Automotive

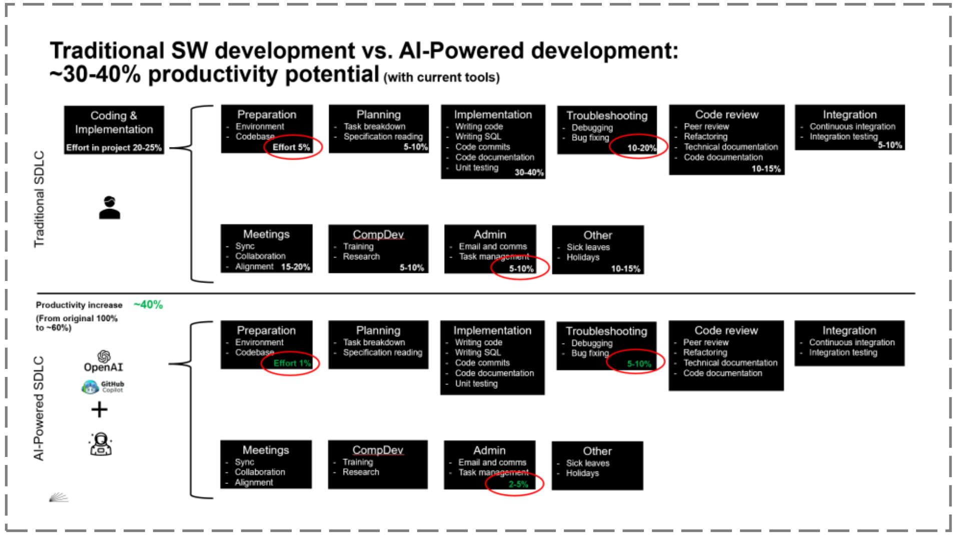
We build on our values: Ambition, Humane, Joy

Pioneer in AI-powered digital development

2

Explore | Design | Data | Engineering | Maintenance

Driver for Siili:	Operational excellence & process innovation
Driver for client:	Faster time to market
Traditional SW development vs. AI-Powered development:	~30-40% productivity potential (with current tools, according to company's view)



Traditional SW development vs. AI-Powered development:
~30-40% productivity potential (with current tools)

Traditional SDLC

Coding & Implementation
Effort in project 20-25%

Preparation
- Environment
- Codebase
Effort 5%

Planning
- Task breakdown
- Specification reading
5-10%

Implementation
- Writing code
- Writing SQL
- Code commits
- Code documentation
- Unit testing
30-40%

Troubleshooting
- Debugging
- Bug fixing
10-20%

Code review
- Peer review
- Refactoring
- Technical documentation
- Code documentation
10-15%

Integration
- Continuous integration
- Integration testing
5-10%

Meetings
- Sync
- Collaboration
- Alignment 15-20%

CompDev
- Training
- Research
5-10%

Admin
- Email and comms
- Task management
5-10%

Other
- Sick leaves
- Holidays
10-15%

Productivity increase ~40%
(From original 100% to ~60%)

AI-Powered SDLC

Preparation
- Environment
- Codebase
Effort 1%

Planning
- Task breakdown
- Specification reading

Implementation
- Writing code
- Writing SQL
- Code commits
- Code documentation
- Unit testing

Troubleshooting
- Debugging
- Bug fixing
5-10%

Code review
- Peer review
- Refactoring
- Technical documentation
- Code documentation

Integration
- Continuous integration
- Integration testing

Meetings
- Sync
- Collaboration
- Alignment

CompDev
- Training
- Research

Admin
- Email and comms
- Task management
2-5%

Other
- Sick leaves
- Holidays

Siili AI trainings

From craftmanship to AI powered development

>25% GenAI **applied** in assignments by 4Q2024

>50% consultants **used** GenAI in assignments by 4Q2024

100% consultants **GenAI trained** by 1Q2024

Training partners

splended

FORGE.

Basics of Gen AI

B

(Bronze certificate)

Studytube basic learning

Focus groups:

- All Siili personnel (Finland +Poland), others voluntary

Participants:

~440+ persons

Training method:

- Studytube course

Content-based learning

S

(Silver certificate)

Intensive courses

Focus groups:

- Lead and senior consultants
- Designers
- Data engineers/scientists
- Cloud engineers/architects

Participants:

-over 100 already

Training method:

- External courses/trainers per group
- Series of online meetings + homework

Hands on competences

G

(Gold certificate)

Learning by doing

Hands on training and coaching

AI based solutions

Focus groups:

- Client work persons and dedicated lead developers

Training method:

- Environment playgrounds, licenses etc.
- Info sessions, Collectives, communities

SIILI®

Research and experiments

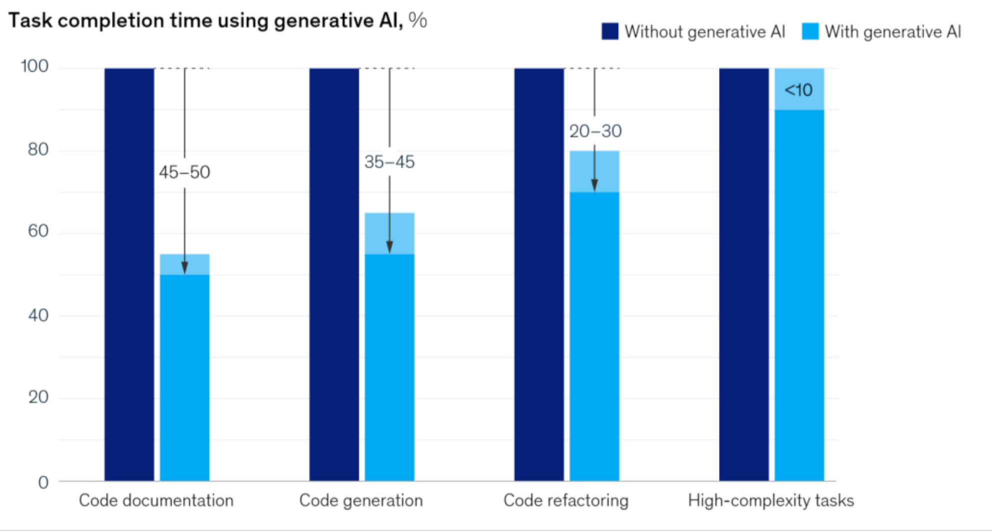
We have already seen up to 50% efficiency improvement in Software Development

According to our experience, software development process includes many routine tasks that can be easily automated with AI

Most potential areas for automation are:

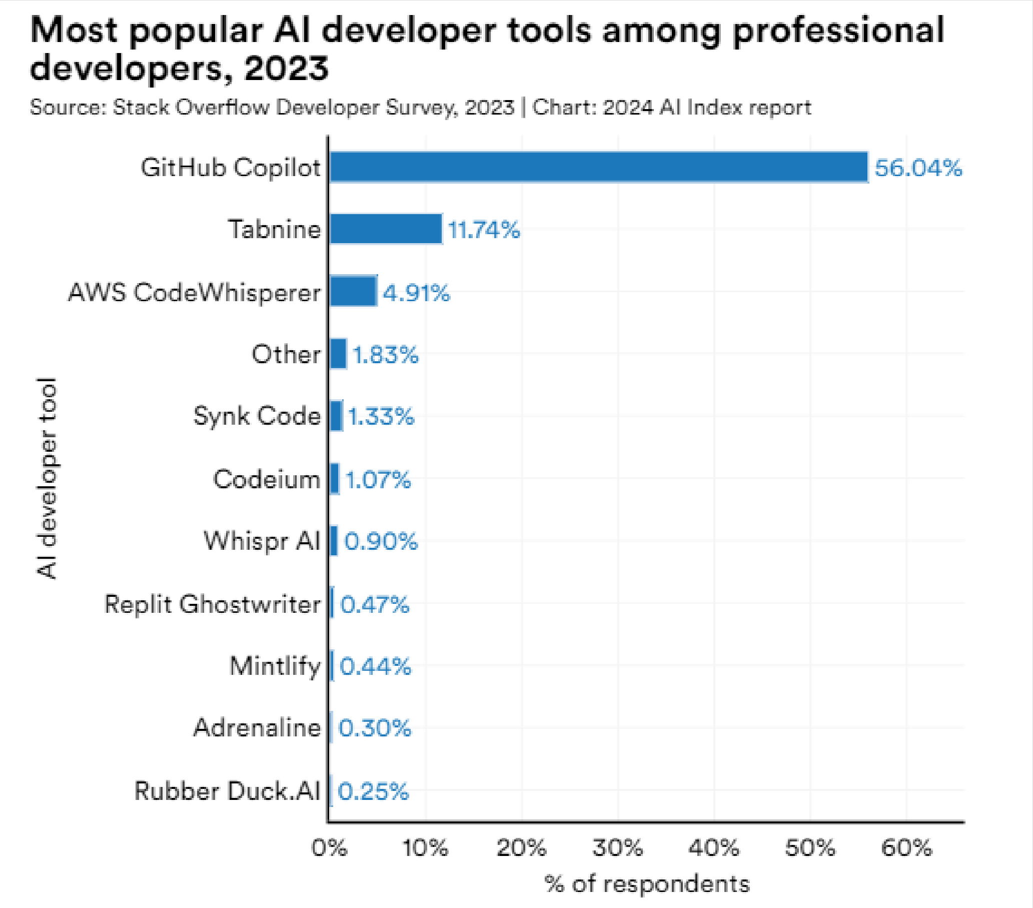
1. Code documentation
2. Code generation
3. Code refactoring
4. High-complexity tasks
   - Debugging and error correction
   - Domain specific requirements and problem solving
   - Timing issues

Research by McKinsey (06/2023)



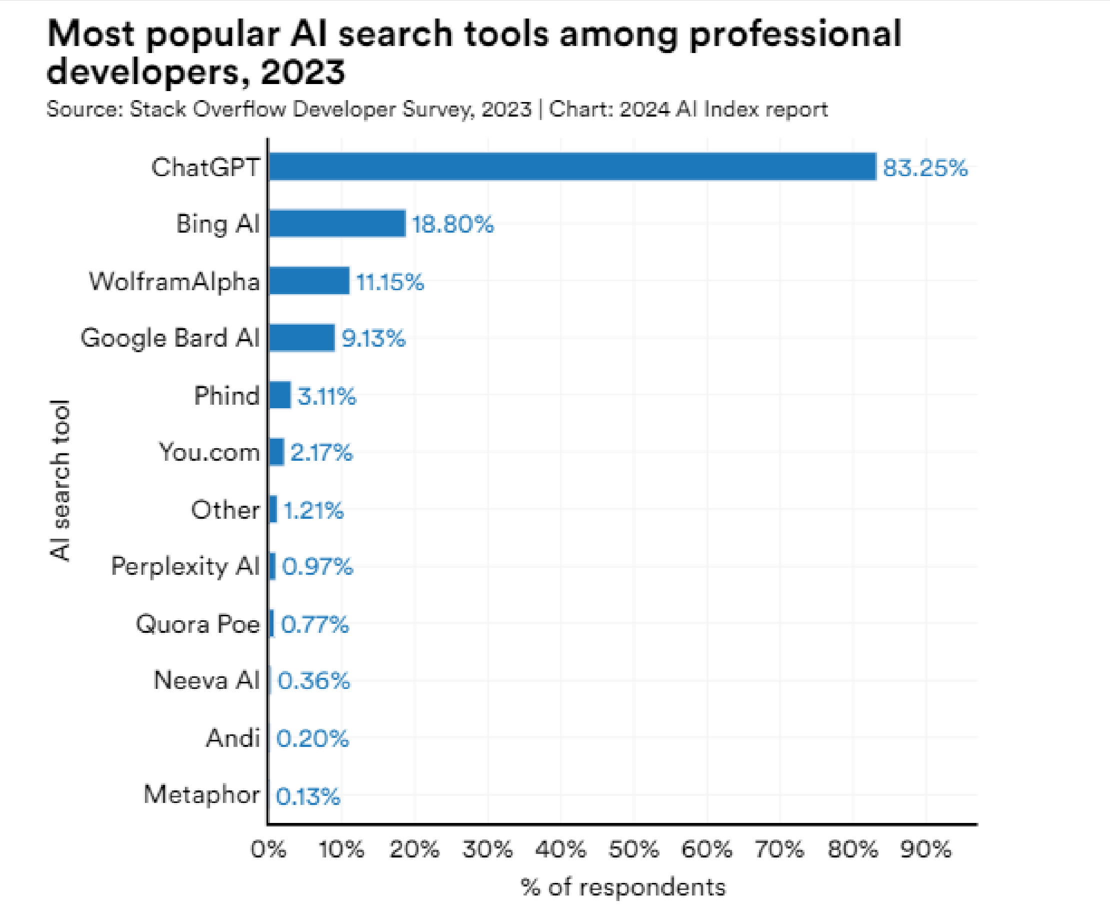
Artificial Intelligence Index Report 2024 by Stanford University

Stack Overflow



Most popular AI developer tools among professional developers, 2023

Source: Stack Overflow Developer Survey, 2023 | Chart: 2024 AI Index report

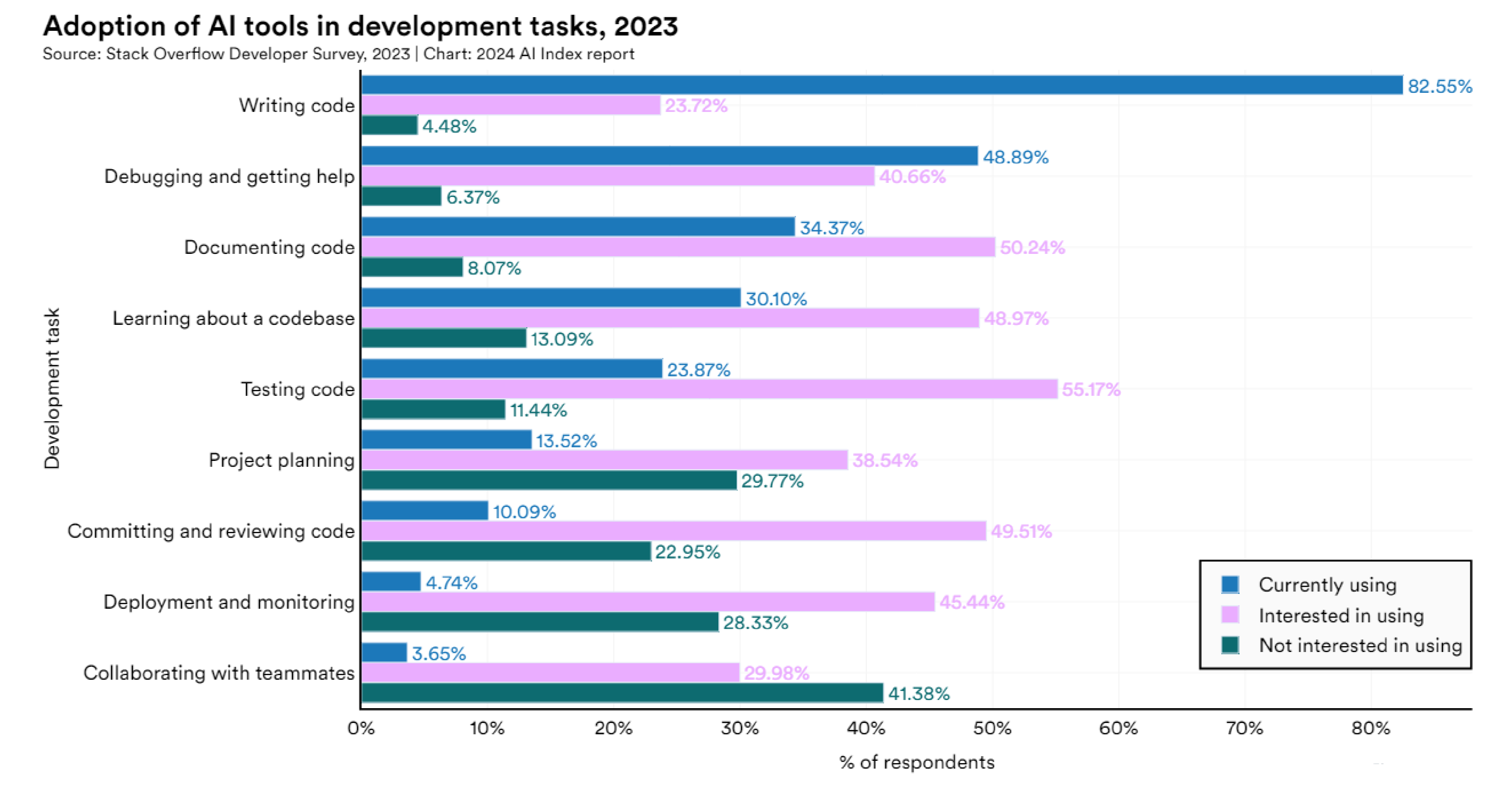


Most popular AI search tools among professional developers, 2023

Source: Stack Overflow Developer Survey, 2023 | Chart: 2024 AI Index report

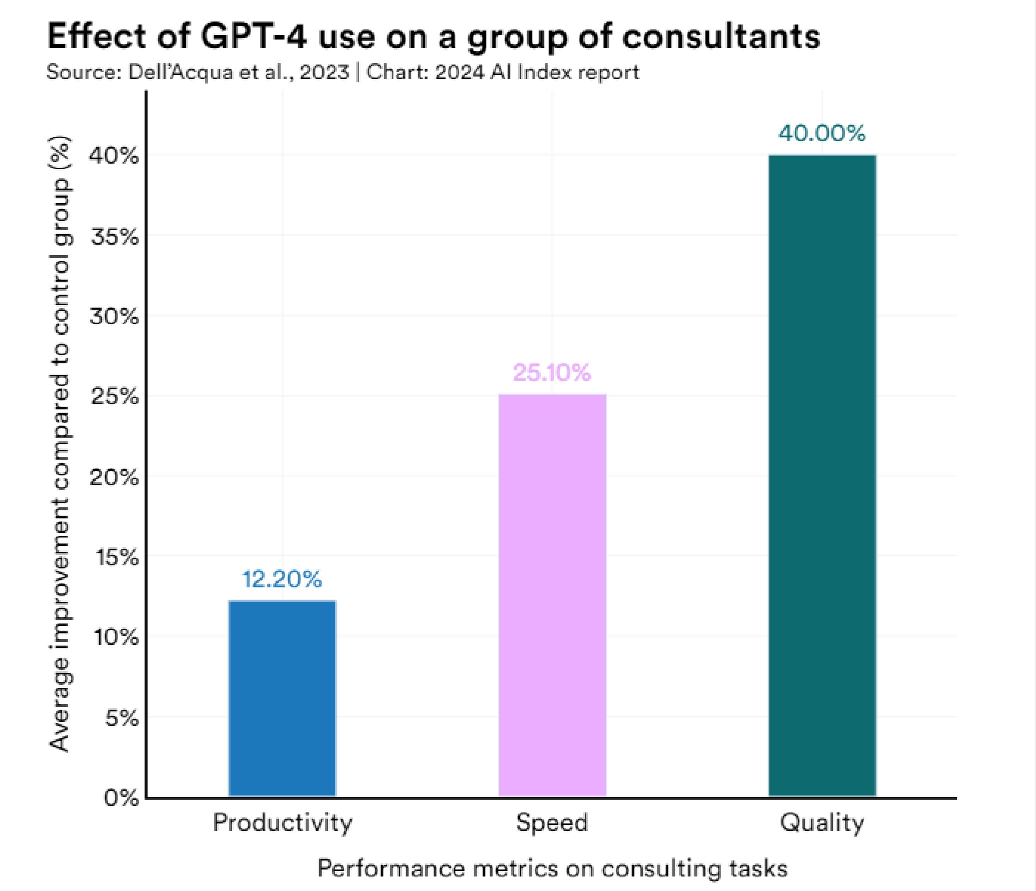
Artificial Intelligence Index Report 2024 by Stanford University

Stack Overflow



Artificial Intelligence Index Report 2024 by Stanford University

Harvard Business School



Effect of GPT-4 use on a group of consultants

Source: Dell'Acqua et al., 2023 | Chart: 2024 AI Index report

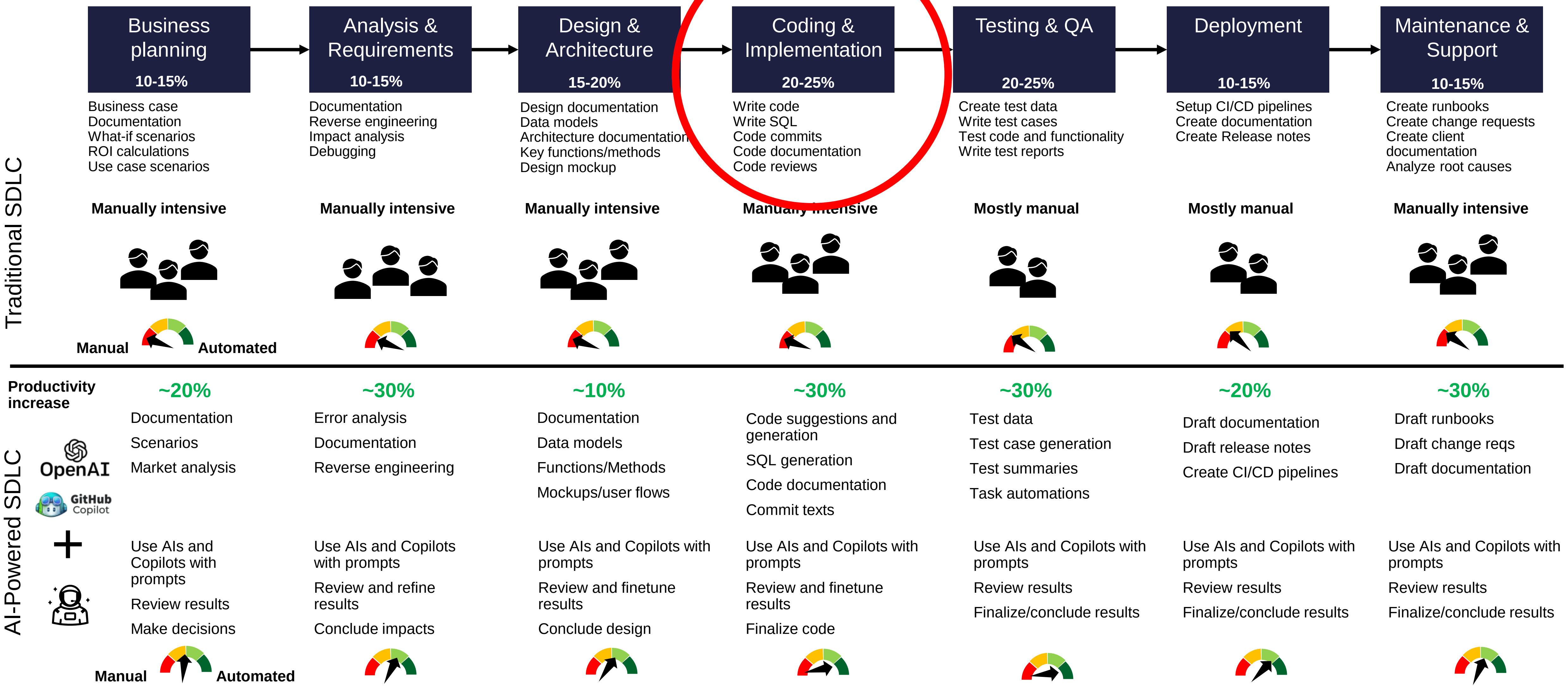
YES I USE CHATGPT FOR CODING FROM TIME TO TIM...
WAKE UP!
imgflip.com

SIILI®

AI Powered development

Traditional SW development vs. AI-Powered development (SDLC)

Phases and activities, waterfall type of project

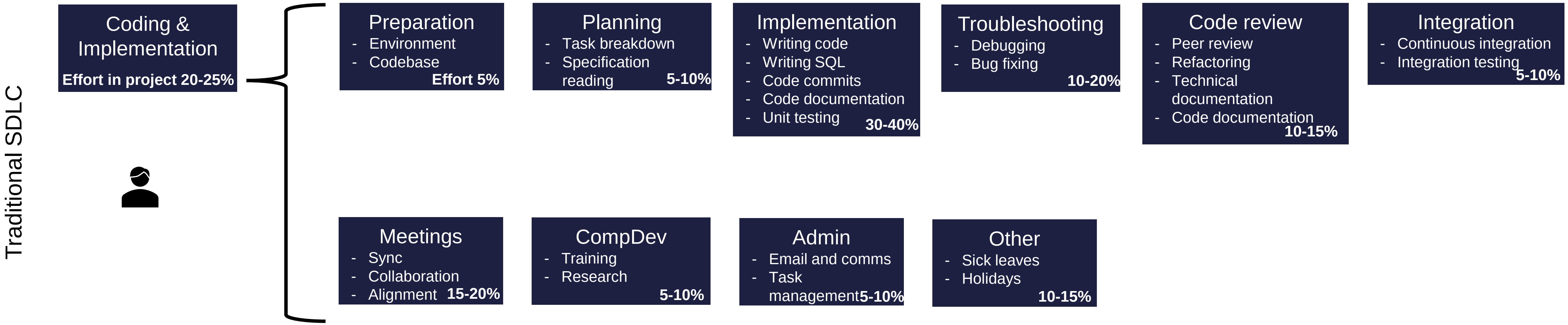


	Business planning	Analysis & Requirements	Design & Architecture	Coding & Implementation	Testing & QA	Deployment	Maintenance & Support
	10-15%	10-15%	15-20%	20-25%	20-25%	10-15%	10-15%
Traditional SDLC	Business case Documentation What-if scenarios ROI calculations Use case scenarios	Documentation Reverse engineering Impact analysis Debugging	Design documentation Data models Architecture documentation Key functions/methods Design mockup	Write code Write SQL Code commits Code documentation Code reviews	Create test data Write test cases Test code and functionality Write test reports	Setup CI/CD pipelines Create documentation Create Release notes	Create runbooks Create change requests Create client documentation Analyze root causes
	Manually intensive	Manually intensive	Manually intensive	Manually intensive	Mostly manual	Mostly manual	Manually intensive
Productivity increase	~20%	~30%	~10%	~30%	~30%	~20%	~30%
AI-Powered SDLC	Documentation Scenarios Market analysis	Error analysis Documentation Reverse engineering	Documentation Data models Functions/Methods Mockups/user flows	Code suggestions and generation SQL generation Code documentation Commit texts	Test data Test case generation Test summaries Task automations	Draft documentation Draft release notes Create CI/CD pipelines	Draft runbooks Draft change reqs Draft documentation
+	Use AIs and Copilots with prompts Review results Make decisions	Use AIs and Copilots with prompts Review and refine results Conclude impacts	Use AIs and Copilots with prompts Review and finetune results Conclude design	Use AIs and Copilots with prompts Review and finetune results Finalize code	Use AIs and Copilots with prompts Review results Finalize/conclude results	Use AIs and Copilots with prompts Review results Finalize/conclude results	Use AIs and Copilots with prompts Review results Finalize/conclude results

Traditional SW development vs. AI-Powered development (SDLC)

Coding & Implementation phase breakdown

Traditional SDLC



Coding & Implementation

Effort in project 20-25%

Preparation
- Environment
- Codebase

Effort 5%

Planning
- Task breakdown
- Specification reading

5-10%

Implementation
- Writing code
- Writing SQL
- Code commits
- Code documentation
- Unit testing

30-40%

Troubleshooting
- Debugging
- Bug fixing

10-20%

Code review
- Peer review
- Refactoring
- Technical documentation
- Code documentation

10-15%

Integration
- Continuous integration
- Integration testing

5-10%

Meetings
- Sync
- Collaboration
- Alignment

15-20%

CompDev
- Training
- Research

5-10%

Admin
- Email and comms
- Task management

5-10%

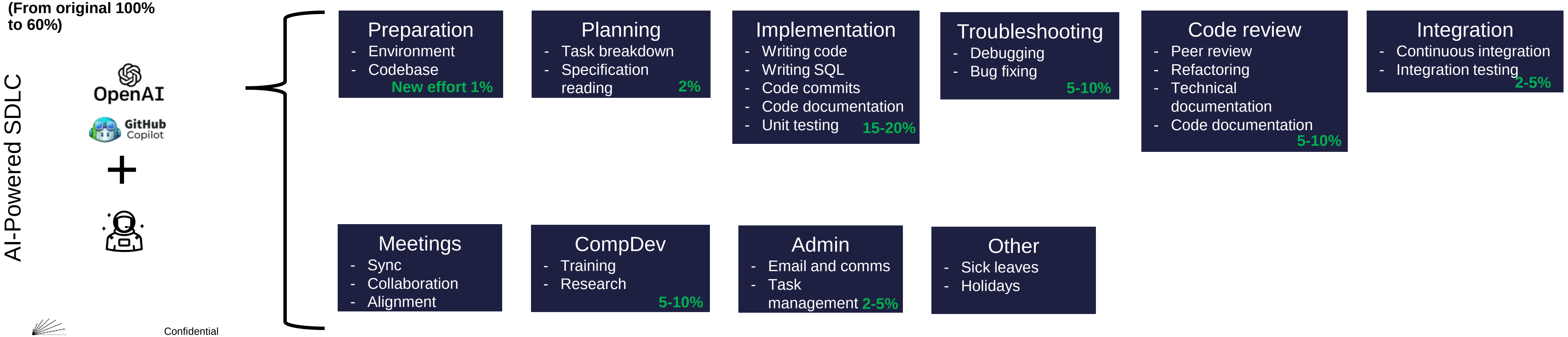
Other
- Sick leaves
- Holidays

10-15%

Productivity increase ~40%

(From original 100% to 60%)

AI-Powered SDLC



Preparation
- Environment
- Codebase

New effort 1%

Planning
- Task breakdown
- Specification reading

2%

Implementation
- Writing code
- Writing SQL
- Code commits
- Code documentation
- Unit testing

15-20%

Troubleshooting
- Debugging
- Bug fixing

5-10%

Code review
- Peer review
- Refactoring
- Technical documentation
- Code documentation

5-10%

Integration
- Continuous integration
- Integration testing

2-5%

Meetings
- Sync
- Collaboration
- Alignment

CompDev
- Training
- Research

5-10%

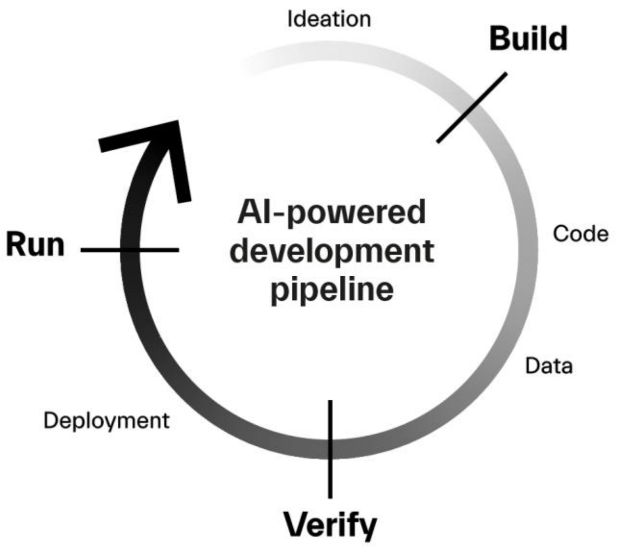
Admin
- Email and comms
- Task management

2-5%

Other
- Sick leaves
- Holidays

Towards a Fully AI-Powered Development Pipeline

Phase	AI Agent	Input	Output
Build	Requirements	User Stories, Market Research, Stakeholder Interviews	Detailed Requirements Document, Updated User Stories, Acceptance Criteria
	Design	Requirements Document, Design Scenarios, Initial Sketches	UX Proposals, Technical Alternatives (ADRs), Prototypes, API Definitions, Database Schemas
	Coding	Finalized Requirements, Existing Code Base, Libraries	Refactored Code, Database Modifications, Automated Unit Tests, New Pull Requests
	Review	New Code Changes, Pull Requests, Review Guidelines	Code Review Approvals, Change Requests, Improvement Suggestions
Verify	Infrastructure	Infrastructure Change Requests, Compliance Policies	Infrastructure Change Approvals, Updated CI/CD Pipelines, Infrastructure as Code (IaC) Updates
	Security	Security Policies, Pull Requests, Threat Models	Security Audit Reports, Identified Vulnerabilities, Remediation Plans
	Data	Data Migration Plans, Database Change Requests, Data Governance Policies	Reviewed Data Migration Scripts, Database Integrity Reports, Data Quality Assessments
	Quality Assurance	Test Data Sets, Defined Test Cases, Change Logs	Executed Automated Test Cases, Impact Assessment Reports, Identified Bugs
Run	Deployment	Deployment Artifacts, Approved Infrastructure Changes, Operational Policies	Successful Deployments, Deployment Notifications, Updated Change Logs, Rollback Plans
	Monitoring	Runtime Monitoring Logs, Audit Logs, System Performance Metrics	Comprehensive Monitoring Reports (covering Data Integrity, Security, and System Performance), Incident Reports, Behavior Analysis
	Maintenance	Maintenance Logs, Current Code Base, Issue Backlogs	Completed Bug Triages, Predictive Maintenance Recommendations, System Updates
	Feedback	User Feedback, Backlog Items, Performance Reports	Iteration Summaries, Sentiment Analysis Reports, Product Improvement Proposals



Tools and AI-Powered pipeline

Prompting tools

1. ChatGPT (several LLMs to be chosen)
2. Claude (several LLMs to be chosen)
   - Artifacts
   - No search
   - Benchmark star
3. Several others which are more or less using LLMs as basis

Developer tools

Integration to IDEs

Code completion, explanation, error debugging, documentation, chatbot etc.

1. Github Copilot
2. Jetbrains AI
3. SuperMaven
   - Github Copilot like approach, but faster
4. Cursor
   - Composer for improved multi-file editing capabilities
5. Codeium
   - Focus on light resources and code completion

AI-assisted Code review

Enhances code quality

1. Coderabbit
   - Intelligent code reviews, analyzing bugs, performance issues etc.
2. CodiumAI PR Agent
   - AI-powered tool designed to enhance the pull request process

Integrated solutions

Coding agents

1. Devin
   - SWE-bench 13.86%
   - A planner agent
   - Coding agents
   - CLI agents
2. OpenDevin/OpenHands (OSS alternative)
   - Best for Python atm
3. Micro-agents (builder.io)
   - Unit tests to meet the reqs
   - Best for TypeScript atm

Workspaces

1. Amazon Q
   - SWE-bench 19.75%
   - Agent activation /dev command
2. Github Copilot Workspaces
   - Integrates closely with GitHub repositories and tools
   - In technical preview still

Agents

Autonomous agents

Iterative loops:

Agents don’t just write code once. They create code, review it, test it, and then improve it. This cycle repeats until the code meets the required standards.

Tool usage:

Agents can use various development tools, such as testing frameworks, linters, and version control systems. They apply these tools as needed throughout the development process.

Current status:

Over 40% of issues solved in swe-bench verified. None of the solutions are quite production ready yet, but as for capabilities they could already take on sizeable chunk of simple tickets.

Issues:

The problems with full autonomy are the exact same as with assistants and “AI pair coders”: context, economy and plumbing.

Present:

We’re not there yet, but there are specialized agents e.g. for code review (CodeRabbit)

Future:

As the core LLMs and the agent solutions keep on evolving, we see it as almost inevitable that these will break through the barrier of usability.

Impact on SWE work:

Obviously getting usable agents is not black/white, but the more they improve, the more the role of developers and development teams change. As an extreme example, a solo “agent hearder” developer could end up doing the work of an entire development team.

Case: Code location

Core problem:

In order to do X, what files do I need to modify and what files do I need to be aware of? For coding agents, this needs to be nearly perfect, but remains a core challenge for assistants too

Impact of larger context windows:

As Claude 3.5 Sonnet can already fit up to 20k lines of code in context, it is technically feasible to even give it entire repositories or large chunks of it. However, it is not economically feasible in bread-and-butter type of tasks, so some form of RAG is required

Vector search:

Create embeddings for the code and find the nearest matches.

Agentless approach:

Feed skeletons of files to a LLM and have it pick out the interesting parts

SLM→LLM:

Have a small, fast and cheap language model shift through the entire codebase and pick the relevant parts for LLM to process

RepoGraph:

Build a graph representation of the repository and its internal dependencies & further populate with context

SIILI®

Human

Leaving the inner loop

Goal:

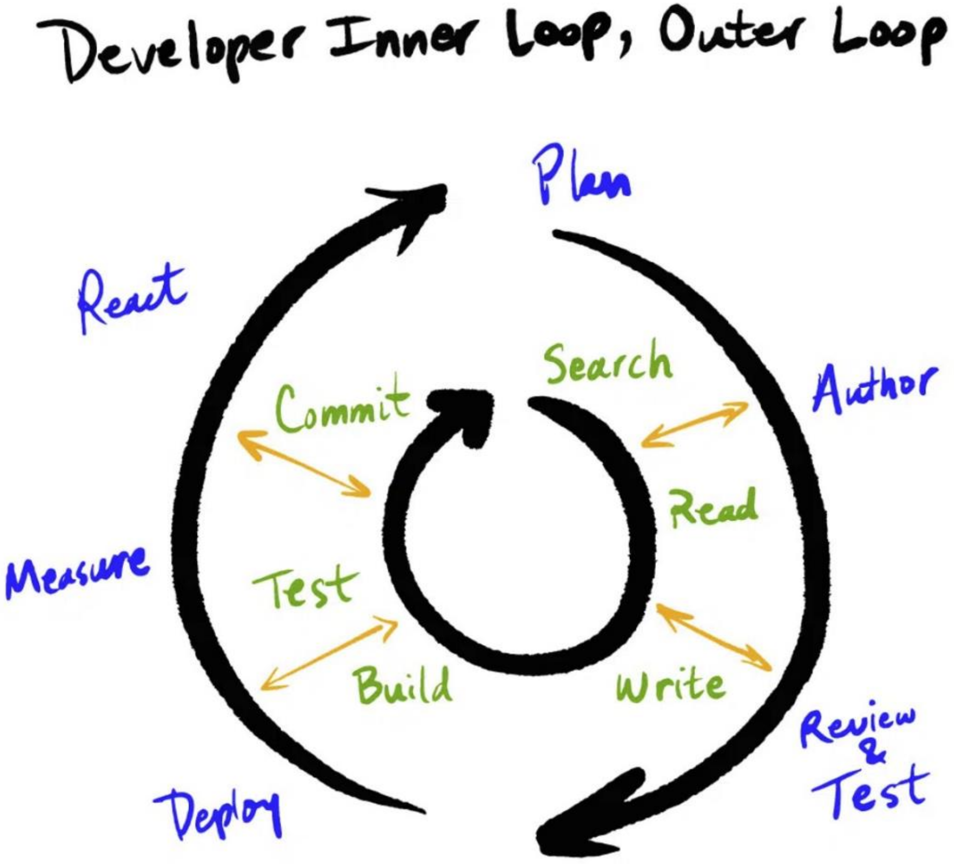
Increasingly agentic workflows, where instead of copy-pasting back and forth, you get the code that is tested, and it actually works.

TDD:

While TDD has its merits, it’s in practice usually too much of a good thing. However, it’s a great fit for agentic workflows. Write the tests, write the code and make sure all tests (including the existing ones) pass before returning.

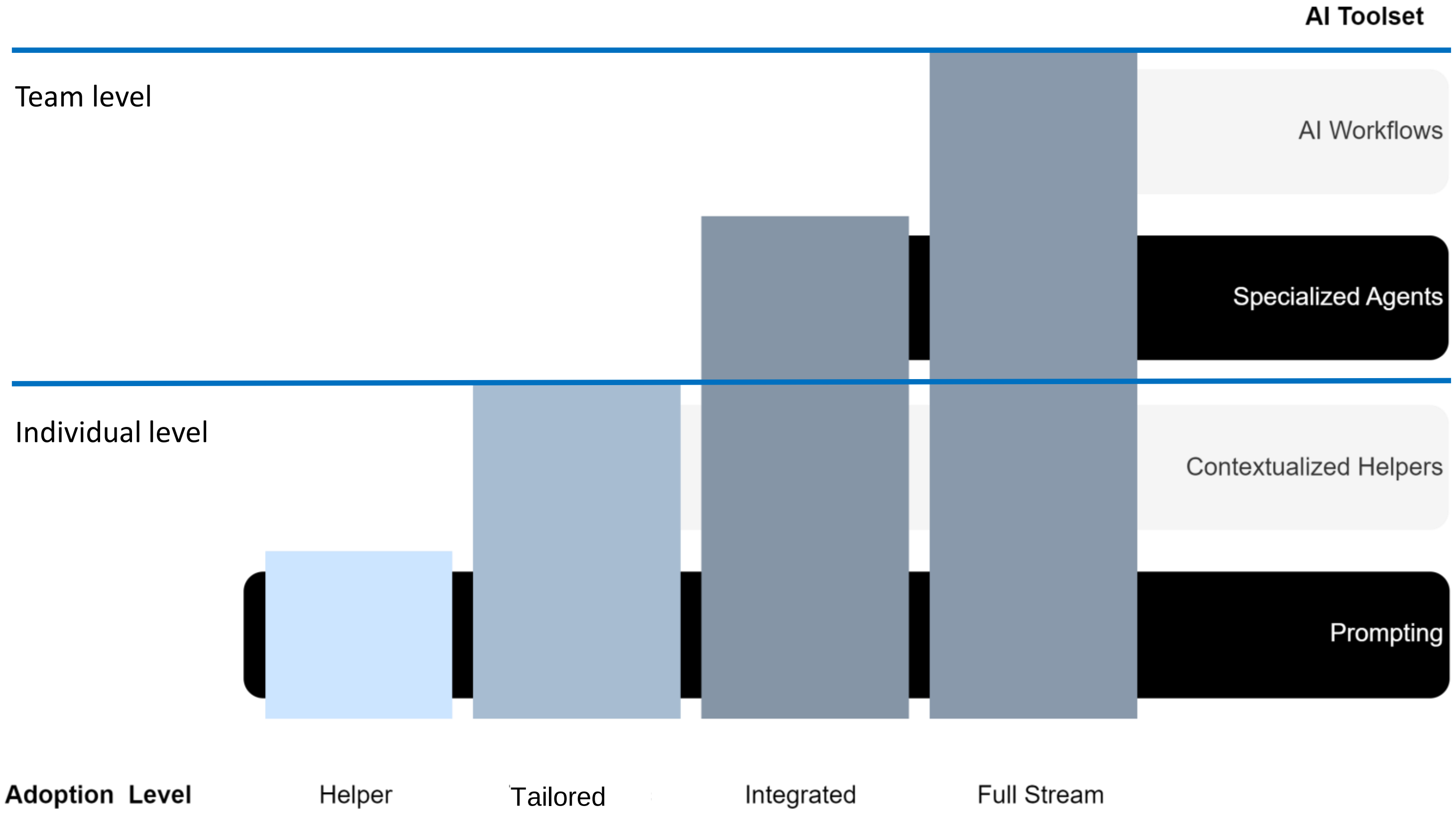
Challenges:

Getting the “plumbing” to work just right is a massive undertaking and we are still early in this. The quality of the solutions will inevitably evolve, but economics of all of this will remain a challenge. And as with pure assistants – providing proper context is essential



AI augmentation ladders

Level	Scope	Capabilities
Helper	Individual	**Prompt-driven LLMs** to solve problems
Tailored	Individual	Contextualized LLM/Generative AI setups with rich context, e.g. via **RAG**
Integrated	Team	**Specialized AI-tools/agents** focused on certain tasks, such as pull request reviews. Possibly fine-tuned models.
Full Stream	Team	**Integrated, orchestrated agents** exchanging information covering several steps of the development stream



Considerations

We expect at least the following trends:

1. **Increased Integration Pushes Technology in the Background**

2. **Tool Selection Remains a Challenge**

For a larger audience to adopt a new tool, the tool needs to

- be stable and capable enough and
- have a reasonable market share, support and future
- provide reasonable ROI on the required trials, trainings and integrations

Way forward

Maximizing AI Usage for Efficiency

AI as the First Step

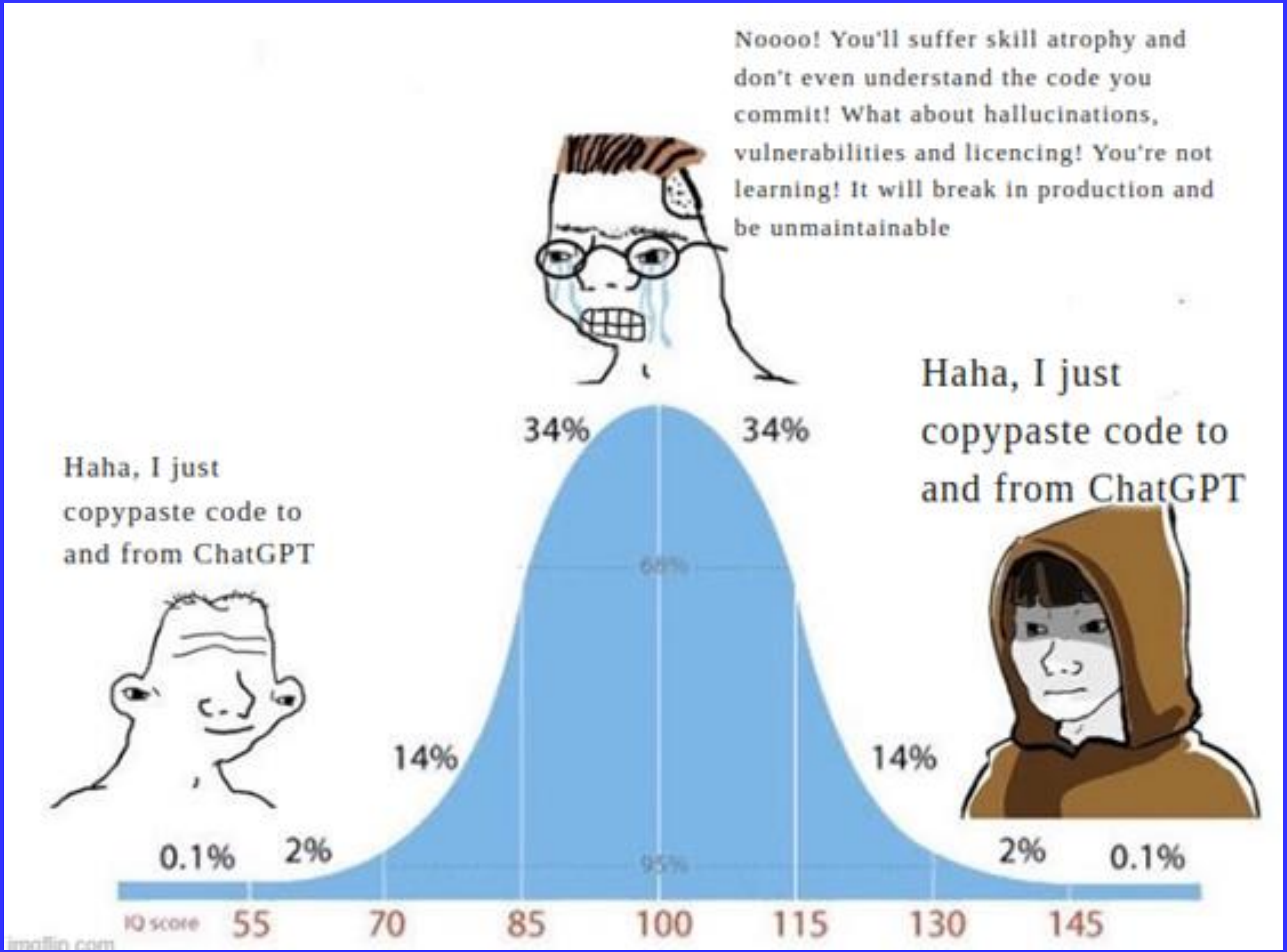
Developing AI-Augmented Skills

Lowering the Threshold for Seeking Assistance

Balancing AI Use and Skill Development

Mitigating obstacles both team and organizational level

Level	Do more	Do less
Juniors	Rapid skill acquisition through AI	Bothering seniors
	Using AI for code review before PRs	Blindly copy-pasting ChatGPT code
	Sharing AI techniques with the team	Focusing on speed over learning
Mid-levels	Boosting productivity with AI assistance	Grunt work
	Tackling senior-level challenges with AI	Sticking to familiar, routine tasks
Seniors	Delegating to AI	Manual work
	Adopting AI and building custom tools	Delegating easy tasks
	Guiding team to use AI	Helping team members with easy stuff
	Guiding stakeholders to use AI	Sitting in meetings
	Challenge your thinking	Assuming your old ways are optimal


Noooo! You'll suffer skill atrophy and don't even understand the code you commit! What about hallucinations, vulnerabilities and licencing! You're not learning! It will break in production and be unmaintainable
Haha, I just copypaste code to and from ChatGPT
Haha, I just copypaste code to and from ChatGPT
34%
34%
68%
14%
14%
0.1%
2%
2%
0.1%
95%
IQ score
55
70
85
100
115
130
145
imgflip.com

SIILI ®