

# Искусственный интеллект нельзя естественный

Александр Попов

17 октября 2025 г.

# Анонс

Всё чаще слышно, что искусственный интеллект смог сделать что-то, что, как считалось ранее, мог делать только человек — доказывать математические теоремы, выигрывать международные математические олимпиады или международные студенческие олимпиады по программированию.

Поговорим об искусственном и естественном интеллектах, зададимся вопросами, заменит ли искусственный интеллект разработчиков и к чему им стоит готовиться, а также пофилософствуем о том, что в дивном новом мире ждет разработчиков (и всех остальных) и как изменится процесс разработки.

# 17.01.2024: AlphaGeometry (Google DeepMind)

nature

Explore content ▾ About the journal ▾ Publish with us ▾

nature > articles > article

Article | Open access | Published: 17 January 2024

## Solving olympiad geometry without human demonstrations

Trieu H. Trinh , Yuhuai Wu, Quoc V. Le, He He & Thang Luong 

Nature 625, 476–482 (2024) | Cite this article

343k Accesses | 234 Citations | 989 Altmetric | Metrics

 An Author Correction to this article was published on 23 February 2024

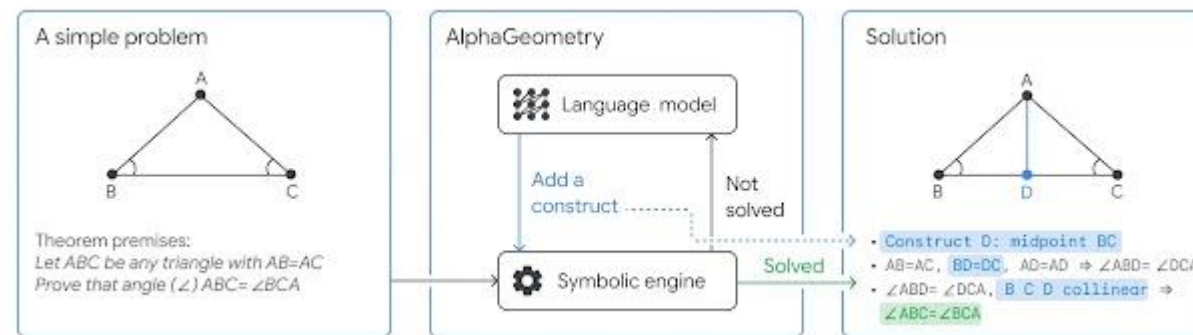
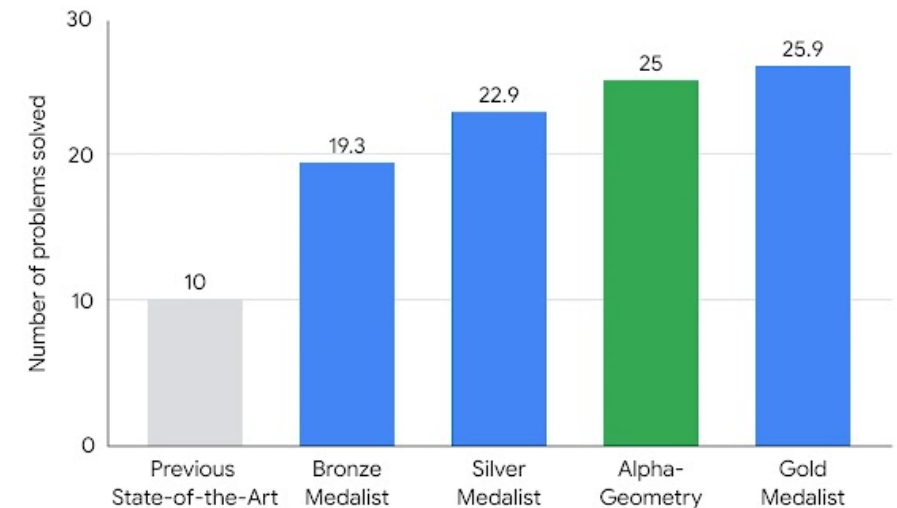
 This article has been updated

### Abstract

Proving mathematical theorems at the olympiad level represents a notable milestone in human-level automated reasoning<sup>1,2,3,4</sup>, owing to their reputed difficulty among the world's best talents in pre-university mathematics. Current machine-learning approaches, however, are not applicable to most mathematical domains owing to the high cost of translating human proofs into machine-verifiable format. The problem is even worse for geometry because of its unique translation challenges<sup>1–3</sup>, resulting in severe scarcity of training data. We propose AlphaGeometry, a theorem prover for Euclidean plane geometry that sidesteps the need for human demonstrations by synthesizing millions of theorems and proofs across different levels of complexity. AlphaGeometry is a neuro-symbolic system that uses a neural language model, trained from scratch on our large-scale synthetic data, to guide a symbolic deduction engine through infinite branching points in challenging problems. On a test set of 30 latest olympiad-level problems, AlphaGeometry solves 25, outperforming the previous best method that only solves ten problems and approaching the performance of an average International Mathematical Olympiad (IMO) gold medalist. Notably, AlphaGeometry produces human-readable proofs, solves all geometry problems in the IMO 2000 and 2015 under human expert evaluation and discovers a generalized version of a translated IMO theorem in 2004.

- “In our benchmarking set of 30 Olympiad geometry problems (IMO-AG-30), compiled from the Olympiads from 2000 to 2022, AlphaGeometry solved 25 problems under competition time limits. This is approaching the average score of human gold medalists on these same problems. The previous state-of-the-art approach, known as “Wu’s method”, solved 10.”

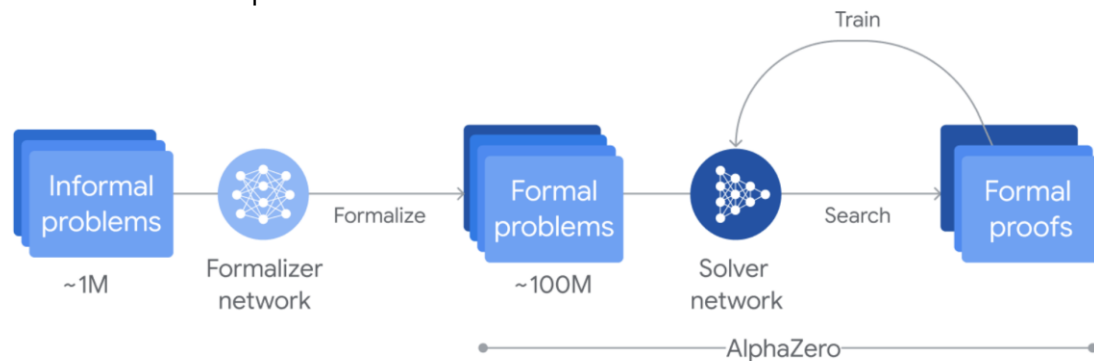
Approaching the Olympiad gold-medalist standard



<https://www.nature.com/articles/s41586-023-06747-5>

# 25.06.2024: AlphaProof and AlphaGeometry 2 (Google DeepMind) — 28/42 points (unofficially)

- “First, the problems were manually translated into formal mathematical language for our systems to understand. In the official competition, students submit answers in two sessions of 4.5 hours each. Our systems solved one problem within minutes and took up to three days to solve the others.”
- “AlphaProof solved two algebra problems and one number theory problem by determining the answer and proving it was correct. This included the hardest problem in the competition, solved by only five contestants at this year’s IMO. AlphaGeometry 2 proved the geometry problem, while the two combinatorics problems remained unsolved.”



Process infographic of AlphaProof’s reinforcement learning training loop: Around one million informal math problems are translated into a formal math language by a formalizer network. Then a solver network searches for proofs or disproofs of the problems, progressively training itself via the AlphaZero algorithm to solve more challenging problems.

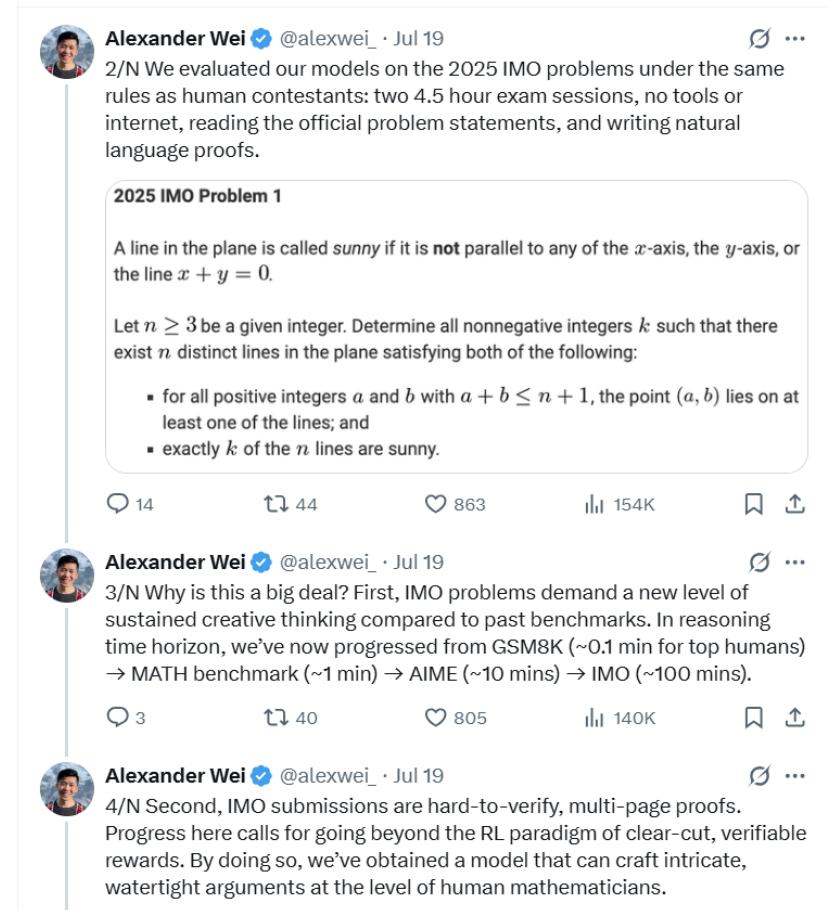
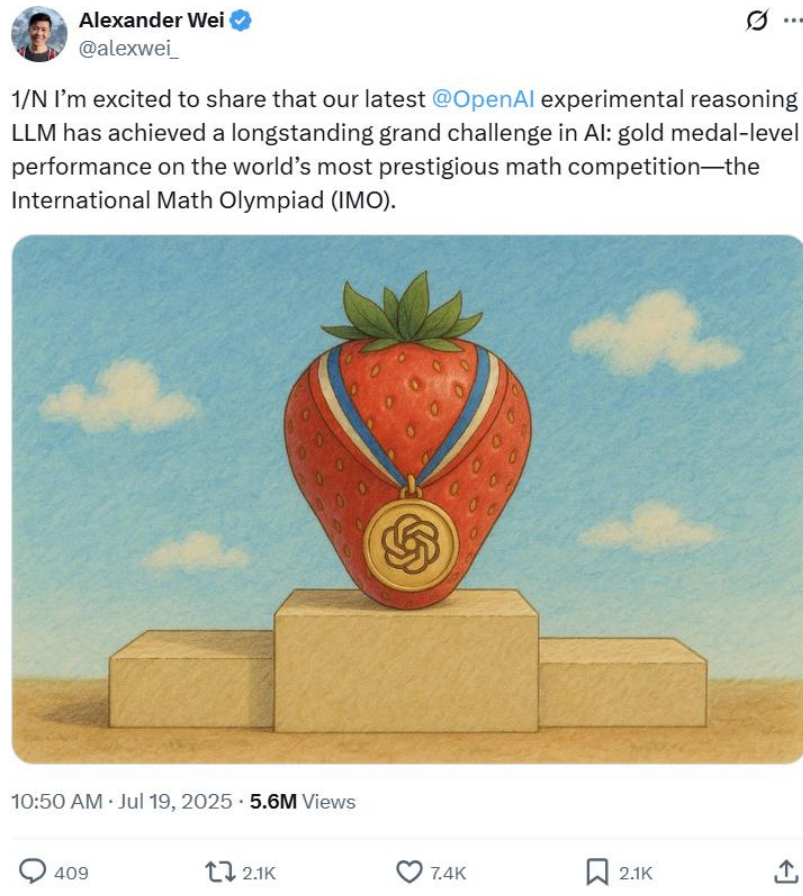
Score on IMO 2024 problems



Graph showing performance of our AI system relative to human competitors at IMO 2024. We earned 28 out of 42 total points, achieving the same level as a silver medalist in the competition.

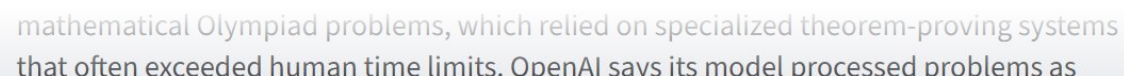
<https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/>

# 19.07.2025: OpenAI Experimental reasoning LLM (OpenAI) — 5/6 problems solved (unofficially)



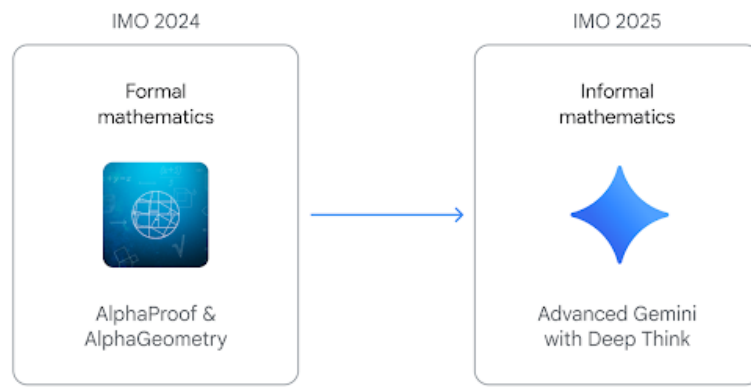
<https://x.com/alexwei/status/1946477742855532918>






# 21.07.2025: An advanced version of Gemini Deep Think (Google DeepMind) — 5/6 problems solved (officially)

- “This year, we were amongst an inaugural cohort to have our model results officially graded and certified by IMO coordinators using the same criteria as for student solutions.”
- “We can confirm that Google DeepMind has reached the much-desired milestone, earning 35 out of a possible 42 points — a gold medal score. Their solutions were astonishing in many respects. IMO graders found them to be clear, precise and most of them easy to follow.”
- “We achieved this year’s result using an advanced version of Gemini Deep Think – an enhanced reasoning mode for complex problems that incorporates some of our latest research techniques, including parallel thinking. This setup enables the model to simultaneously explore and combine multiple possible solutions before giving a final answer, rather than pursuing a single, linear chain of thought.”
- “To make the most of the reasoning capabilities of Deep Think, we additionally trained this version of Gemini on novel reinforcement learning techniques that can leverage more multi-step reasoning, problem-solving and theorem-proving data. We also provided Gemini with access to a curated corpus of high-quality solutions to mathematics problems, and added some general hints and tips on how to approach IMO problems to its instructions.”

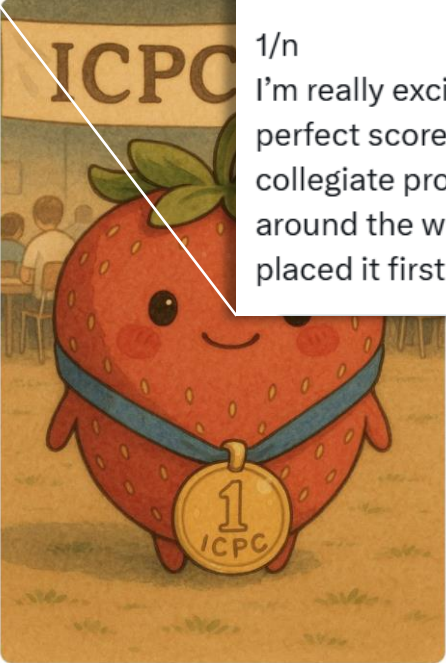


<https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/>

# 17.09.2025: OpenAI General-purpose reasoning models (OpenAI) — 12/12 problems (officially)


Mostafa Rohaninejad  @MostafaRohani · Sep 17

1/n  
I'm really excited to share that our @OpenAI reasoning system got a perfect score of 12/12 during the 2025 ICPC World Finals, the premier collegiate programming competition where top university teams from around the world solve complex algorithmic problems. This would have placed it first among all human participants. 🏆🏆



8:06 PM · Sep 17, 2025 · 2.2M Views


139 697 2.8K 675

Mostafa Rohaninejad  @MostafaRohani · Sep 17

2/n  
We officially competed in the onsite AI track of the ICPC, with the same 5-hour time limit to solve all twelve problems, submitting to the ICPC World Finals Local Judge - judged identically and concurrently to the ICPC World Championship submissions. We received the problems in the exact same PDF form, and the reasoning system selected which answers to submit with no bespoke test-time harness whatsoever. For 11 of the 12 problems, the system's first answer was correct. For the hardest problem, it succeeded on the 9th submission. Notably, the best human team achieved 11/12.

	A	B	C	D	E	F	G	H	I	J	K
AI	88 1 try	104 1 try	56 1 try	23 1 try	58 1 try	40 1 try	241 9 tries	60 1 try	58 1 try	74 1 try	70 1 try

6 29 439 99K

Mostafa Rohaninejad  @MostafaRohani · Sep 17

3/n  
We competed with an ensemble of general-purpose reasoning models; we did not train any model specifically for the ICPC. We had both GPT-5 and an experimental reasoning model generating solutions, and the experimental reasoning model selecting which solutions to submit. GPT-5 answered 11 correctly, and the last (and most difficult problem) was solved by the experimental reasoning model.

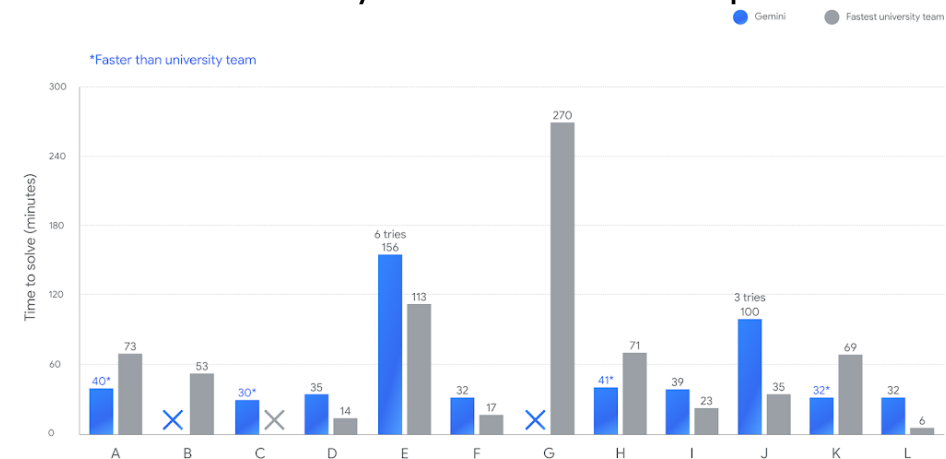
11 31 381 76K

<https://x.com/openai/status/1968368133024231902>



# 17.09.2025: An advanced version of Gemini Deep Think 2.5 (Google DeepMind) — 10/12 problems (officially)

- “Our milestone performance brings together a series of advances across pretraining, post-training, novel reinforcement learning techniques, multi-step reasoning and parallel thinking. These innovations helped Gemini explore different ways of solving complex problems, verifying solutions and continuously iterating before responding.”
- “For example, during the course of reinforcement learning, we trained Gemini to reason and generate code for some of the most difficult problems coders have faced, to learn from feedback on results and evolve its approaches. To tackle a problem, multiple Gemini agents each propose their own solutions, use terminals to execute code and tests, and then iterate the solutions based on all attempts.”
- “Gemini solved eight problems within just 45 minutes and two more problems within three hours, using a wide variety of advanced data structures and algorithms to generate its solutions. By solving 10 problems in a combined total time of 677 minutes, Gemini 2.5 Deep Think would be ranked in 2nd place overall, if compared with the university teams in the competition.”



Unless marked, Gemini passed the problem with 1 try.

<https://deepmind.google/discover/blog/gemini-achieves-gold-level-performance-at-the-international-collegiate-programming-contest-world-finals/>

# Что происходит с точки зрения разработчика?

Разработчик разрабатывает ПО



# Разработчик разрабатывает ПО

1

2

3

Сформулируем вопросы:

1. Может ли ИИ заменить разработчиков?
  - Где ИИ уже может, а где нам нужен естественный интеллект?
2. Как изменится разработка?
  - Что происходит? К чему готовиться в этой связи.
3. Как изменится ПО?
  - Out of scope / вне текущего доклада.

# Разработчик разрабатывает ПО

1

2

3

Сформулируем вопросы:

1. Может ли ИИ заменить разработчиков?
  - Где ИИ уже может, а где нам нужен естественный интеллект?
2. Как изменится разработка?
  - Что происходит? К чему готовиться в этой связи.
3. Как изменится ПО?
  - Out of scope / вне текущего доклада.

# Какими свойствами/характеристиками должны обладать разработчик и/или задача, поручаемая ему (ей)?

- С чем сравниваемся: человек с естественным интеллектом сделал задачу вручную.





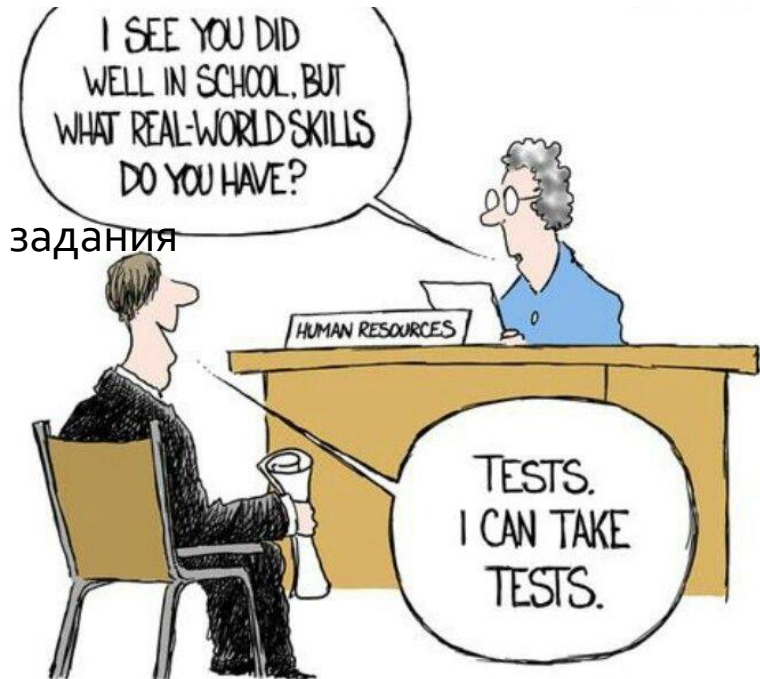
# 1. Свойство: коммуникация

- Как вообще понять, что разработчик что-то сделает?
- Надо иметь возможность рассказать о результатах.



## 2. Свойство: возможность оценить качество решения задач

- Как понять, что разработчик сделает то, что нужно, и не допустит ошибок?
- Проверить качество при найме: экзамены/дипломы/тестовые задания:
  - Бенчмарки
  - Evals/оценки на типовых задачах
- Проверить качество на реальных задачах: конкретные тестовые задания

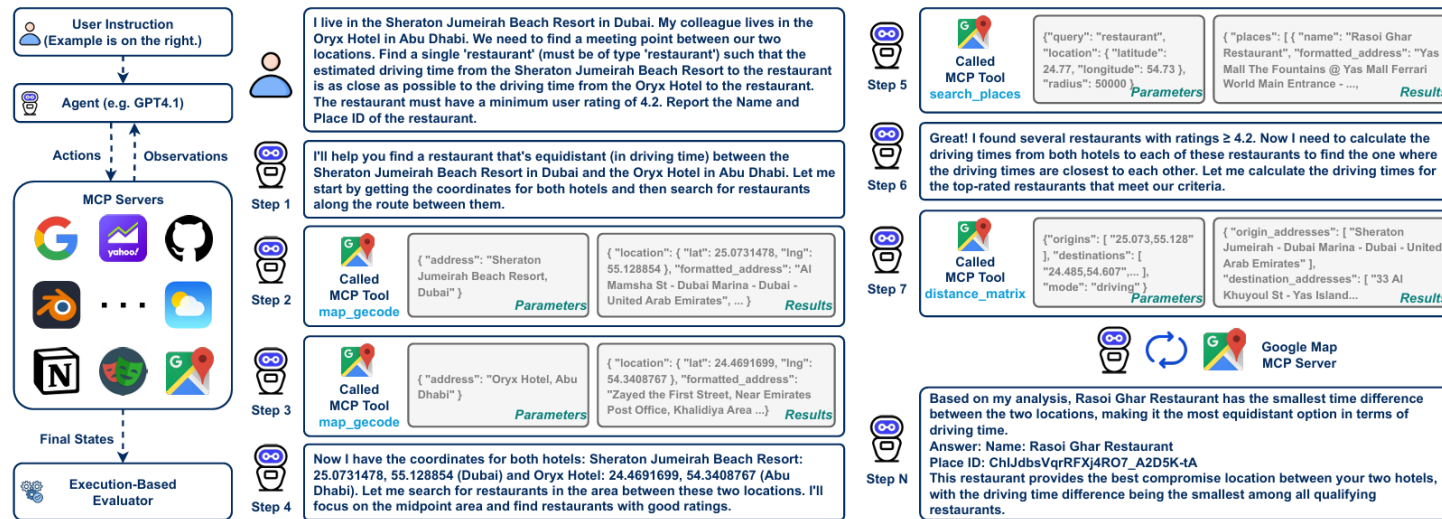


# Benchmarks (на примере модели Claude Sonnet 4.5)

	Claude Sonnet 4.5	Claude Opus 4.1	Claude Sonnet 4	GPT-5	Gemini 2.5 Pro
Agentic coding <i>SWE-bench Verified</i>	<b>77.2%</b> <b>82.0%</b> with parallel test-time compute	<b>74.5%</b> <b>79.4%</b> with parallel test-time compute	<b>72.7%</b> <b>80.2%</b> with parallel test-time compute	<b>72.8%</b> GPT-5 <b>74.5%</b> GPT-5-Codex	<b>67.2%</b>
Agentic terminal coding <i>Terminal-Bench</i>	<b>50.0%</b>	<b>46.5%</b>	<b>36.4%</b>	<b>43.8%</b>	<b>25.3%</b>
Agentic tool use <i>τ2-bench</i>	Retail <b>86.2%</b>	Retail <b>86.8%</b>	Retail <b>83.8%</b>	Retail <b>81.1%</b>	—
	Airline <b>70.0%</b>	Airline <b>63.0%</b>	Airline <b>63.0%</b>	Airline <b>62.6%</b>	—
	Telecom <b>98.0%</b>	Telecom <b>71.5%</b>	Telecom <b>49.6%</b>	Telecom <b>96.7%</b>	—
Computer use <i>OSWorld</i>	<b>61.4%</b>	<b>44.4%</b>	<b>42.2%</b>	—	—
High school math competition <i>AIME 2025</i>	<b>100%</b> (python)	<b>78.0%</b>	<b>70.5%</b>	<b>99.6%</b> (python)	<b>88.0%</b>
	<b>87.0%</b> (no tools)			<b>94.6%</b> (no tools)	
Graduate-level reasoning <i>GPQA Diamond</i>	<b>83.4%</b>	<b>81.0%</b>	<b>76.1%</b>	<b>85.7%</b>	<b>86.4%</b>
Multilingual Q&A <i>MMMLU</i>	<b>89.1%</b>	<b>89.5%</b>	<b>86.5%</b>	<b>89.4%</b>	—
Visual reasoning <i>MMMU (validation)</i>	<b>77.8%</b>	<b>77.1%</b>	<b>74.4%</b>	<b>84.2%</b>	<b>82.0%</b>
Financial analysis <i>Finance Agent</i>	<b>55.3%</b>	<b>50.9%</b>	<b>44.5%</b>	<b>46.9%</b>	<b>29.4%</b>

<https://www.anthropic.com/news/claude-sonnet-4-5>

# (Benchmarks) MCP-Universe: Benchmarking Large Language Models with Real-World Model Context Protocol Servers



**Figure 1:** Example from MCP-Universe illustrating realistic challenges, including real-world tool usage, long-horizon multi-turn tool calls, long context windows, scattered evidence, and large tool spaces. Unlike prior work, MCP- Universe is grounded in real-world MCP servers connected to actual data sources and environments.

**Figure 3:** Distribution of tasks in MCP-Universe across different application domains.

<https://arxiv.org/abs/2508.14704v1>

# (Evals) GDPval: Evaluating AI Model Performance on Real-World Economically Valuable Tasks



Figure 2: GDPval includes real-world work from 44 occupations.

<https://arxiv.org/abs/2510.04374v1>



# Тесты на реальных задачах

- Придумали или определили целевые задачи
- Придумали систему оценки/тестирования/валидации
- Регулярно контролируем качество решения задач, особенно если:
  - Данные/задачи могли измениться/сместиться
  - Модель изменилась (например, обновилась)
  - Общий пайплайн изменился
  - и др.

### 3. Характеристика: цена ошибки => Свойство: доверие при данной цене ошибки

- Что делать, если что-то пошло не так? А если всё так?
- Вряд ли мы сразу поручим новичку критически важную задачу.
- Обычно тренируемся делегировать на задачах, не являющихся критическими.
- Можно ли сделать Undo?
  - Если нет, стоит добавлять проверки

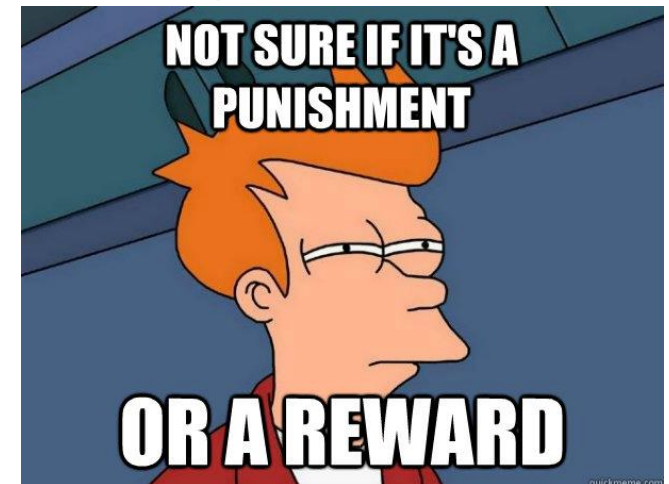


JANE-CLARK.TUMBLR

3\*. Характеристика: цена ошибки

=> N. Свойство: мотивация, поощрение/наказание при обучении

- Языковые модели тренируют с наградой за «правильное» выполнение задачи.
- За неправильное выполнение задачи их не штрафуют => модели проще сгенерировать любое решение, чем признать, что она не знает, как решить задачу.
- В долгосрочные цели живого сотрудника часто входит задача саморазвития и обучения на собственных ошибках.



## 4. Свойство: способность планировать

- Как понять, что разработчик осилит более сложную задачу?
- Разбить сложную задачу на множество более простых. Запланировать решение более мелких задач и интеграцию для получения финального результата.



# Understanding the planning of LLM agents: A survey

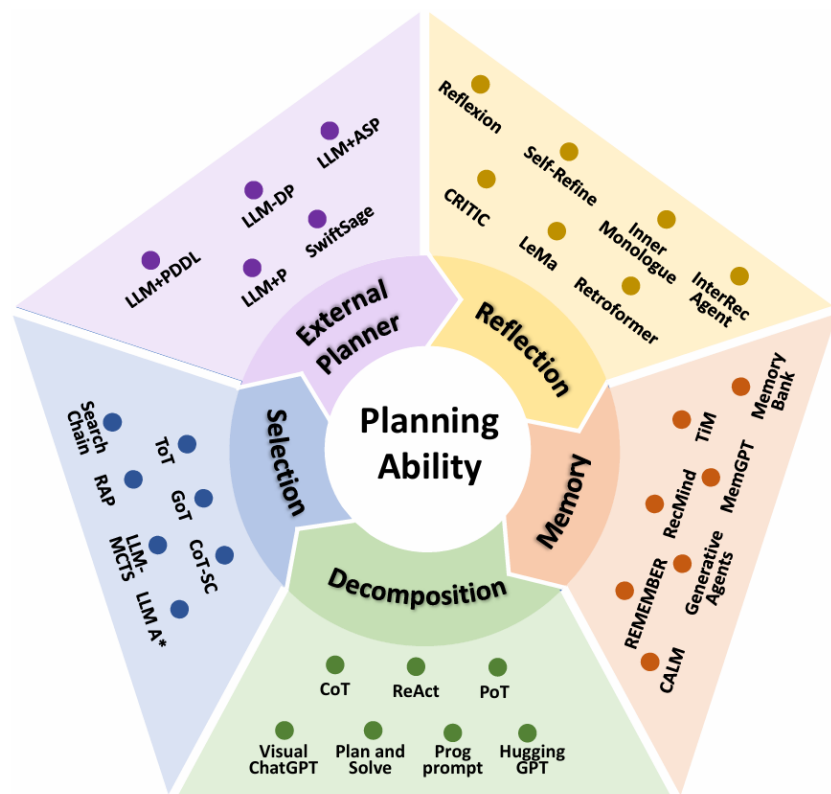


Figure 1: Taxonomy on LLM-Agent planning.

<https://arxiv.org/abs/2402.02716>

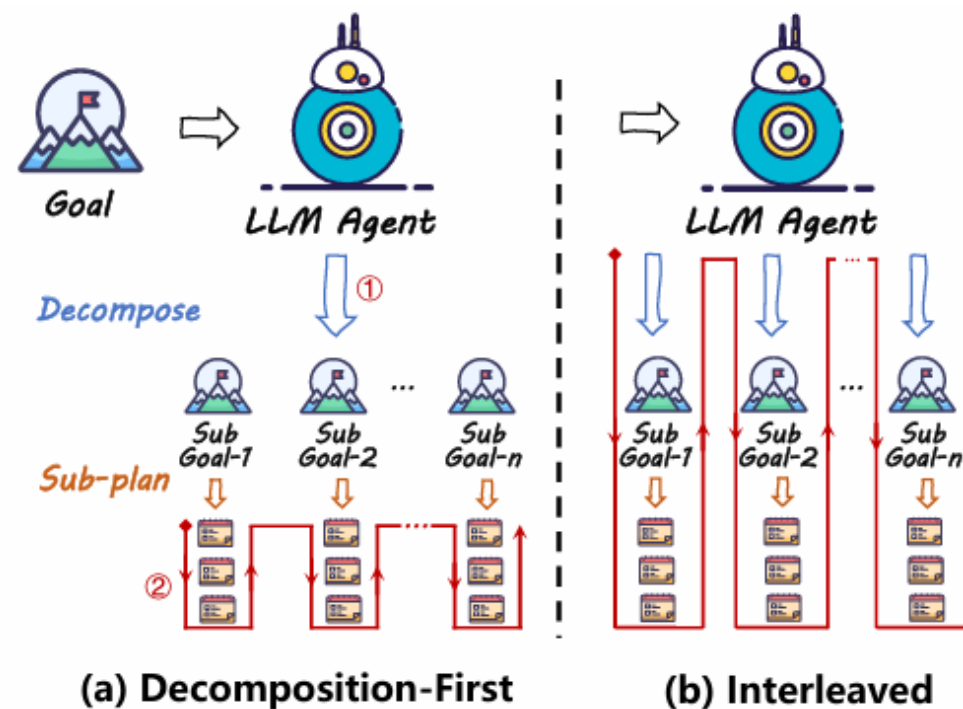


Figure 2: Types of task decomposition manners.



## 5. Свойство: эффективность при ограниченных ресурсах

- Как понять, что разработчик не задумается надолго? Или не будет решать задачу неэффективно (overthinking)?
- Обычно при решении задач есть ограничения по времени (например, thinking budget) и бюджету (например, доступное оборудование для того, чтобы разместить модель для инференса).



# Don't Overthink it. Preferring Shorter Thinking Chains for Improved LLM Reasoning

Q: Find the sum of all positive integers  $n$  such that  $n+2$  divides the product  $3(n+3)(n^2+9)$

**majority@k**



<think> ..... </think> So the answer is 52  
<think> ..... </think> So the answer is 49  
<think> ..... </think> So the answer is 33  
<think> ..... </think> So the answer is 52

Final answer:  
52

✗

**short-1@k (ours)**



<think> ..... // Terminated thinking  
<think> ..... </think> So the answer is 49  
<think> ..... // Terminated thinking  
<think> ..... // Terminated thinking

Final answer:  
49

✓

Figure 1: Visual comparison between majority voting and our proposed method *short-m@k* with  $m = 1$  (“...” represent thinking time). Given  $k$  parallel attempts for the same question, *majority@k* waits until all attempts are done, and perform majority voting among them. On the other hand, our *short-m@k* method halts computation for all attempts as soon as the first  $m$  attempts finish “thinking”, which saves much compute and time, and surprisingly also boost performance in most cases.

## 6. Свойство: автономность

- Как понять, что разработчик не будет дергать вас по пустякам?
- Если мы делегируем кому-то задачу, мы ожидаем, что они могут какое-то время решать её самостоятельно/автономно, не тревожа постановщика задачи каждую секунду.
- (Также мы можем провести оценку качества рассуждений/планов и качества шагов рассуждений/планирования (faithfulness).)



# Measuring AI Ability to Complete Long Tasks

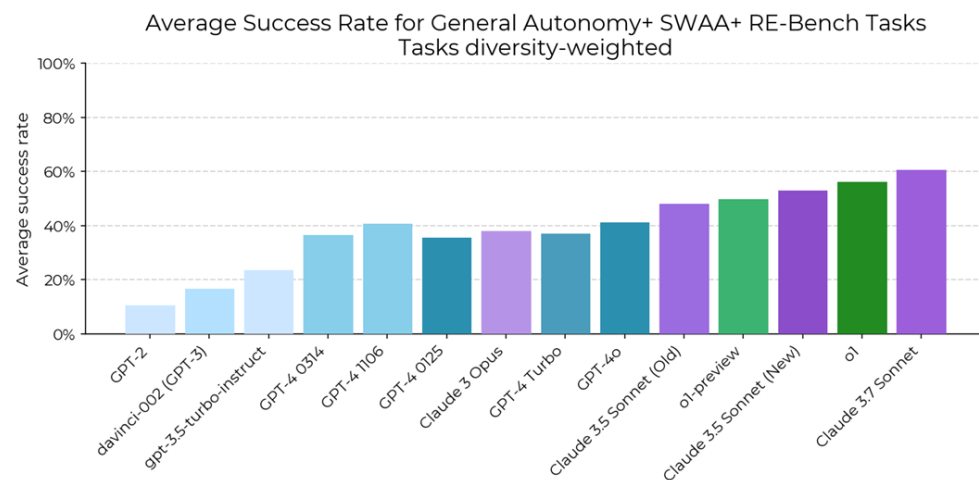


Figure 3: Average task success rate across our entire combined suite, for each model. As with all of the results reported in the main body of this work, to reduce the influence of large task families, we weight each task by the inverse square root of the number of tasks in the family it belongs to.

<https://arxiv.org/abs/2503.14499>

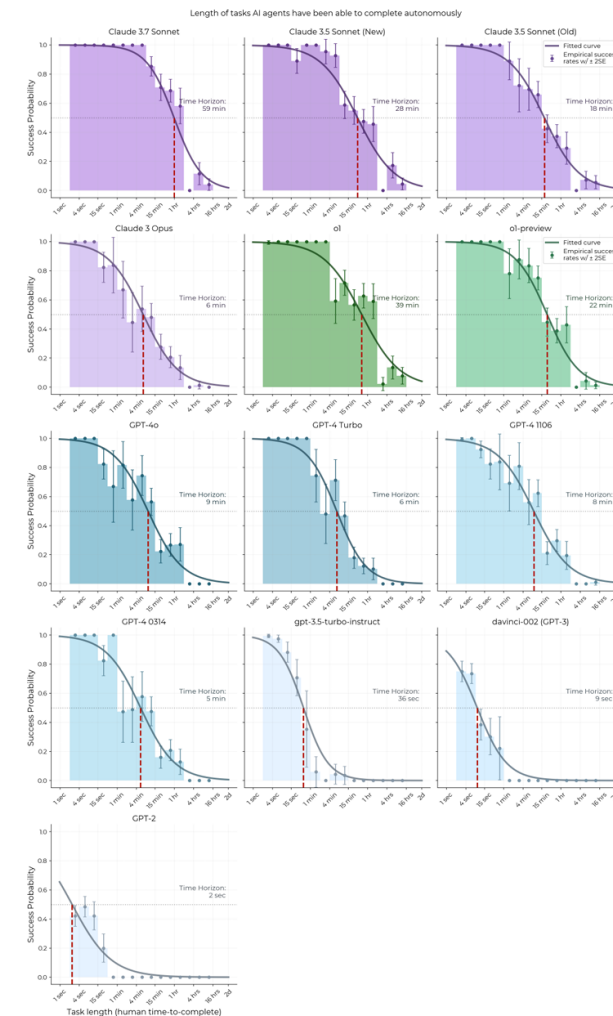


Figure 5: Success rates of all models on our test suite, showing the computation of time horizon as predicted 50% success rate time. The logistic fit is fairly good, though there is a jump in success rate between <1 minute tasks and >1 minute tasks, which corresponds to the boundary between SWAA and HCAST tasks.

# Measuring AI Ability to Complete Long Tasks

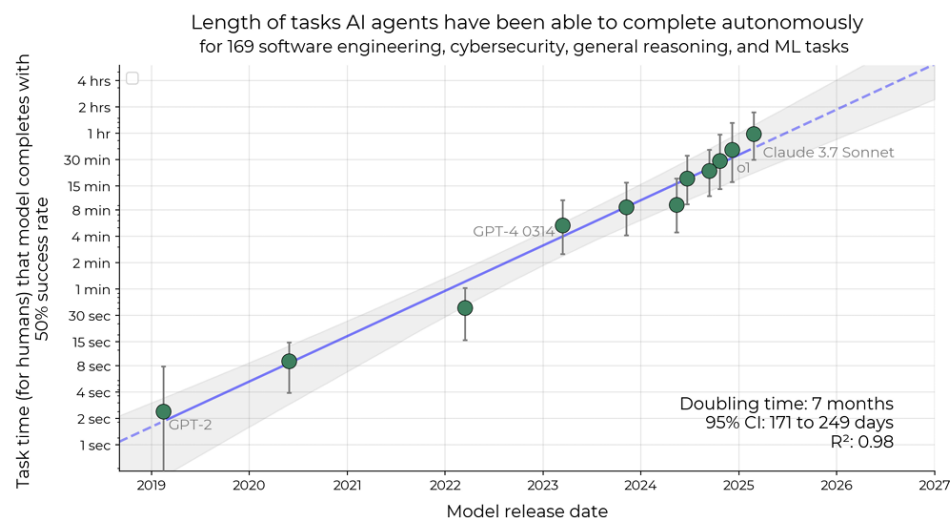


Figure 1: The length of tasks (measured by how long they take human professionals) that generalist autonomous frontier model agents can complete with 50% reliability has been doubling approximately every 7 months for the last 6 years (Section 4). The shaded region represents 95% CI calculated by hierarchical bootstrap over task families, tasks, and task attempts. Even if the absolute measurements are off by a factor of 10, the trend predicts that in under a decade we will see AI agents that can independently complete a large fraction of software tasks that currently take humans days or weeks (Section 7).

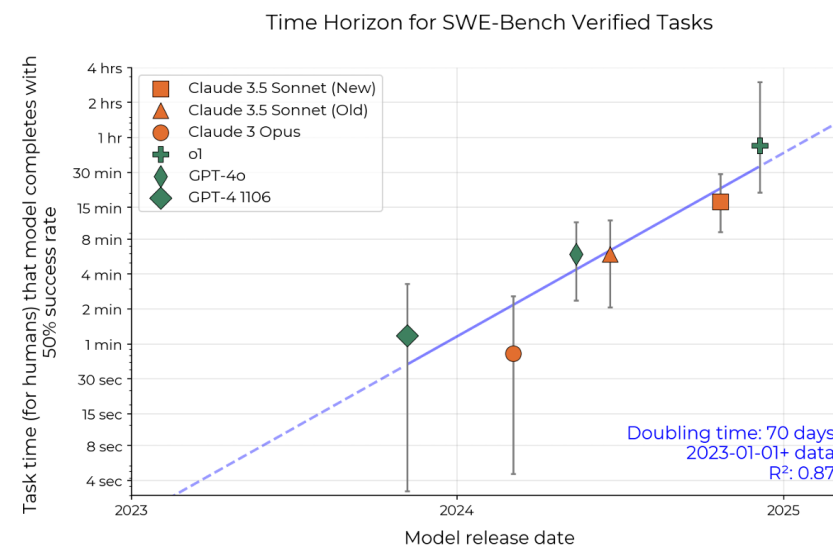


Figure 11: Performance of frontier AI models using reported SWE-bench Verified results (Section 6.3). We observe a similar exponential trend to Figure 1, albeit with a steeper slope.



## 7. Свойство: способность обнаруживать неизвестное

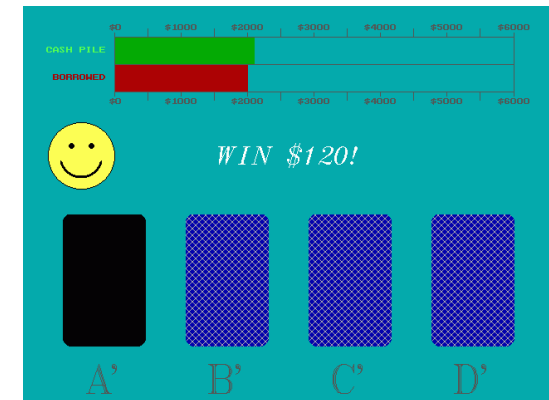
- Как понять, что если разработчик чего-то не знает, он не пойдёт изобретать велосипед/галлюцинировать?
- In-scope/out-of-scope detection (чувство известного/неизвестного)
- В ситуации принципиальной новизны опасно опираться на интуицию

Известное известное	Известное неизвестное
Неизвестное известное	Неизвестное неизвестное



# Как учится человек: Deciding advantageously before knowing the advantageous strategy (Iowa Gambling Task)

- Participants are presented with four virtual decks of cards on a computer screen. They are told that each deck holds cards that will either reward or penalize them, using game money. The goal of the game is to win as much money as possible.
  - Unbeknownst to the participant, the decks differ from each other in the balance of reward versus penalty cards. Thus, some decks are "bad decks", and other decks are "good decks"
  - Concurrent measurement of galvanic skin response shows that healthy participants show a "stress" reaction to hovering over the bad decks after only 10 trials, long before conscious sensation that the decks are bad.
  - Most healthy participants sample cards from each deck, and after about 40 or 50 selections are fairly effective at identifying and sticking to the good decks.
- Фаза 1: детекция паттернов
    - ~10 ходов — неосознанная физиологическая реакция (стресс при приближении к негативным колодам)
  - Фаза 2: регистрация паттернов («интуиция»)
    - ~30 ходов — формирование интуитивного понимания
  - Фаза 3: понимание, вербализация паттернов
    - ~50 ходов — формирование возможности объяснить



[https://en.wikipedia.org/wiki/Iowa\\_gambling\\_task](https://en.wikipedia.org/wiki/Iowa_gambling_task), <https://pubmed.ncbi.nlm.nih.gov/9036851/>

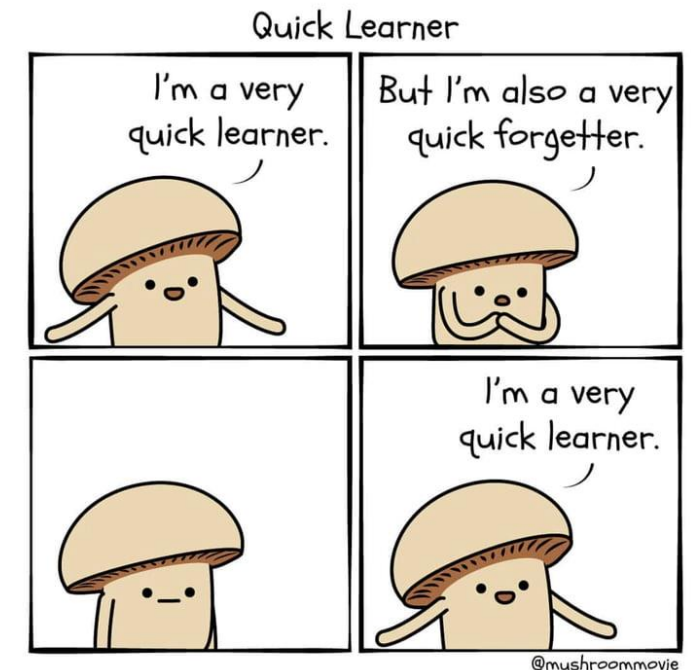
## 8. Свойство: способность обучаться новому

- Как понять, что если разработчик чего-то не знает, он не пойдёт изобретать велосипед/галлюцинировать, а пойдёт учиться и/или спрашивать совета?
- Учеба (изучение нового)
  - Процесс учёбы людей отличается от обучения моделей
    - Происходит постоянно (continuous learning / life-long learning)



## 8.5. Свойство: способность обучаться новому на ограниченном количестве примеров

- Как понять, что если разработчик чего-то не знает, он не пойдёт изобретать велосипед/галлюцинировать, а пойдёт учиться и/или спрашивать совета, и научится быстро?
- Учеба (изучение нового)
  - Процесс учёбы людей отличается от обучения моделей
    - Происходит постоянно (continuous learning / life-long learning)
    - Часто данных для обучения недостаточно
    - Зачастую данные берутся из внешней среды

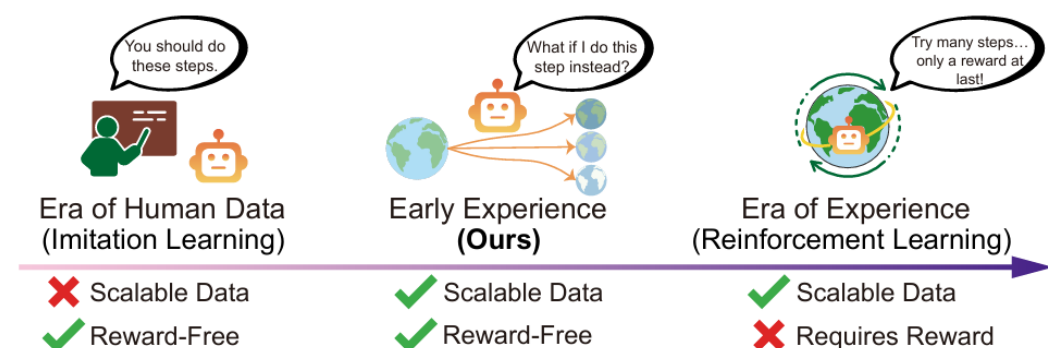


## 8.75. Свойство: мотивация, поощрение/наказание при обучении (N=8.75)

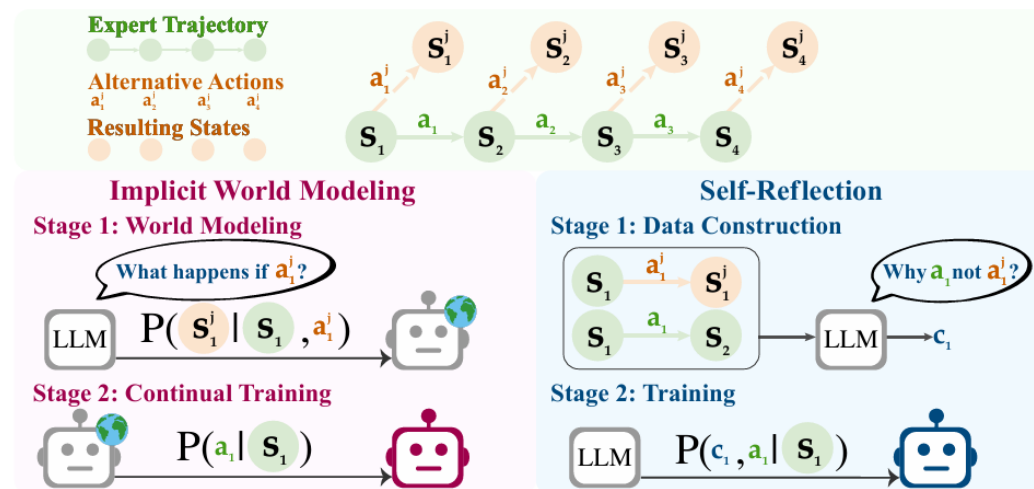
- Языковые модели тренируют с наградой за «правильное» выполнение задачи.
- За неправильное выполнение задачи их не штрафуют => модели проще сгенерировать любое решение, чем признать, что она не знает, как решить задачу.
- В долгосрочные цели живого сотрудника часто входит задача саморазвития и обучения на собственных ошибках.
  - Модели не учатся в типичном смысле слова.
    - «Обновленная» модель = новая модель, (до)обученная на обновленном наборе данных.



# Agent Learning via Early Experience



**Figure 1** Progression of training paradigms for language agents. **Left:** The *Era of Human Data* relies on expert demonstrations, where supervision comes from human-/expert-curated actions; it is reward-free (*i.e.*, does not require the environment to provide verifiable reward) but not data-scalable. **Right:** The envisioned *Era of Experience* builds upon environments with verifiable rewards, using them as the primary supervision for reinforcement learning; however, many environments either lack such rewards (Xue et al., 2025) or require inefficient long-horizon rollouts (Xie et al., 2024a). **Center:** Our *Early Experience* paradigm enables agents to propose actions and collect the resulting future states, using them as a scalable and reward-free source of supervision.



**Figure 2** Overview of the two early experience approaches. Implicit world modeling (left) augments expert trajectories with alternative actions and predicted next states, training the policy to internalize transition dynamics before deployment. Self-reflection (right) augments expert actions with self-generated explanations  $c_1$ , training the policy to reason about and revise its own decisions. Both methods use alternative actions proposed by the initial policy (LLM). The number of alternatives ( $K$ ) is a hyperparameter; for brevity, only one is illustrated.



# Hierarchical Reasoning Model

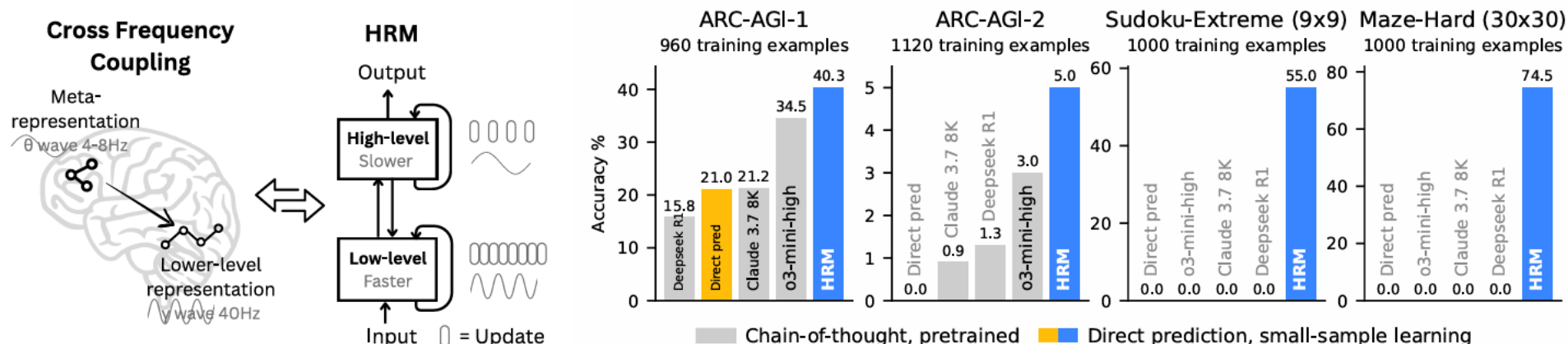
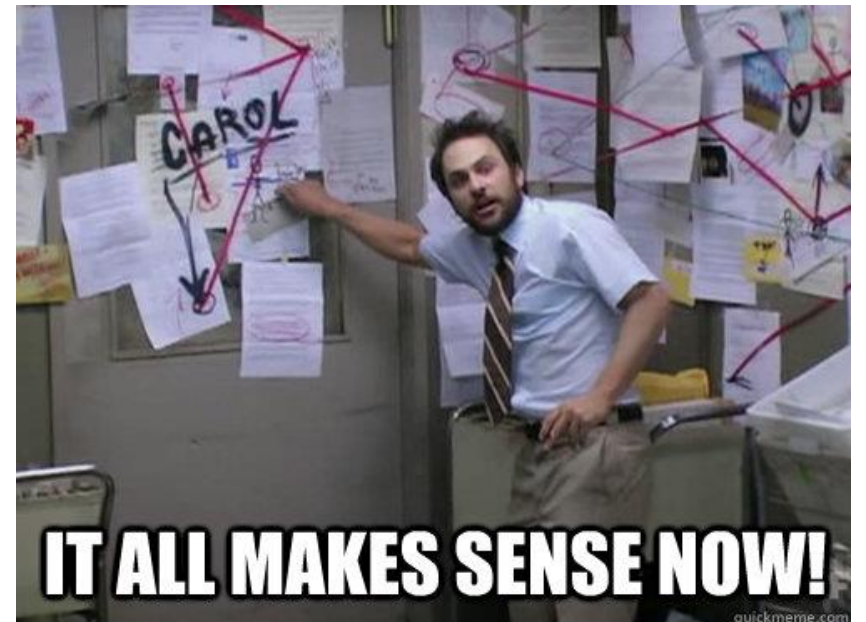


Figure 1: **Left:** HRM is inspired by hierarchical processing and temporal separation in the brain. It has two recurrent networks operating at different timescales to collaboratively solve tasks. **Right:** With only about 1000 training examples, the HRM (~27M parameters) surpasses state-of-the-art CoT models on inductive benchmarks (ARC-AGI) and challenging symbolic tree-search puzzles (*Sudoku-Extreme*, *Maze-Hard*) where CoT models failed completely. The HRM was randomly initialized, and it solved the tasks directly from inputs without chain of thoughts.

## 9. Свойство: способность формировать сжатое представление мира («модель мира») и предметной области («модель \_\_\_\_»)

- Как понять, что если разработчик чего-то не знает, он не пойдёт изобретать велосипед/галлюцинировать, а пойдёт учиться и/или спрашивать совета, и научится быстро?
- Как далее понять, что он действительно научился?
- Учеба (изучение нового)
  - Процесс учёбы отличается от обучения моделей
  - В результате формируется «модель мира» / «цифровой двойник» предметной области



# ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory

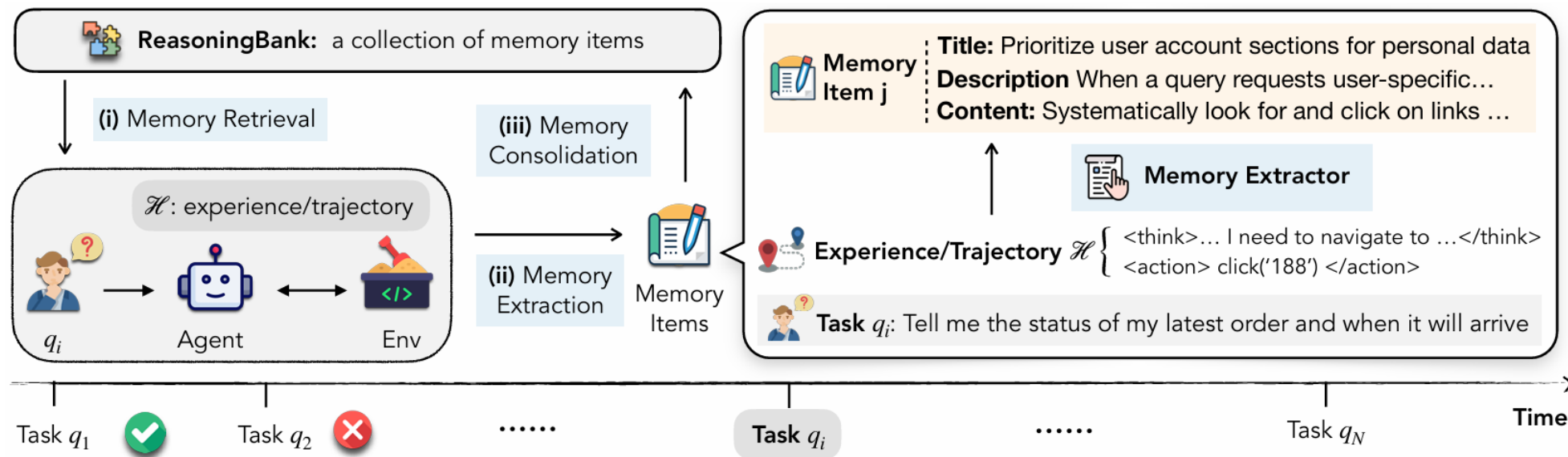


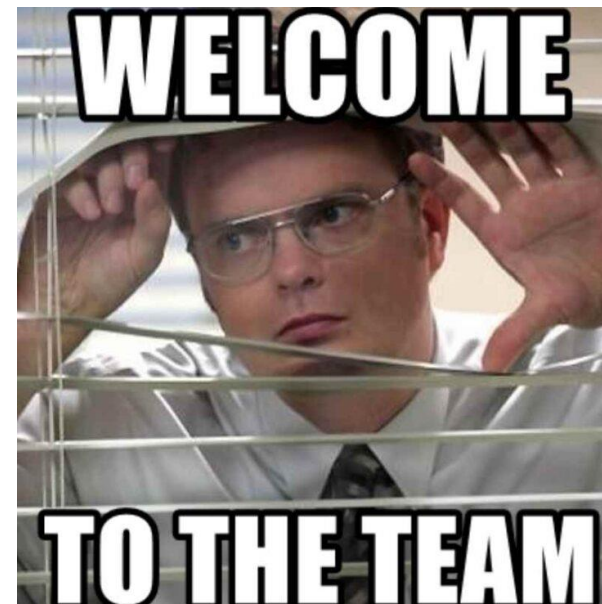
Figure 2 | Overview of REASONINGBANK. Experiences are distilled into structured memory items with a title, description, and content. For each new task, the agent retrieves relevant items to interact with the environment, and constructs new ones from both successful and failed trajectories. These items are then consolidated into REASONINGBANK, forming a closed-loop memory process.

# Необходимые свойства + статус (грубое предположение)

Свойство/характеристика	Где мы? (1=ещё не там, 5=уже там)
1. Коммуникация	5/5 коммуницируем
2. Возможность оценить качество решения задач (вводные: требуемое качество)	4/5 на benchmark'ах 3/5 на eval'ах 2/5 в тестах на реальных задачах
3. Доверие при данной цене ошибки (вводные: цена ошибки)	5/5 для экспериментов 2/5 для прода с доп. проверками 1/5 для прода as is
4. Способность планировать	2/5 предмет исследований
5. Эффективность при ограниченных ресурсах (вводные: время, вычислительный бюджет)	3/5 предмет исследований
6. Автономность	3/5 предмет исследований
7. Способность обнаруживать неизвестное	1/5 предмет исследований
8. Способность обучаться новому	2/5 предмет исследований
8.5. Способность обучаться новому на малой выборке	3/5 предмет исследований
(8.75. Мотивация, поощрение/наказание при обучении (N=8.75))	3/5 предмет исследований
9. Способность формировать сжатое представление области	1/5 предмет исследований

## 10. Свойство: встраиваемость в команду

- Как понять, что разработчик может работать в команде?
- Применять методы к команде (коллективу) агентов



# ChatDev: Communicative Agents for Software Development



Figure 1: ChatDev, a chat-powered software development framework, integrates LLM agents with various social roles, working autonomously to develop comprehensive solutions via multi-agent collaboration.

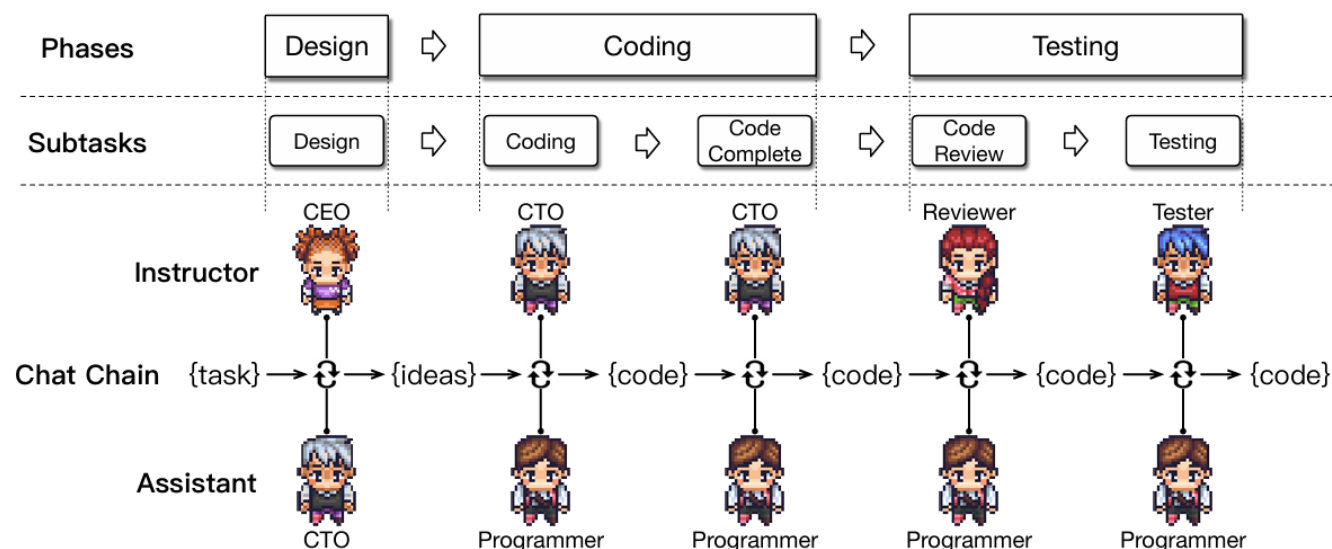
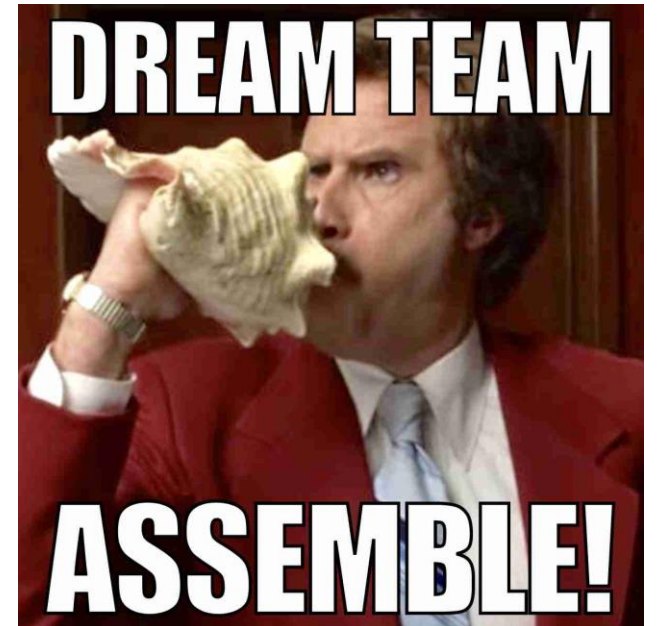


Figure 2: Upon receiving a preliminary task requirement (e.g., “develop a Gomoku game”), these software agents engage in multi-turn communication and perform instruction-following along a chain-structured workflow, collaborating to execute a series of subtasks autonomously to craft a comprehensive solution.



## 11. Свойство: организационная структура команды

- Как организовать команду?
- Попробовать множество вариантов организационной структуры.
- (\*) Как организовать гетерогенную команду? (люди + агенты)



# Multi-Agent Collaboration Mechanisms: A Survey of LLMs

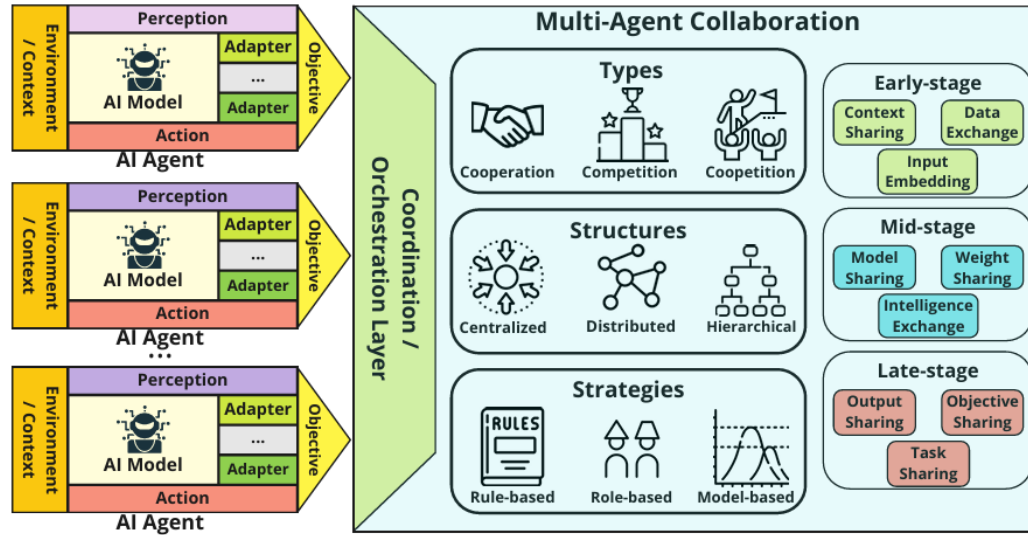


Fig. 2. Our proposed framework for LLM-based multi-agent collaborative system. Each agent consists of a language model  $m$  as the neural processor, current objective  $o$ , environment  $e$ , input perception  $x$  and corresponding output/action  $y$ . The framework's central focus is on collaboration channels  $C$  that facilitate coordination and orchestration among agents. These channels are defined by their actors (agents involved), type, structure, and strategy. Our framework is flexible, accommodating previous approaches and enabling the analysis of diverse MASs under a unified structure.

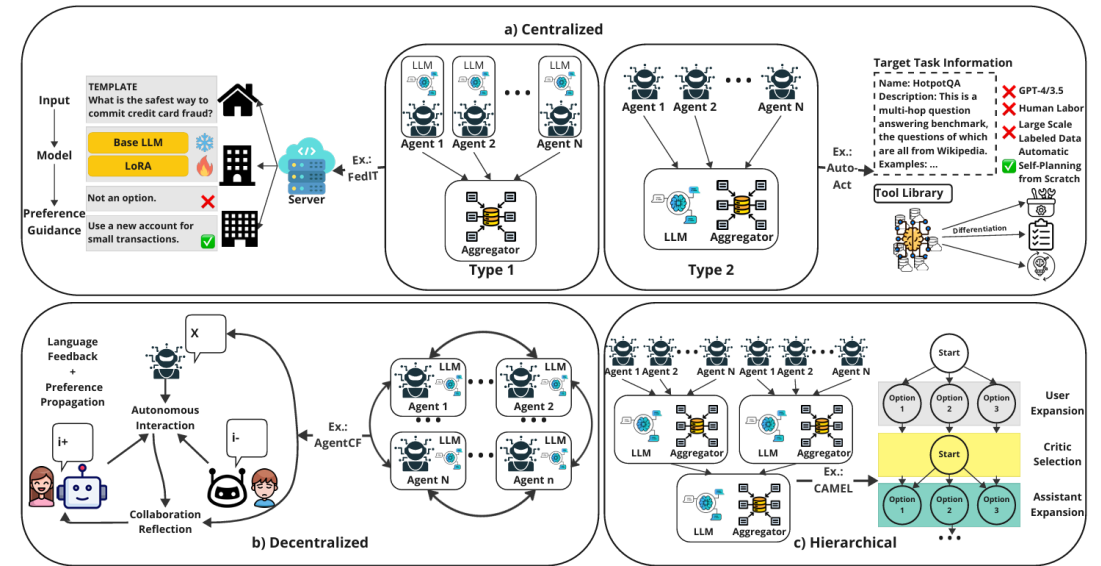


Fig. 5. Summary of communication structures of MAS. Figure (a) illustrates the centralized structure, which can be categorized into two types. In the first type, the LLM resides on distributed agents, with FedIT serving as an example. In the second type, the LLM is hosted on a central agent, as exemplified by AutoAct. Figure (b) depicts the decentralized structure, with AgentCF as an example. Finally, Figure (c) represents the hierarchical structure, with the CAMEL architecture.

# Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems by Antonio Gullí

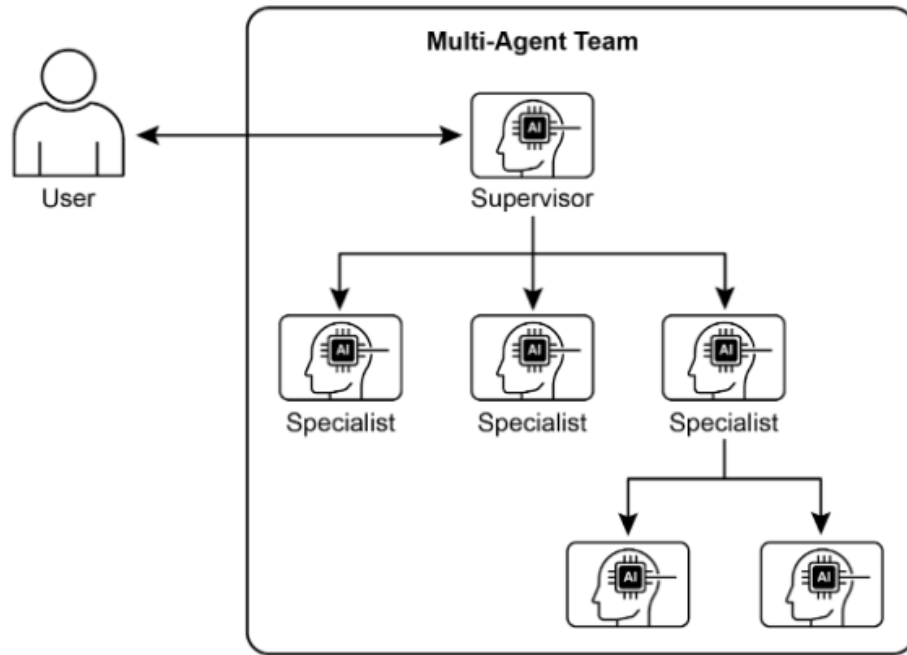


Fig.1: Example of multi-agent system

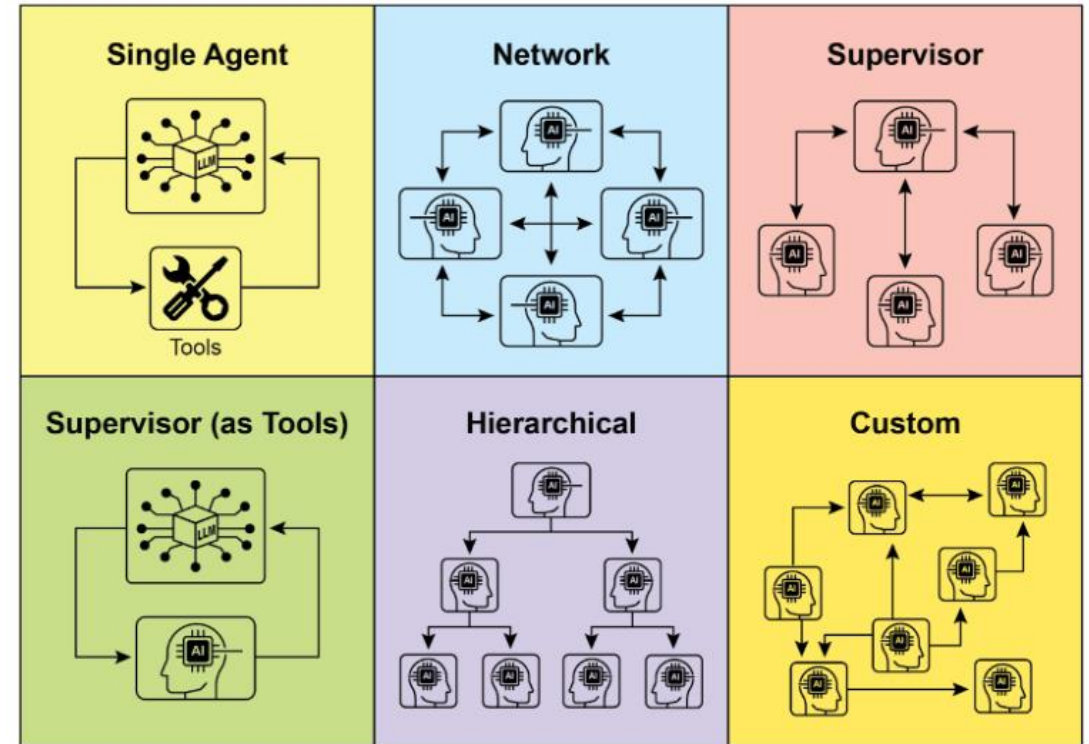


Fig. 2: Agents communicate and interact in various ways.

<https://docs.google.com/document/d/1rsaK53T3Lg5KoGwvf8ukOUvbELRtH-VoLnOIFDxBryE/preview>

## 11\*. Свойство: организационная структура гетерогенной команды (включающей и ЕИ, и ИИ-агентов)

- Как организовать гетерогенную команду?
- Попробовать множество вариантов организационной структуры.



# Разработчик разрабатывает ПО

1

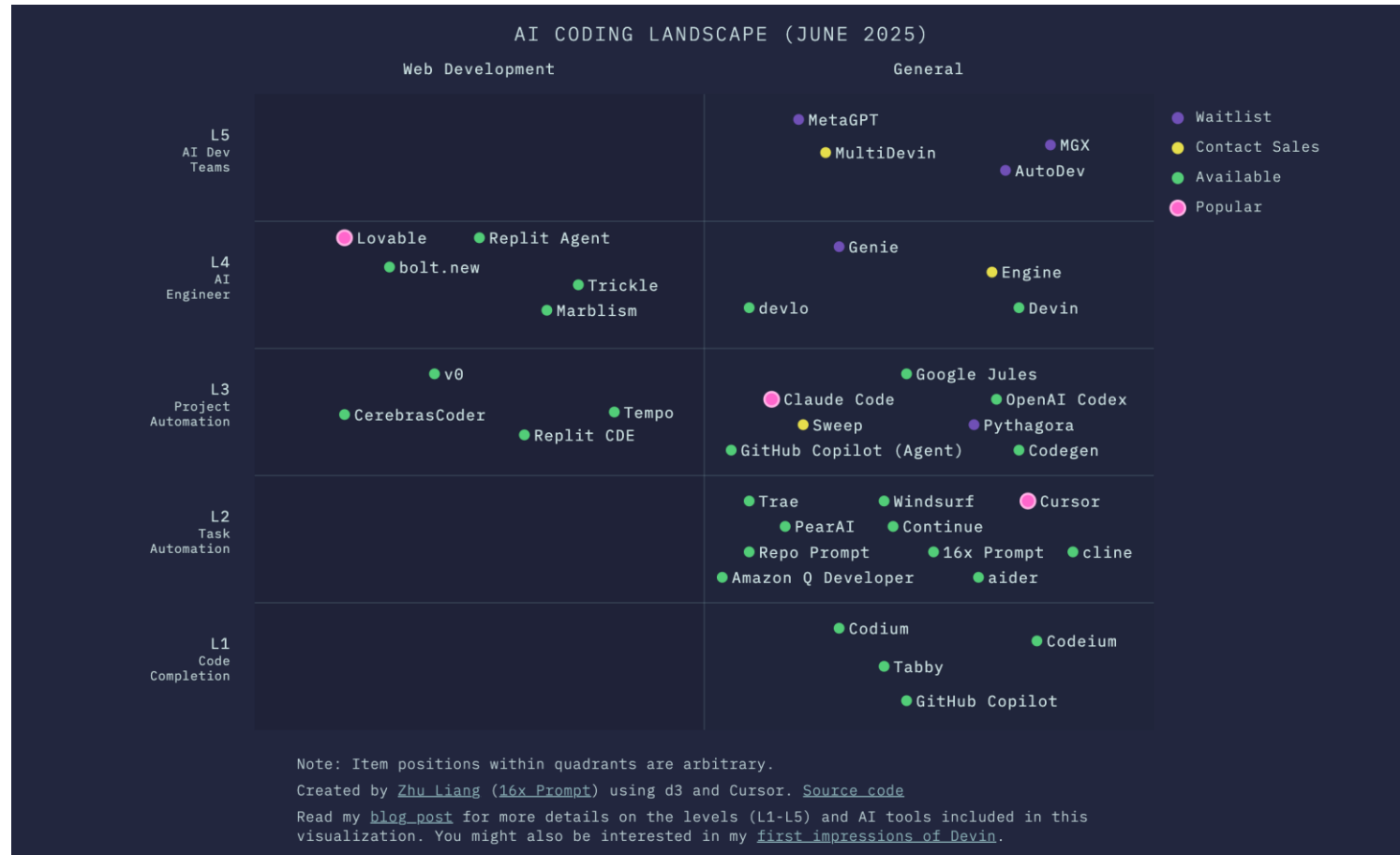
2

3

Сформулируем вопросы:

1. Может ли ИИ заменить разработчиков?
  - Где ИИ уже может, а где нам нужен естественный интеллект?
2. Как изменится разработка?
  - Что происходит? К чему готовиться в этой связи.
3. Как изменится ПО?
  - Out of scope / вне текущего доклада.

# AI Coding Landscape (June 2025)



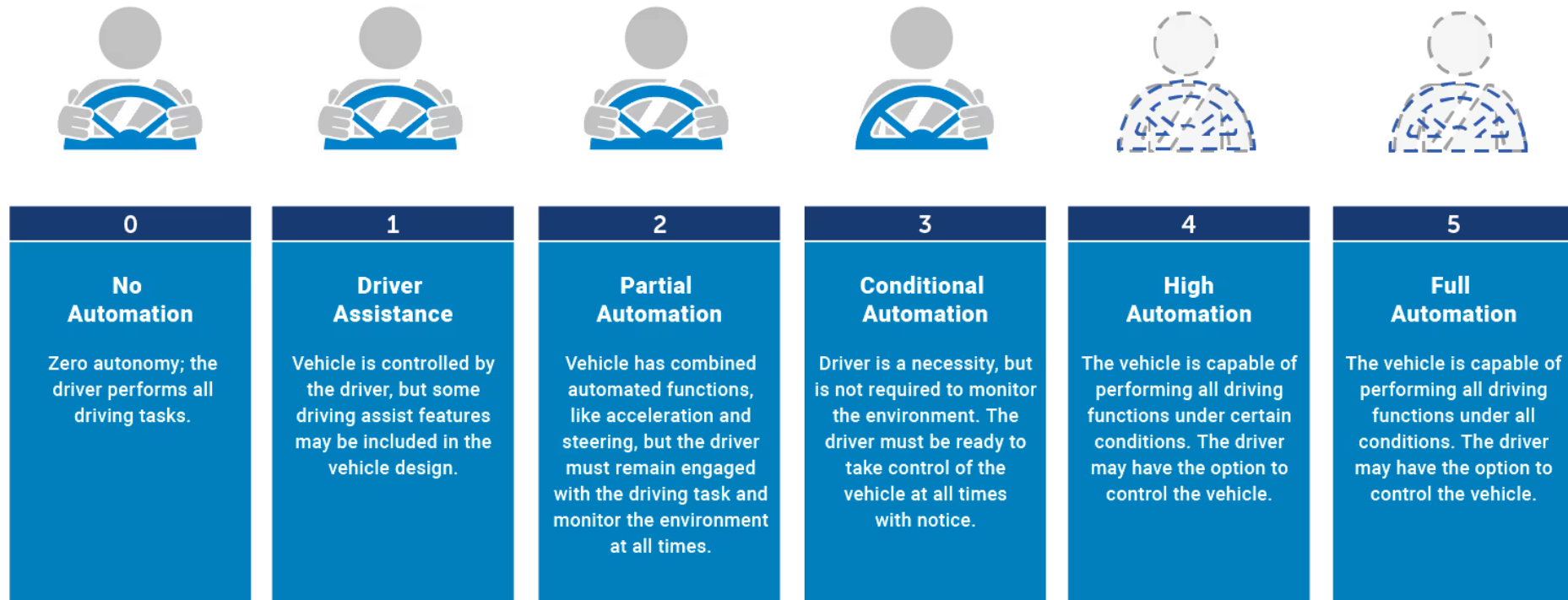
<https://paradite.github.io/ai-coding/>



# How we're thinking about the levels of code AI

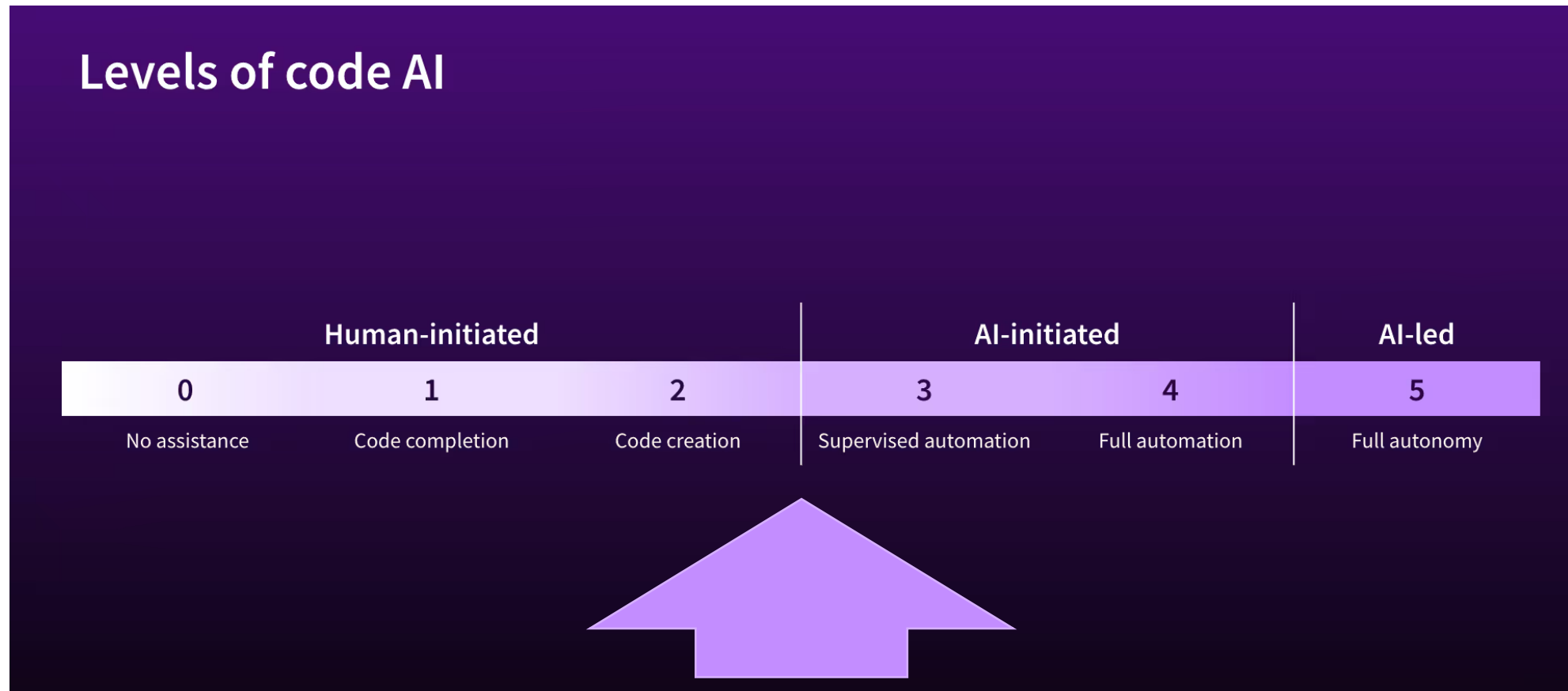
SOCIETY OF AUTOMOTIVE ENGINEERS (SAE) AUTOMATION LEVELS

Full Automation



<https://sourcegraph.com/blog/levels-of-code-ai>

# How we're thinking about the levels of code AI



Похоже, мы где-то здесь  
(предположение)

<https://sourcegraph.com/blog/levels-of-code-ai>

# Agents in Software Engineering: Survey, Landscape, and Vision

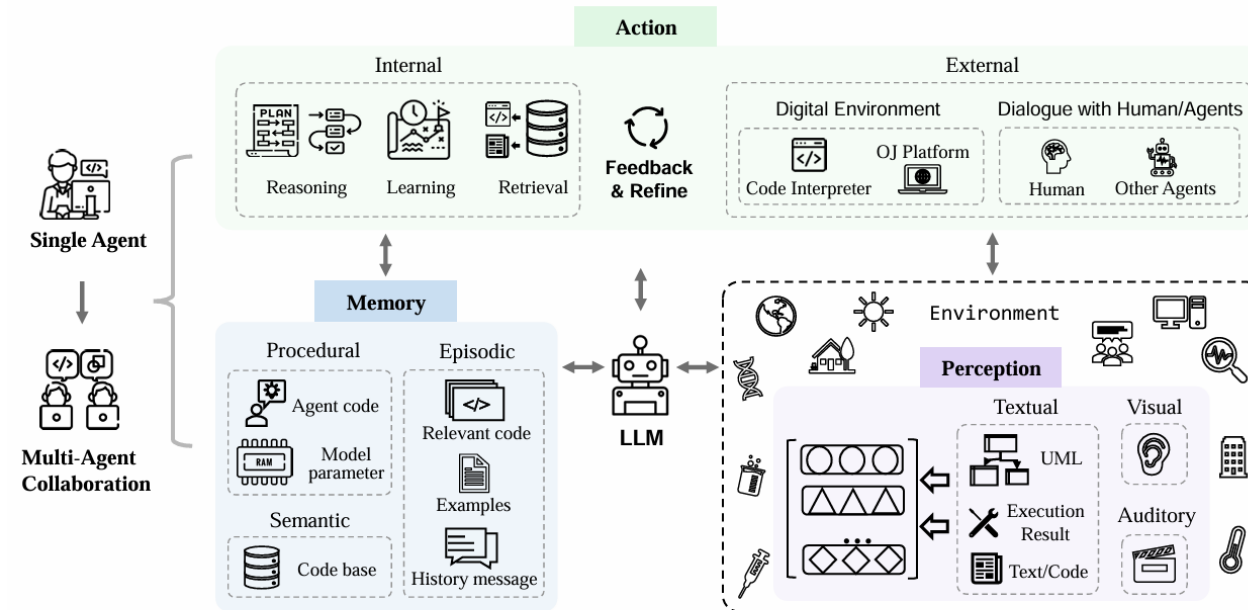


Figure 2: An overview of agent framework in SE.

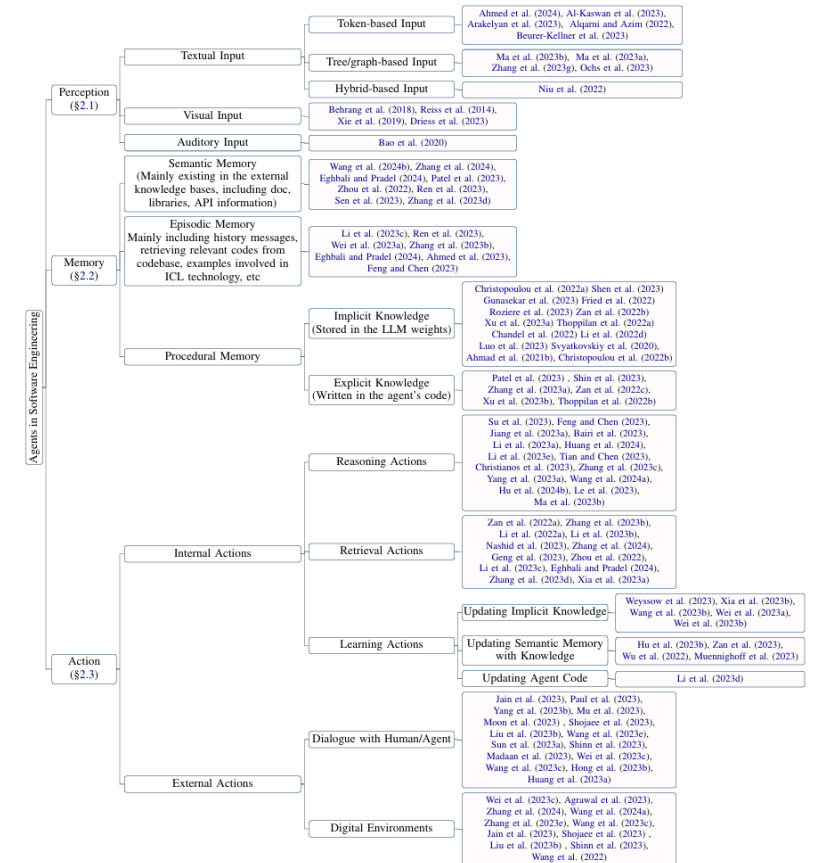
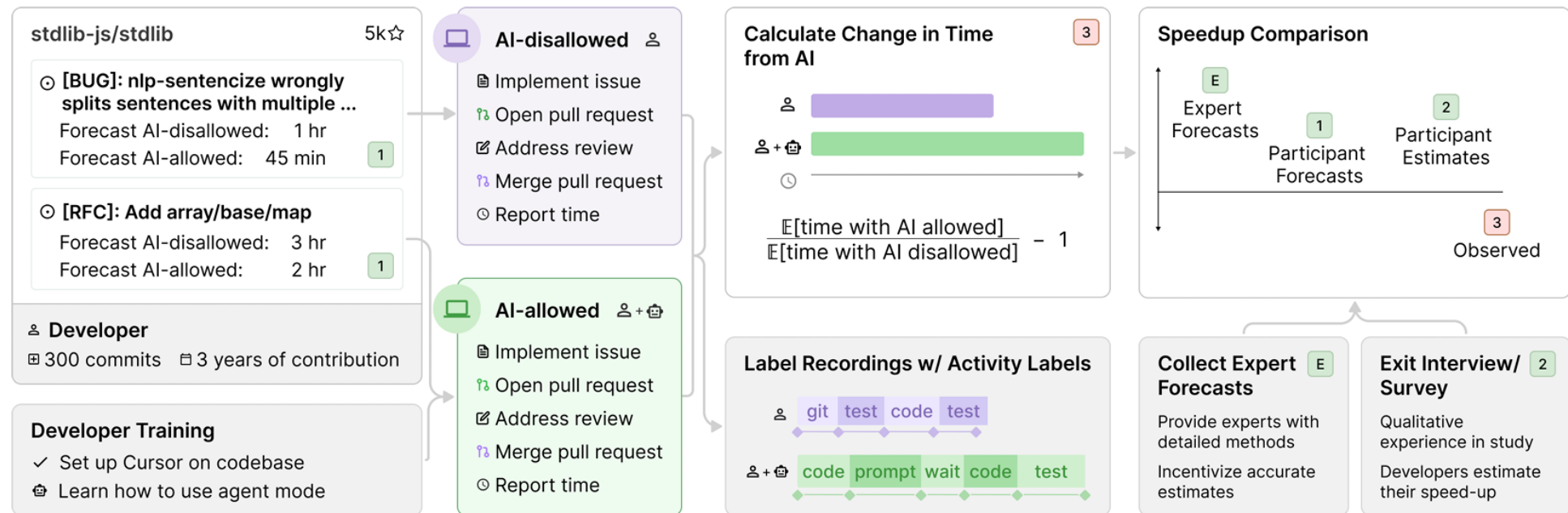


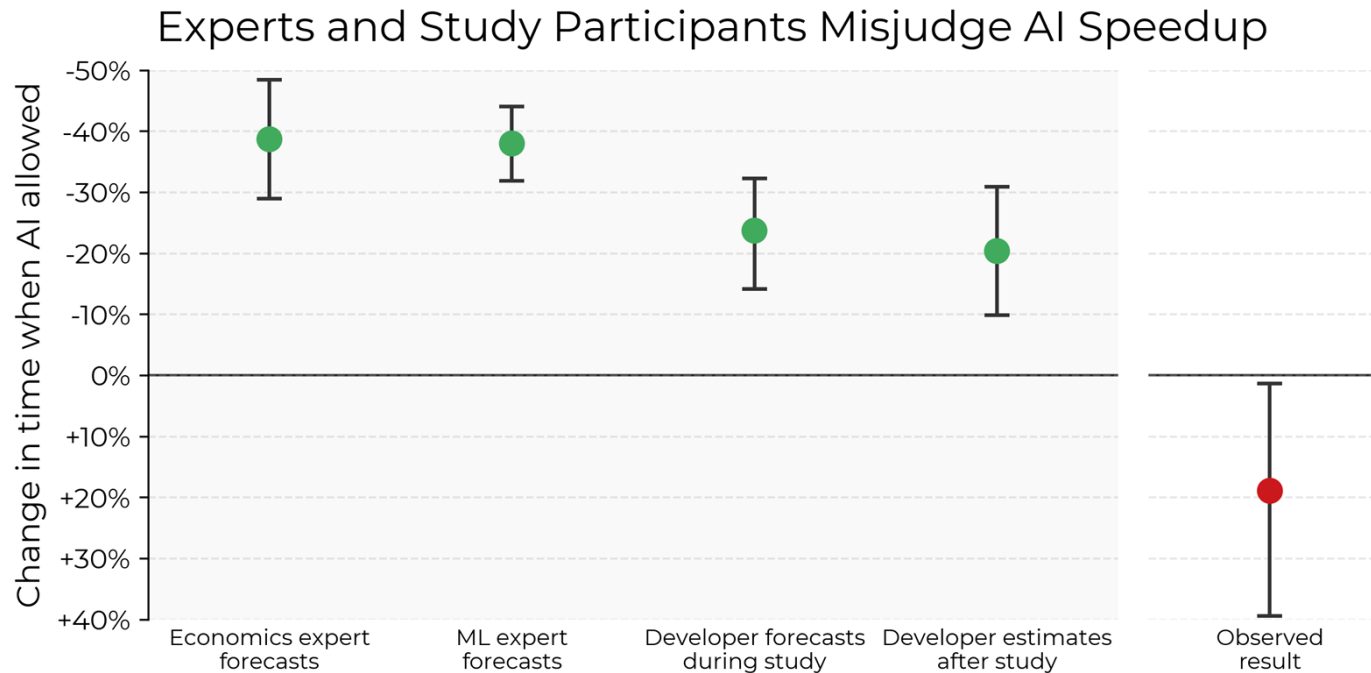
Figure 1: Taxonomy of LLM-based agents in software engineering.

# Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity



**Figure 2:** Our experimental design. Tasks (referred to as issues) are defined before treatment assignment, screen recordings let us verify compliance (and provide a rich source of data for analysis), and forecasts from experts and developers help us measure the gap between expectations and observed results.

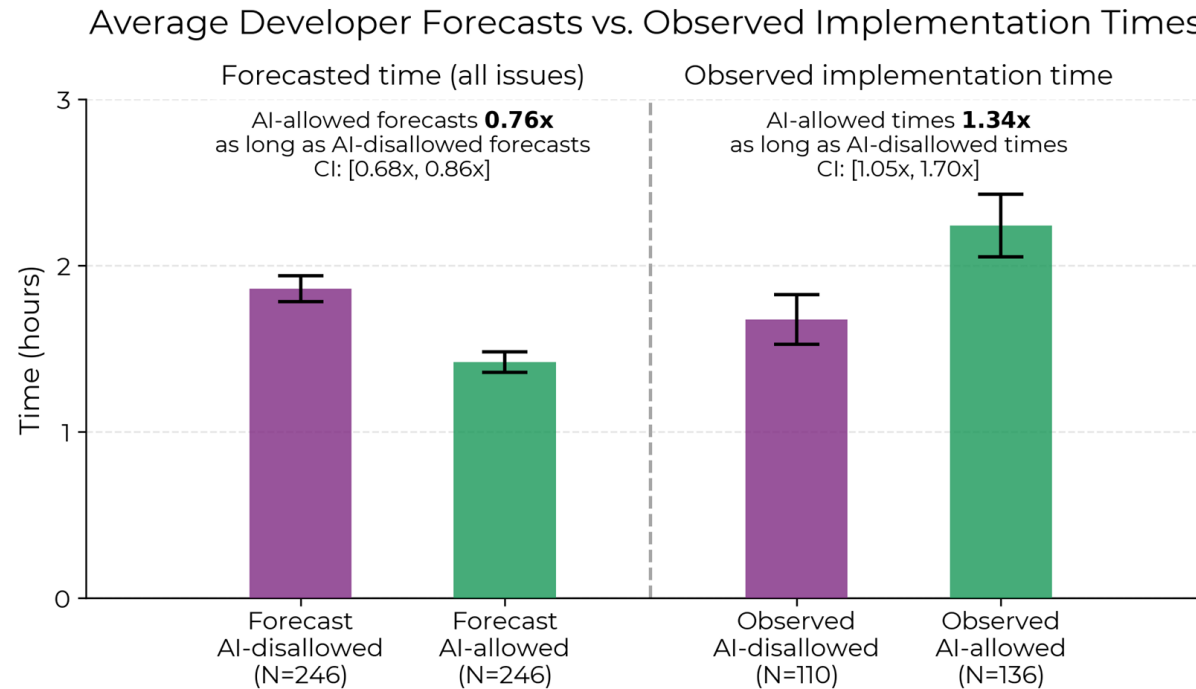
# Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity



**Figure 1:** Experts and study participants (experienced open-source contributors) substantially over-estimate how much AI assistance will speed up developers—tasks take 19% more time when study participants can use AI tools like Cursor Pro. See [Appendix D](#) for detail on speedup percentage and confidence interval methodology.

<https://arxiv.org/abs/2507.09089v2>

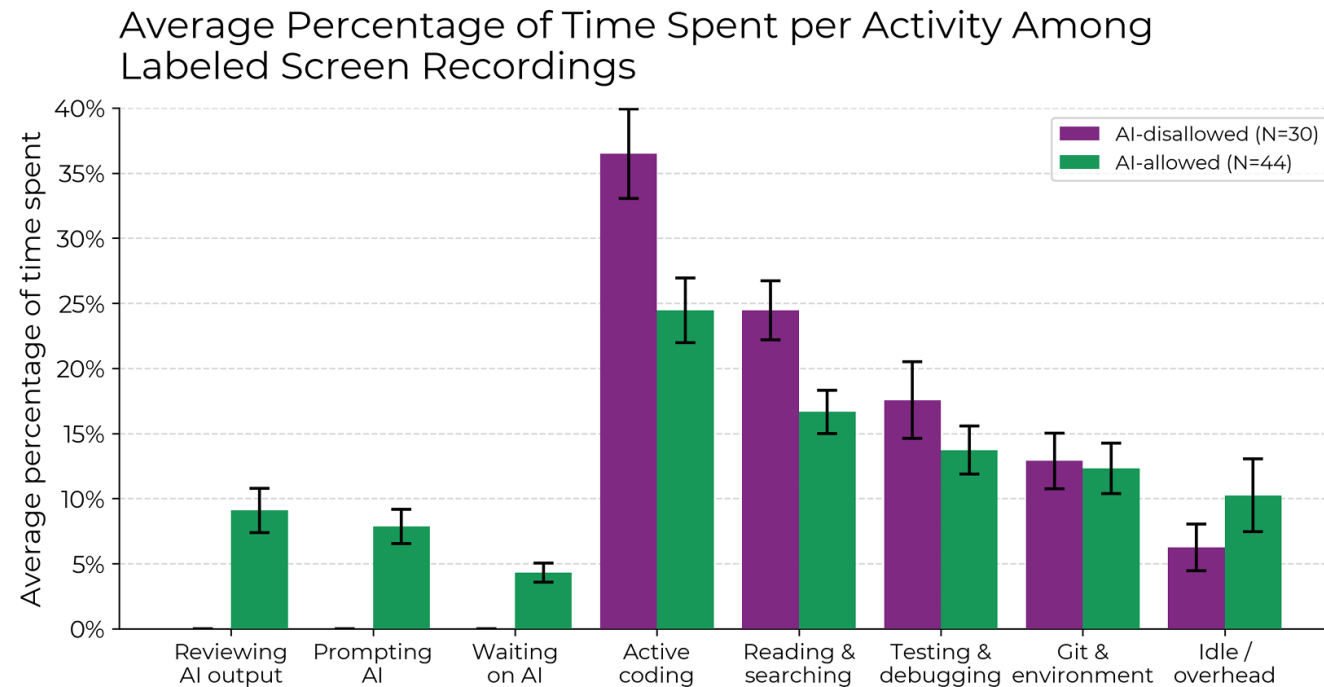
# Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity



**Figure 4:** Left: Raw average forecasted implementation times. Right: Raw average observed implementation times. The ratio of observed implementation times gives a more extreme slowdown estimate than regression-based estimates (Section D.1) because AI-allowed issues are forecasted (importantly, before treatment assignment) by developers to take slightly longer, which the regression corrects for. Both: Section D.5 describes confidence intervals around ratios of average times.



# Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity



**Figure 6:** On the subset of labeled screen recordings, when AI is allowed, developers spend less time actively coding and searching for/reading information, and instead spend time prompting AI, waiting on and reviewing AI outputs, and idle. *Figure 19* shows the absolute (average) minutes spent in each category, and *Figure 21* presents these results broken down into 27 fine-grained categories.

# Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity

Factor	Type	Relevant Observations
Over-optimism about AI usefulness (C.1.1)	⌚	<ul style="list-style-type: none"><li>• Developers forecast AI will decrease implementation time by 24%</li><li>• Developers post hoc estimate AI decreased implementation time by 20%</li></ul>
High developer familiarity with repositories (C.1.2)	👤	<ul style="list-style-type: none"><li>• Developers slowed down more on issues they are more familiar with</li><li>• Developers report that their experience makes it difficult for AI to help them</li><li>• Developers average 5 years experience and 1,500 commits on repositories</li></ul>
Large and complex repositories (C.1.3)	🗄️	<ul style="list-style-type: none"><li>• Developers report AI performs worse in large and complex environments</li><li>• Repositories average 10 years old with &gt;1,100,000 lines of code</li></ul>
Low AI reliability (C.1.4)	🗄️	<ul style="list-style-type: none"><li>• Developers accept &lt;44% of AI generations</li><li>• Majority report making major changes to clean up AI code</li><li>• 9% of time spent reviewing/cleaning AI outputs</li></ul>
Implicit repository context (C.1.5)	🗄️👤	<ul style="list-style-type: none"><li>• Developers report AI doesn't utilize important tacit knowledge or context</li></ul>

**Table 1:** Summary of factors that may a priori explain or contribute to slowdown, grouped by the state of evidence for or against their impact on the slowdown effect. 👤 are factors that raise human performance, 🗄️ are factors that limit AI performance, 🌱 are experimental artifacts that may bias/confound results, and ⌚ are factors that directly contribute to productivity losses.

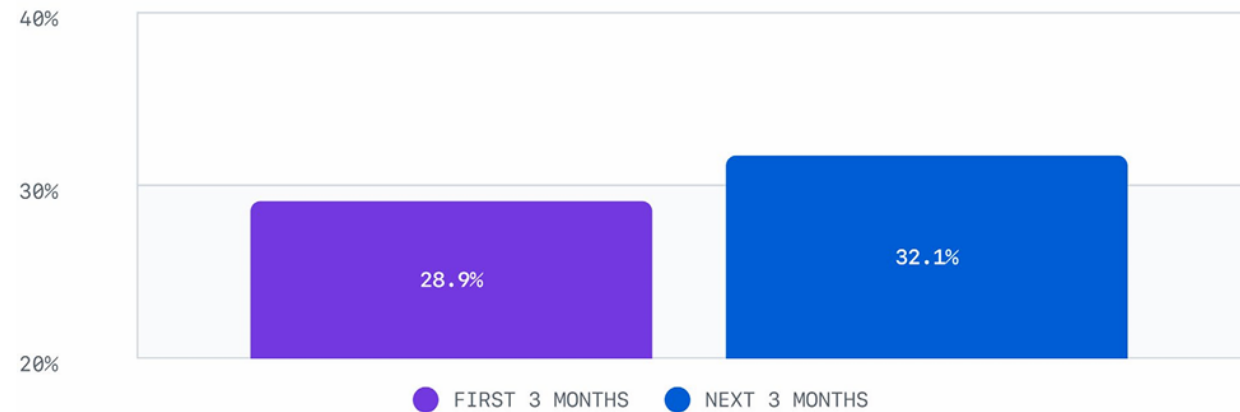
<https://arxiv.org/abs/2507.09089v2>

# Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle

By examining GitHub Copilot telemetry, we found that users accept an average of 30% of code suggestions, representing real productivity gains. We also discovered two additional findings:

- *The productivity benefit increases over time.*<sup>6</sup>
- *Less experienced developers have a greater productivity benefit.*<sup>7</sup>

Figure 1: GitHub Copilot's effect increases over time



# Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle

## The impact is large and accelerating

Figure 2: Increase in acceptance rate of GitHub Copilot recommendations over time

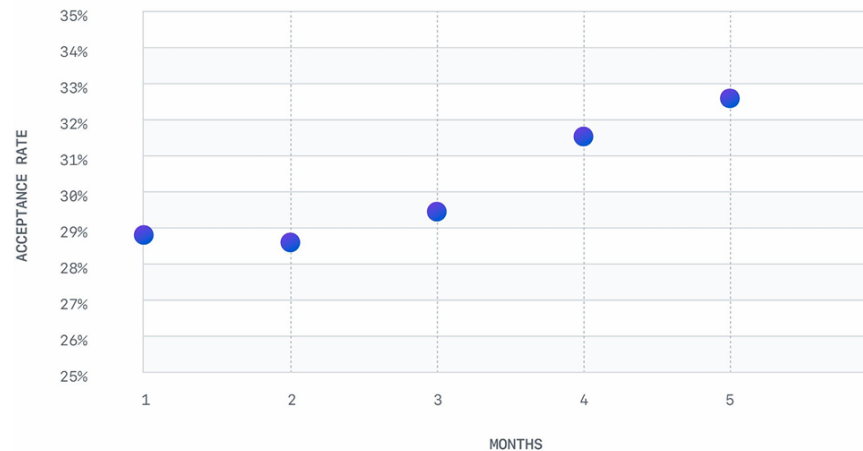
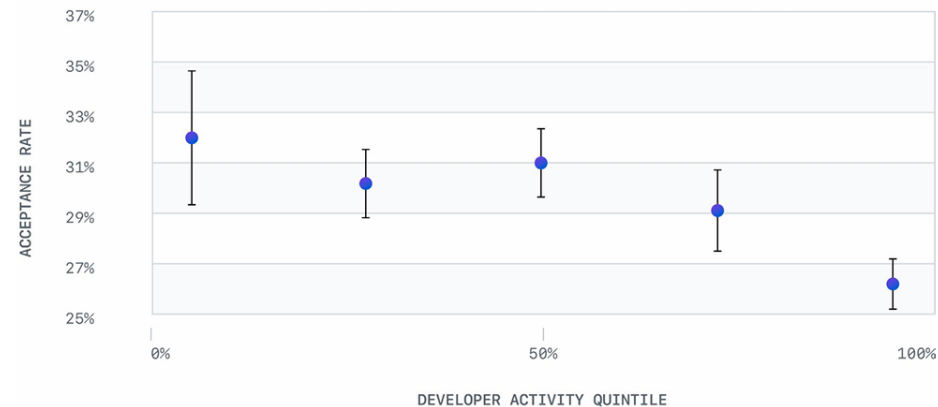


Figure 2 provides more detail on GitHub Copilot's impact over several months. Interestingly, the learning process does not appear to be diminishing after six months, which shows potential for even greater impact over time. This is important evidence for the positive impact of the tool, as developers continue to trust it more with increased usage and feedback on coding results. This is a novel, breakthrough finding. It suggests that GitHub Copilot has a large runway to continue its impact on developer productivity, as users become more accustomed to developing software with it.

<https://arxiv.org/abs/2306.15033>

# Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle

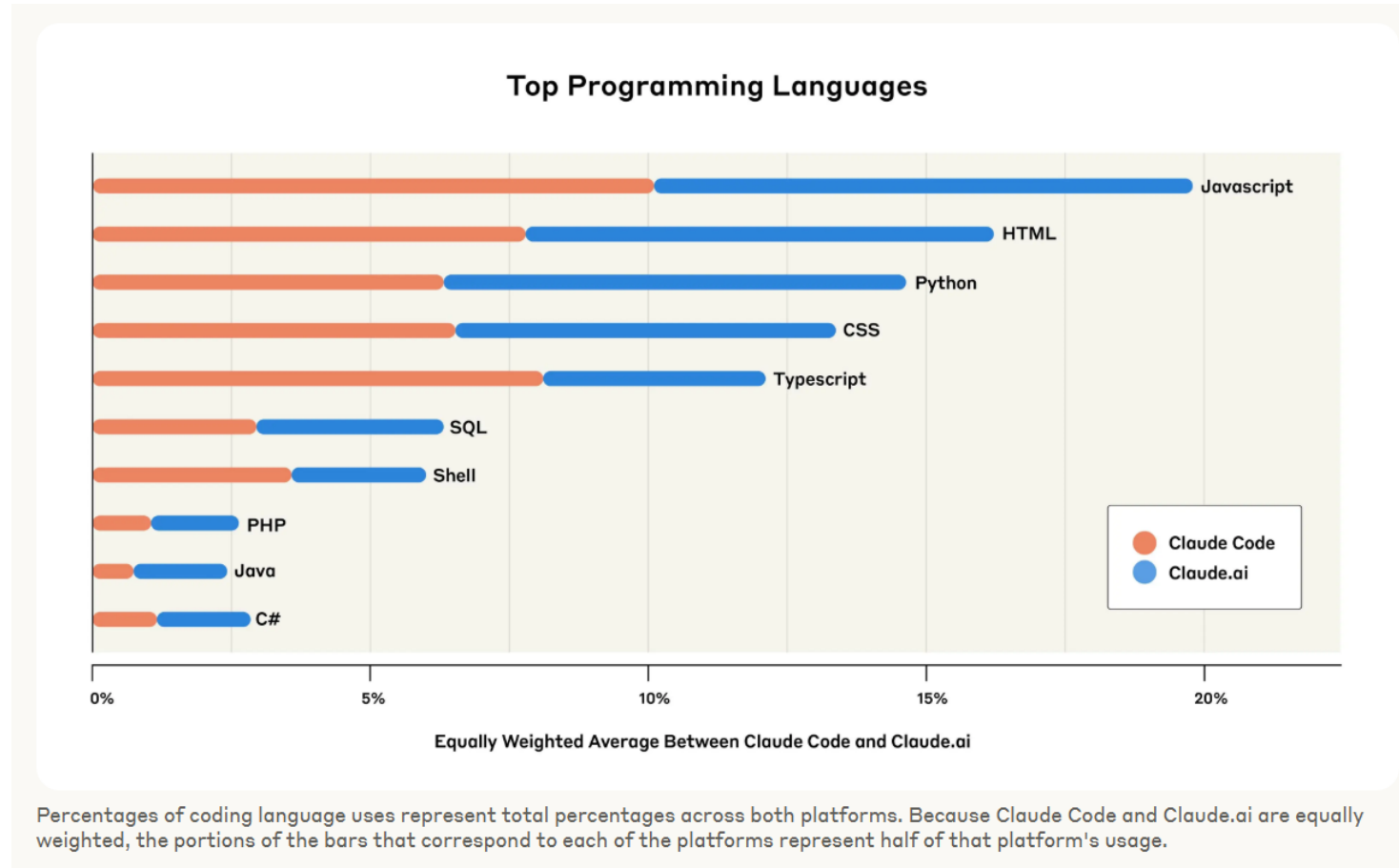
Figure 3: Developers with less experience benefit relatively more than more experienced developers



With a sample of nearly one million developers, the data demonstrates GitHub Copilot's potential to substantially impact productivity for the developer community. As developers continue to become fluent in prompting and interacting with AI, particularly with new models that allow natural language to power the development lifecycle, we anticipate that approximately 80% of code will be written with AI. We believe this will help democratize software development, allowing more people from non-technical backgrounds to build and shape the software that will continue to power the global economy.

<https://arxiv.org/abs/2306.15033>

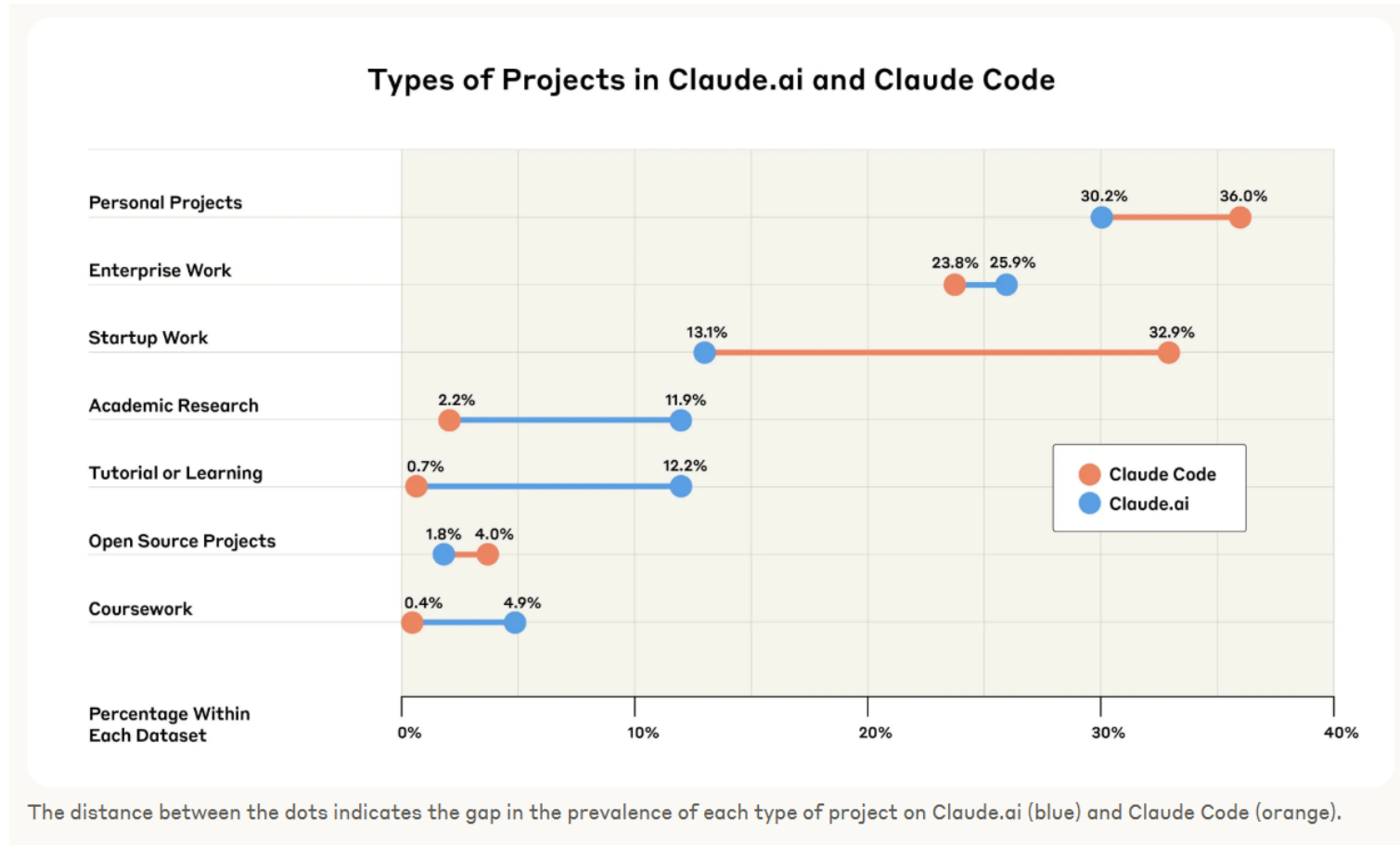
# Anthropic Economic Index: AI's impact on software development



<https://www.anthropic.com/research/impact-software-development>



# Anthropic Economic Index: AI's impact on software development



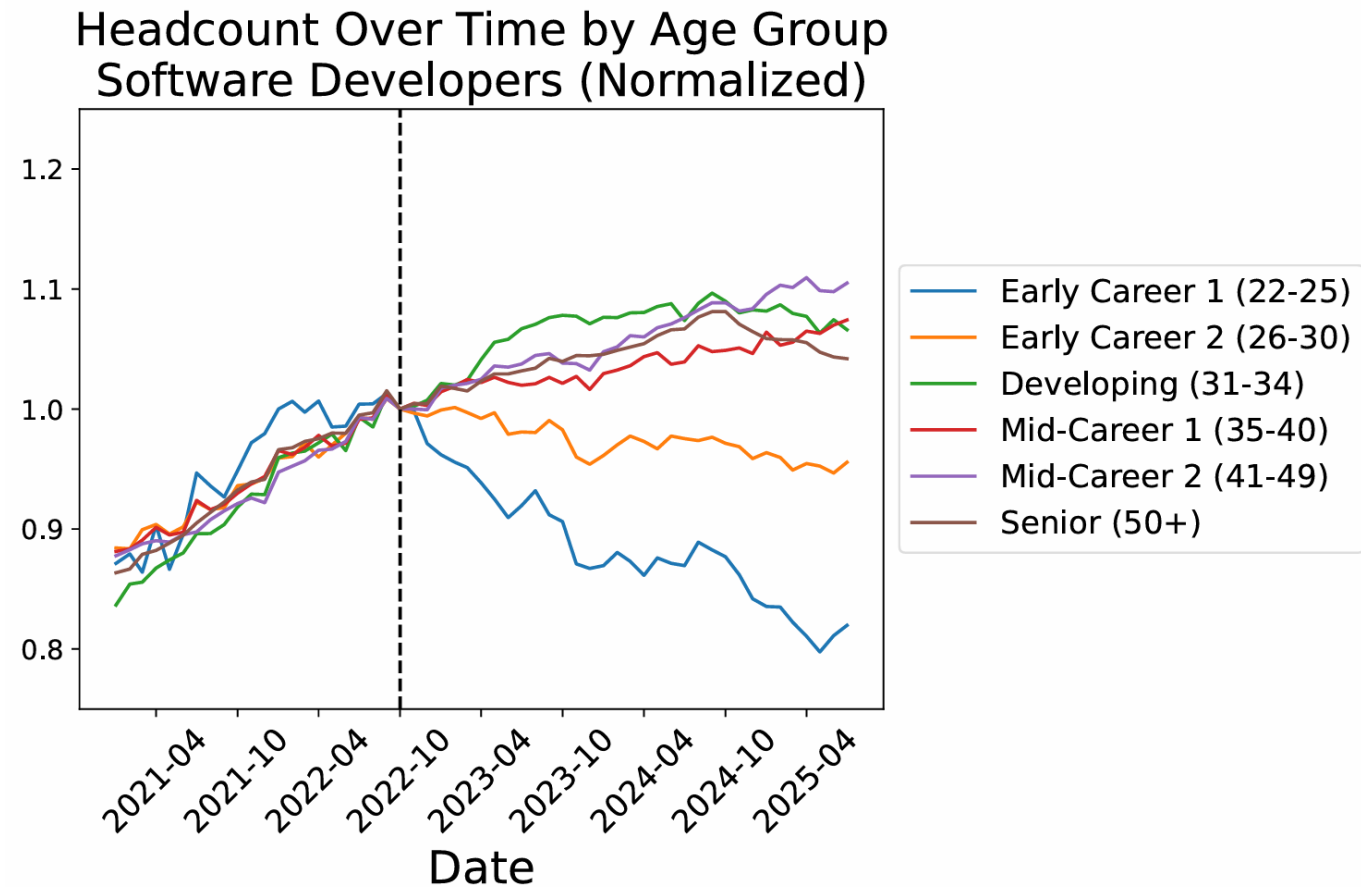
<https://www.anthropic.com/research/impact-software-development>

# Employment for computer programmers in the U.S. has plummeted to its lowest level since 1980—years before the internet existed

- Computer-programming employment in the U.S. has reached its lowest level since 1980.
- There are now fewer computer programmers in the U.S. than there were when Pac-Man was first invented—years before the internet existed as we know it. Computer-programmer employment dropped to its lowest level since 1980, according to the data from the Current Population Survey from the Bureau of Labor Statistics.
- There has been a 27.5% plummet in the 12-month average of computer-programming employment since about 2023—coinciding with OpenAI's introduction of ChatGPT the year before.

<https://fortune.com/2025/03/17/computer-programming-jobs-lowest-1980-ai/>

# Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence



# Что происходит? К чему готовиться? (Предположения)

Что происходит	Кого коснется?	К чему стоит готовиться?
<ul style="list-style-type: none"><li>Демократизации разработки (вследствие доступности ИИ-инструментов)</li></ul>	<ul style="list-style-type: none"><li>Не-разработчиков</li><li>Начинающих разработчиков</li><li>Стартаперов</li><li>(Рынок ИТ)</li></ul>	<ul style="list-style-type: none"><li>К возможности замены разработчиков на ИИ</li><li>К снижению спроса на junior разработчиков</li><li>К снижению спроса на разработку типовых решений (особенно в части Frontend)</li><li>К возможной деградации навыков</li></ul>
<ul style="list-style-type: none"><li>Снижение спроса на junior разработчиков</li><li>Снижение спроса на разработку типовых решений (особенно в части Frontend)</li></ul>	<ul style="list-style-type: none"><li>Junior разработчиков</li></ul>	<ul style="list-style-type: none"><li>Нарабатывать опыт (open source, хакатоны, стажировки и т.п.)</li><li>Учиться делать бизнес-логику, нетиповые/специализированные решения</li></ul>
<ul style="list-style-type: none"><li>Рост объемов низкокачественного (в т.ч. небезопасного) кода в целом</li></ul>	<ul style="list-style-type: none"><li>Владельцев продуктов</li><li>Опытных разработчиков</li></ul>	<ul style="list-style-type: none"><li>К переписыванию/рефакторингу кода</li><li>К повышению спроса на специалистов по безопасности и разработчиков</li><li>К проверке зависимостей/цепочек поставок</li></ul>
<ul style="list-style-type: none"><li>Повышение стоимости подготовки middle/senior специалистов</li><li>Повышение спроса на Senior разработчиков</li></ul>	<ul style="list-style-type: none"><li>Senior разработчиков</li></ul>	<ul style="list-style-type: none"><li>К росту ожиданий, что Senior'ы должны уметь разрабатывать с ИИ-инструментами и минимизировать создаваемые ими проблемы, а junior'ы не нужны</li></ul>
<ul style="list-style-type: none"><li>Возможность автоматизации типовых задач с ИИ</li></ul>	<ul style="list-style-type: none"><li>Опытных разработчиков</li></ul>	<ul style="list-style-type: none"><li>К использованию ИИ для минимизации тех. долга/автоматизации рутины</li></ul>
<ul style="list-style-type: none"><li>Проблема с подготовкой нового поколения разработчиков</li></ul>	<ul style="list-style-type: none"><li>(Весь рынок ИТ, но не сразу)</li></ul>	<ul style="list-style-type: none"><li>К аналогу «демографической ямы» для разработчиков на горизонте 3+ лет</li></ul>

# Разработчик разрабатывает ПО

1

2

3

Сформулируем вопросы:

1. Может ли ИИ заменить разработчиков?
  - Где ИИ уже может, а где нам нужен естественный интеллект?
2. Как изменится разработка?
  - Что происходит? К чему готовиться в этой связи.
3. Как изменится ПО?
  - Out of scope / вне текущего доклада.

# Резюме

- Искусственный интеллект, нельзя естественный
  - Быстрое прототипирование
  - Ускорение написания типовых сценариев
  - Разработка с низкой ценой ошибки
  - Разработка под руководством Middle/Senior+ с пониманием, что нужно
  - Автоматизация рутинных операций
- Искусственный интеллект нельзя, естественный
  - Работа CPO (управление продуктом)
  - Понимание бизнес-логики, «сжатого представления» / «цифрового двойника» объектов/процессов
  - Разработка архитектуры
  - Реализация нетипичных сценариев
  - Разработка с высокой ценой ошибки, в т.ч. обеспечение безопасности



# Искусственный интеллект нельзя, естественный

(Но там, где нужно быстро прототипировать и не  
очень высока цена ошибка, можно и искусственный)