

1人のAIで十分か

「強い1人」説を、本番の開発チームで確かめる

Is One Agent Enough? Testing the Strong Single-Agent Hypothesis on a Production Code-Generation Team

湯村 信司 (Shinji Yumura) AQUA合同会社 2026年10月12日

この報告が確かめた論文

J. Xu ほか「Rethinking the Value of Multi-Agent Workflow: A Strong Single Agent Baseline」
arXiv:2601.12307 (2026年1月)

P. Jwalapuram ほか「The Illusion of Multi-Agent Advantage」 arXiv:2606.13003 (2026年6月)

「1人のAIで、AIのチームと同じ出来になる」という2つの主張を、本番の開発チームで、同じお題・同じ時刻に比べて確かめた。

85.4 / 77.1%

よく考える1人（1回で全部を書く）と、5人のチームの審査の合格。差ははっきりしない

59.4%

1人が同じ会話で5つの役を順に演じる形。5人よりはっきり下がった ($p = 0.013$)

+30%

よく考える1人の料金（5人に比べて）。AIを呼ぶ回数は6.2回 → 2.2回 に減る

ABSTRACT

Two 2026 papers argue that a single LLM agent can match multi-agent workflows: Xu et al. show that a homogeneous workflow can be simulated by one agent playing each role in turn within a single conversation, and Jwalapuram et al. find that, at equal compute, automatically designed multi-agent systems rarely beat a single agent. We tested both ideas on the five-agent team (planner, three file-type specialists, reviewer) that builds React apps in YUA's Dev Mode, against two single-agent variants, on a 24-task benchmark with 96 generations per condition run concurrently (Gemini 3.5 Flash-Lite). A single agent that planned and wrote all files in one call with a high thinking level passed an LLM judge in 85.4% of runs versus 77.1% for the team ($p = 0.20$), using 2.2 instead of 6.2

model calls but costing 30% more and finishing 14 seconds later; its first versions failed at runtime more often and relied on our in-browser auto-repair. A single agent role-playing the five roles in one conversation, as in Xu et al., dropped to 59.4% ($p = 0.013$), with twice the input tokens and less than half the thinking. In our setting, “one agent is enough” held when the single agent was allowed to think more, but not when it merely took turns as the team.

要旨

2026年の2つの論文が、「1人のAIで、AIのチームと同じ出来になる」と主張している。Xu ほかは、同じモデルでできたチームは、1人のAIが同じ会話の中で役を順に演じれば同じ出来になると示した。Jwalapuram ほかは、計算をそろえると、自動で設計したチームはほとんど1人に勝てないと報告した。

本稿では、YUA の開発モードで React のアプリを作っている5人のチーム（計画・ファイルの種類ごとの3人・見直し）と、2つの「1人」の形を比べた。条件は、24題で各条件96回、3つを同じ時刻に測定、作るAIは Gemini 3.5 Flash-Lite である。

よく考える設定（high）で、1回で全ファイルを書く1人は、AIの審査役の合格が85.4%で、5人の77.1%と差がはっきりしなかった（ $p = 0.20$ ）。AIを呼ぶ回数は6.2回から2.2回に減ったが、料金は3割高く、仕上がりは14秒遅かった。最初の版で動かないことも多く、画面の自動の直しに頼っていた。一方、Xu ほかの形（1人が同じ会話で5つの役を順に演じる）は59.4%に下がった（ $p = 0.013$ ）。入力は2倍に増え、考えた分は半分以下だった。私たちの場面では、「1人で十分」は、1人によく考えさせたときには成り立ったが、役を順に演じさせただけでは成り立たなかった。

1 はじめに

前回の報告（003）では、5人のチームを3人に減らすと出来が下がった。世の中の流れはその逆で、2026年には「チームより強い1人」を示す論文が注目を集めている。

- **Xu ほか [1]**：同じモデルのAIでできたチームは、1人のAIが同じ会話の中で役を順に演じれば再現でき、出来も同じになる。
- **Jwalapuram ほか [2]**：計算（料金）をそろえると、自動で設計されたチームは、1人が何度か考えて多数決をとる方法（CoT-SC）に負けることが多い。ただし、専門家が設計したチームは別である。

私たちのチームは、人が設計し、本番で毎日使われている。本稿は、この2つの主張を本番のチームで確かめる。論文の主張をそのまま試すために、2つの「1人」の形を用意した。

2 比べた3つの形

形	中身	考える深さ
F：5人（本番）	計画 → 見た目 → 型と道具 → 画面の部品（順に、先に書いたものを見ながら） → 見直し。見直しが基準に届かなければ直しを頼む（3回まで）	medium
S1：1人・1回	1人が頭の中で設計し、全ファイルを1回で書く。実行環境の説明と出来の良さの決まりはFと同じ	high
S2：1人が役を演じる	Xu ほかの形。1人が同じ会話の中で、Fと同じ指示文を使って、計画→ 見た目→型→部品→見直し（直しは1回まで）を順に演じる	medium

表1 比べた形。作るAIはすべて gemini-3.5-flash-lite。どの形の結果も、同じ画面で動かし、エラーがあれば同じ仕組みで自動で直す（2回まで）。

S1 の考える深さを high にしたのは、Jwalapuram ほかの「計算をそろえる」考え方に近づけるためである。呼ぶ回数が減るぶん、1回で深く考えさせた。

3 測り方

- **お題**：これまでと同じ24題。それぞれ2回ずつ頼み、これを2回くり返した（各条件96回）。3つの形を別々の場所で立ち上げ、同じ時刻に測った。
- **審査役**：gemini-3.8-flash に最後の版の写真とコードを見せ、0～2点で3回採点させた。その真ん中の点を使った。
- **記録**：AIを呼んだ回数、トークン（入力・出力・考えた分）、料金（入力 \$0.30・出力と考えた分 \$2.50、100万トークンあたり）、時間、自動の直しの回数。

4 結果

指標	F : 5人	S1 : 1人・1回	S2 : 1人が役を演じる
審査役の合格（2点）	74/96（77.1%）	82/96（85.4%）	57/96（59.4%）
1回目・2回目の合格	37・37	44・38	26・31
動いた	96/96	94/96	96/96
AIを呼んだ回数（1題あたり）	6.21	2.21	5.97
入力トークン	14,360	2,350	27,582
出力・考えた分のトークン	4,167・2,747	6,939・3,997	3,286・1,150
料金（100題あたり）	\$2.16	\$2.81	\$1.94
最初の版まで・仕上がるまで（中央値）	31秒・46秒	43秒・60秒	18秒・33秒
自動の直しが入った回	2	19	4

表2 結果（各96回）。合格の差は、FとS1が $p=0.20$ 、FとS2が $p=0.013$ （フィッシャーの正確確率検定）。本物のグラフ部品の使用と日本語の画面は、3つともほぼ100%。

S1（よく考える1人）は、5人と同じかそれ以上だった。差は統計的にははっきりしないが、2回の測定のどちらでも5人を上回った。AIを呼ぶ回数は3分の1になり、入力トークンも6分の1になった。その代わり、出力と考えた分が増えたため、料金は3割高く、仕上がりは14秒遅かった。また、最初の版がエラーで動かず、画面の仕組みが自動で直した回が19回あった（5人は2回）。

S2（1人が役を演じる）は、はっきり下がった。速く安かったが、合格は59.4%だった。会話が積み重なるため、入力は5人の約2倍になった。考えた分のトークンは、5人の半分以下（1,150と2,747）だった。

5 考察

5.1 「1人で十分」は、考える量しだい

3つの形を、前回の報告（003）の3人の形と並べると、筋が見える。考える深さ medium で1人が全部を書く形（003のNoa）は68.8%で、5人より下だった。同じ1人でも、highで考えさせると（S1）5人に並んだ。チームの値打ちの一部は、役を分けること自体ではなく、全体として考える量を増やすことにあったと考えられる。Jwalapuram ほかの「計算をそろえると、チームの得は消える」という主張と、方向は合っている。

5.2 役を演じ分けるだけでは、チームにならない

Xu ほかの形（S2）が下がった理由として、次の2つが考えられる。

- 同じ会話の中で、前の役の答え（計画のJSON、書いたファイル）が次々に積み重なり、指示が埋もれた。入力が増えたことが、これを示している。
- 全体として考えた量が少なかった（考えた分のトークンは5人の半分以下）。

Xu ほかは、同じ会話を続けることで、計算の使い回し（KV キャッシュ）による効率の得を示した。私たちが使ったAPIの呼び方では、この得はほとんど生かせない。本番で使う呼び方によって、同じ形でも結果が変わりうる。

5.3 1人の形は、まわりの仕組みに頼る

S1は最初の版でエラーが出ることが多く、画面の側の見張りや自動の直し（002）が、その分を受け止めていた。チームの中の見直し役の仕事を、画面の外側の仕組みが代わりにしたことになる。「1人で十分」かどうかは、その1人のまわりにどんな確かめの仕組みがあるかと切り離して語れない。

5.4 本番の選び方

出来を最優先するならS1、速さと料金のつり合いなら5人、速さだけならS2（出来は下がる）、という選び方になる。ただしS1と5人の差ははっきりしないので、本番は5人のままとし、S1は条件を変えて測り直す（お題を増やす、料金をそろえる）。

6 限界

- 規模：各96回。S1と5人の差は統計的にははっきりしない。

- **計算は完全にはそろえていない**：S1 は全体のトークンが少ない一方、料金（出力と考えた分）は多い。考える深さとチームの形を同時に変えている。
- **Jwalapuram ほかの CoT-SC**（何度か作って多数決）は試していない。コードの「多数決」には別の工夫が要するため、今後の課題とする。
- **審査**：作るAIと審査役が同じ系統（Gemini）で、人による評価はない。審査の合格は回によって十数ポイント揺れる（002）。比べたのはすべて同じ時刻に測ったものどうしである。
- **お題**：日本語の小さなアプリ（2〜4ファイル）だけ。
- **5人のチームの料金**：報告 005 で見つけた、見直しの点を誤って書きかえる不具合のむだを含む（96回中9回）。この9回を除くと、100題あたり \$1.92 になり、S1 との料金の差は表2より大きい。出来への影響は小さいと考えられる（この不具合が起きた回でも、審査の合格は下がっていなかった）。

7 関連研究

役を分けたAIのチームでソフトウェアを作る試みとして MetaGPT [3] や ChatDev [4] がある。これに対して、1人のAIで十分とする主張が [1][2] である。本稿は、人が設計して本番で使われているチームを相手に、[1] の形と、[2] の「計算を1人に集める」考えを試した。前回までの報告 [5][6] とあわせて、同じ時刻に比べる測り方をとった。

8 まとめ

本番の5人のチームに対して、「強い1人」説を2つの形で確かめた。よく考える1人は、5人と同じかそれ以上の出来だった。呼ぶ回数は減ったが、料金と時間は増え、まわりの自動の直しに頼っていた。一方、1人が同じ会話で役を順に演じる形は、はっきり出来が下がった。「1人かチームか」より、「全体でどれだけ考えさせるか」と「まわりにどんな確かめがあるか」が、出来を左右していた。

参考文献

1. J. Xu, A. Koesdwiady, S. Bei, et al. “Rethinking the Value of Multi-Agent Workflow: A Strong Single Agent Baseline.” arXiv:2601.12307, 2026.
2. P. Jwalapuram, H. Lin, C. Li, et al. “The Illusion of Multi-Agent Advantage.” arXiv:2606.13003, 2026.
3. S. Hong, M. Zhuge, J. Chen, et al. “MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework.” *ICLR*, 2024. arXiv:2308.00352.
4. C. Qian, W. Liu, H. Liu, et al. “ChatDev: Communicative Agents for Software Development.” *ACL*, 2024. arXiv:2307.07924.

5. 湯村 信司. 「AIの開発チームは何人がいいか」 . AQUA Research Technical Report 003, 2026.
6. 湯村 信司. 「AIが作ったアプリを、AIの審査で正しく測れるか」 . AQUA Research Technical Report 002, 2026.