# MineMate: Multi-Agent Emergent Capability with Humans

An Embodied Study of Many-to-Many Human–Agent Teaming in Minecraft

Minkyu Kang
UNIST (Ulsan National Institute of Science and Technology)
Ulsan, Republic of Korea
perelman@unist.ac.kr

## ABSTRACT

Foundation-model agents have moved well beyond question answering into *acting* on a user's behalf: large language model (LLM) agents now operate software, and vision–language–action (VLA) models are beginning to control robots. As single-agent capability matures, the binding constraint is shifting from whether one agent can act to whether *people and many agents can work together*. Yet large-scale, long-horizon studies of the many-human × many-agent regime are largely absent, because the most relevant embodied platforms, real robots, are too costly and unsafe for such ablations. I present MineMate, a human–agent teaming testbed built in Minecraft, and report a main experiment ($N$=4) in which people direct teams swept over Human:Agent ratios from 1:1 to 1:16 on open-ended, long-horizon tasks. The bottleneck was not agent capability but coordination: team yield rose with the agent count only for decomposable, parallel work and saturated as soon as a task required agents to collaborate. Left to communicate freely, capable agents talked in circles, stalled, and lost track of the goal. I trace these breakdowns to three coupled design variables: inter-agent communication structure, the observation channel shared by human and agents, and how the human addresses the team. I use these to organize a design space and a study agenda for many-to-many human–agent teaming. My central finding is that embodied LLM agents are *strong as singletons but weak at collaboration*, and that the human's role is rapidly becoming that of a coordinator of emergent, only-partially-legible teammates. Project page: https://mk040412.github.io/MineMate/.

## CCS CONCEPTS

• **Human-centered computing** → **Interaction paradigms**; **Empirical studies in collaborative and social computing**; • **Computing methodologies** → *Multi-agent systems.*

## KEYWORDS

Human–agent interaction, multi-agent systems, embodied agents, large language models, human–AI collaboration, Minecraft, coordination

## 1 INTRODUCTION

End users now interact with systems powered by large language models (LLMs) *directly*, and well beyond question-and-answer. These systems are increasingly *agentic*: they perceive, plan, and act on the user's behalf across long, multi-step workflows. Such agents read natural-language instructions to operate web browsers, mobile apps, and desktop software through their graphical user interfaces (GUIs) [22]; they call external tools and APIs, retrieve and reason over documents, and write, run, and debug code; and they chain these abilities into multi-step tasks carried out on the user's behalf. The same paradigm is also beginning to cross into the physical world, where vision–language–action (VLA) models let robots execute language-conditioned manipulation and navigation [3, 10].

Despite this shift, my understanding of *many-to-many human–agent cooperation*, how multiple people and many autonomous agents coordinate as a joint team, has not kept pace with these capabilities. Most multi-agent LLM research to date has focused on *digital* agents, and particularly on code- and software-oriented ones: frameworks such as CAMEL [11], AutoGen [21], MetaGPT [8], and ChatDev [14] compose role-playing LLM agents to write software and solve reasoning tasks, and a growing survey literature catalogues their coordination patterns [6]. These agents are typically *disembodied*, and the interaction is largely *agent–agent*: a human supplies one up-front goal and then steps out of the loop. In *embodied* multi-agent settings, the dominant line of work has instead been multi-agent reinforcement learning [15, 23], where the emphasis is on training agents to optimize a fixed, predefined objective, often with no human in the loop, rather than on interacting with a user through high-level instructions to accomplish open-ended goals. As a result, the embodied, human-in-the-loop case remains comparatively underexplored: to my knowledge, no prior work studies many-to-many human–agent teaming that scales the number of both humans *and* embodied agents while keeping people in the loop. Such a study is costly and unsafe to run on real robots at this scale, and existing embodied multi-agent systems [1, 20] largely keep the human outside the team.

I argue that Minecraft is a strong surrogate for this gap. It is an open-ended, embodied, partially observable world with rich long-horizon tasks, and recent work establishes it as a legitimate testbed for multi-agent LLM collaboration [20]. No prior work, however, uses Minecraft to ablate the *human* side of the team: how a team's composition ratio, communication structure, and the spatial role assignment between humans and agents shape teaming efficiency and the user's experience. MineMate targets exactly this gap. Concretely, it is a human-subjects study in which real participants play Minecraft alongside teams of LLM-driven agents: I vary the team's composition and how it is coordinated, and observe, through gameplay, interviews, and session logs, how efficiently the human–agent team works and how people prefer to direct it.

I report a main experiment ($N$=4) that instantiates this design space: each participant directs a Minecraft team swept over Human:Agent ratios from 1:1 to 1:16 on two open-ended subtasks, choosing how to configure the team along each axis. The bottleneck I observed was rarely the agents' raw capability: individual

agents followed free-form instructions well. What broke down instead was coordination. Team yield grew with the agent count only for work that decomposes into independent parallel effort, and saturated as soon as a task required agents to collaborate; left to communicate freely, individually competent agents talked in circles, stalled, and lost track of the goal. These are interaction and coordination failures rather than capability failures, and a natural object of study for HCI.

**Contributions.** Unlike multi-agent LLM and reinforcement-learning work that keeps the human out of the loop, this paper studies teaming with people *inside* the team across the many-human × many-agent regime, and identifies when such teams work best. Using MineMate, I contribute:

- the dynamics of *model communication* and *user–model interaction* as humans and agents scale;
- which mutual *observations* between user and agents actually help; and
- the team configuration under which task performance is best.

## 2 RELATED WORK

My work sits at the intersection of two lines of research: agents built and studied inside Minecraft, and the study of how people interact and collaborate with capable agents. I review each in turn.

### 2.1 Embodied Agents and Minecraft as a Testbed

Minecraft has become a standard proving ground for embodied, open-world agents, because it couples an effectively unbounded task space with partial observability and genuinely long horizons. Early infrastructure made this practical: MineRL released a large simulator-paired dataset of human Minecraft demonstrations [7], and MineDojo [4] added thousands of open-ended, language-specified tasks plus an internet-scale knowledge base of videos, wikis, and forum posts. These benchmarks established Minecraft as a venue where open-ended *competence*, not narrow task success, is what is measured.

Large language models turned that competence into instruction-following behavior. Voyager [18] showed that an LLM agent can explore Minecraft under an automatic curriculum and accumulate a growing library of reusable, named skills, improving over its lifetime without gradient updates. JARVIS-1 [19] decomposed natural-language instructions from goal to high-level plan to low-level control, backed by a multimodal memory, while GITM [24] reached high success rates on long crafting chains by mapping goals onto a text-based knowledge and memory store rather than raw pixels. These systems supply the per-agent ingredients MineMate reuses: skill libraries, hierarchical instruction decomposition, and explicit memory. They also build on general cognitive-architecture ideas, notably the recency/importance/relevance memory stream and reflection step of Generative Agents [13] and the verbal self-correction of Reflexion [17]. MineMate's single-agent loop (Section 4.1) is assembled from these components.

Most relevant to this paper is the move to *many* agents. Project Sid [1] ran many-agent Minecraft "civilizations" and reported emergent role-specialization, division of labor, and even social behaviors such as deception, none of which were explicitly programmed. MindAgent [5] framed multi-agent coordination as an *emergent gaming-interaction* problem and, beyond its CuisineWorld cooking benchmark, ported the same scheduling infrastructure *into Minecraft*, where two in-game agents (Alex and Steve) cooperate on cooking tasks with a human in the loop, while TeamCraft [12] turned embodied multi-agent collaboration in Minecraft into a benchmark of thousands of multi-modal task variants for vision–language–action agents. The MINDcraft framework [20] then demonstrated that Minecraft supports scientifically valid study of multi-agent LLM collaboration on long-horizon tasks, and identified natural-language communication as the primary bottleneck *between* agents. All either keep the human outside the team or confine it to a single, constrained game; MineMate instead puts multiple humans *inside* the team and asks how it behaves as both humans and agents scale.

### 2.2 Human–Agent Interaction and Multi-Agent Collaboration

Foundation-model agents increasingly *act* rather than answer: GUI agents operate browsers, apps, and desktop software for the user [22], and vision–language–action models such as OpenVLA [10] and $\pi_0$ [3] carry the same paradigm into the physical world. As single-agent competence grows, the open question is how agents and people coordinate, which MineMate studies where coordination is concrete (a shared physical world) rather than conversational.

When several agents are combined, the prevailing approach is to let them talk to one another: role-playing dialogue in CAMEL [11], a programmable framework of conversable agents in AutoGen [21], and a software team passing artifacts down an assembly line in MetaGPT [8] and ChatDev [14], with surveys now cataloguing their communication and orchestration patterns [6]. These systems are overwhelmingly digital and *agent–agent*: a person supplies one up-front specification and then leaves the loop. In embodied or simulated-physical settings, the dominant multi-agent line is instead reinforcement learning. Benchmarks such as the StarCraft Multi-Agent Challenge [15] and the broader MARL literature [23] train teams of agents to coordinate from local observations toward a fixed, predefined objective; this produces strong cooperative policies, but it optimizes a reward rather than following a person's high-level, in-the-moment instructions, and the human is typically absent from the loop entirely.

HCI has long studied how people should share control with capable automation: Horvitz's principles of mixed-initiative interaction [9] cast the human and an automated service as continually negotiating who acts when, and the guidelines for human–AI interaction [2] distilled recurring needs around making a system's capability and behavior legible. For agent *teams*, a survey argues that full autonomy suits few deployments and maps out the dimensions to study [25], while Schömbs et al. [16] show that even the *one-human, many-agents* case strains transparency, orchestration, and cognitive load, shifting the user from primary actor to *composer*. Empirical work in the genuinely many-human, embodied, human-in-the-loop regime nonetheless remains scarce; MineMate targets that gap and observes which structures users actually *choose* at scale, not which they say they prefer.

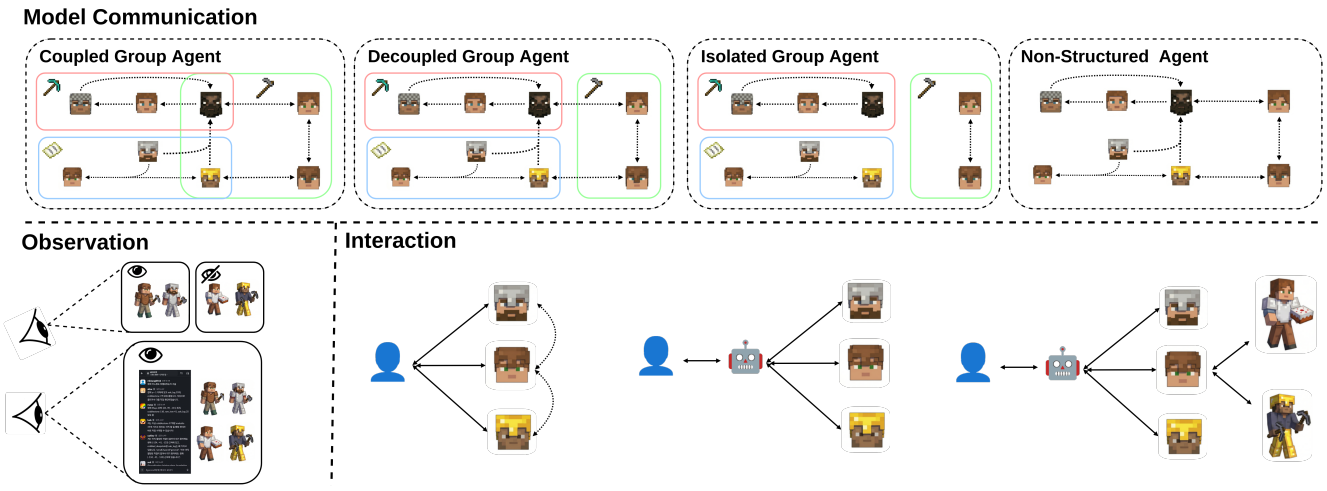


**Figure 1: The MineMate human–agent teaming design space, spanning three axes. Top (Model Communication): how agents coordinate among themselves during task execution, shown left to right over four regimes of *decreasing* imposed structure. *Coupled*: one group shares a joint plan and context and acts as a single unit. *Decoupled*: groups exchange only summaries through a representative. *Isolated*: groups never communicate across boundaries, so only the user bridges them. *Non-Structured*: no grouping, a single many-to-many web where any agent may address any other. Bottom left (Observation): the modality through which each party perceives the process. The partial, first-person in-game *scene* view (eye / crossed-eye icons mark who can and cannot see a given event) versus the full-but-shallow *text* channel of what agents and users report. Bottom right (Interaction): how the user's command enters the team and propagates, shown left to right. *Direct Interaction*: the user addresses each agent individually. *Centralized Interaction*: the user speaks to one coordinator agent that relays and allocates work to a fixed set of agents. *Hierarchical Centralized Interaction*: that coordinator in turn delegates through further agents, extending the chain of command down a multi-level hierarchy.**

## 3 THE HUMAN–AGENT TEAMING INTERACTION DESIGN SPACE

Preliminary pilot sessions motivated the axes I vary. Their lesson was simple: a single agent was rarely the problem, since each acted with enough agency that participants felt little was missing when directing one, but as more agents joined, the team drifted from what the user intended, with capable agents interfering, looping, and reassigning roles. The difficulty was not per-agent capability but how a growing team is *directed*, how its members *relate* to one another, and how the user can still *see* what each is doing; tellingly, the more agents there were, the more users leaned on *text* rather than watching the world to stay aware. These three pressures define the design space (Figure 1).

Concretely, when a person directs a team toward a shared goal, the user issues an instruction; some agent must *interpret* it and the team must *propagate* it (*user–model interaction*); the agents must then *coordinate* as they execute it (*model communication*); and throughout, both sides must *perceive* enough to stay aligned through a chosen *observation modality* (*scene* vs. *text*). MineMate treats all three as variables rather than fixed choices.

*Axis 1: user–model interaction (giving and propagating commands).* The user reaches the team in the *text* modality, issuing a natural-language instruction. Two questions follow. First, how does the receiving agent *process* that instruction, decomposing a high-level goal into a plan and low-level actions? Second, how is the instruction *propagated* to the rest of the team? The figure (bottom right) shows three regimes of increasing reach: **direct interaction**, where the user addresses each agent individually; **centralized interaction**, where the user speaks to one coordinator agent that relays and allocates work to a fixed set of others; and **hierarchical centralized interaction**, where that coordinator in turn delegates through further agents, extending the chain of command down a multi-level hierarchy. These trade the user's control against the user's workload: direct interaction is most precise but scales worst with team size, whereas deeper delegation offloads work onto the agents at the cost of the user knowing exactly who is doing what.

*Axis 2: model communication (coordinating during execution).* Once a command has propagated, the agents must coordinate among themselves while doing the task. I vary how much *structure* is imposed on this inter-agent communication, along one ordered spectrum read left to right off the top of Figure 1: **coupled** (a group shares a joint plan and context and communicates freely, acting almost as a single unit, with maximal intra-group teamwork but maximal per-prompt context); **decoupled** (groups are defined but exchange only summary results through a representative, lowering context noise at the cost of weaker real-time coordination); **isolated** (agents are partitioned into groups that never communicate across boundaries, so only the user bridges them); and **non-structured** (no fixed grouping; any agent may address any other in a single many-to-many web). This spectrum interpolates between the decentralized and centralized regimes of Schömbs et al. [16] and

operationalizes the communication and orchestration dimensions of Zou et al. [25].

*Axis 3: observation modality.* Finally, both the user and the agents must *perceive* this unfolding process, and the modality through which they do so is itself a design choice (Figure 1, bottom left). The *scene* modality is the in-game, first-person view of what is physically happening; it is grounded but *partial*, since an observer standing elsewhere, or seeing only one viewpoint, cannot read a situation that unfolds across the map. The *text* modality is the channel of what agents and users *report* to one another; it is effectively *full*, since every reported state can be gathered in one place, but shallow, since what an agent *says* need not match what it *does*. Which modality should serve the user, and which the agents, is an open question that Section 4 settles for MineMate and that the main experiment (Section 5) probes.

Crossing these three axes, command propagation, inter-agent communication structure, and observation modality, defines the design space. The remainder of the paper instantiates concrete points in it: Section 4 describes the MineMate system that realizes them, and Section 5 reports how users and agents behave as the space is explored.

## 4 THE MINEMATE SYSTEM

MineMate builds on the MINDcraft/MineCollab framework [20], which supplies the Minecraft binding (`mineflayer`), bot management, and a low-level `!command` interface. Each agent is assembled from established agent-architecture ideas rather than a new cognitive model; I describe the stack only so it can be read against the design space of §3.

### 4.1 Single-agent loop

A single agent runs an *observe* → *update state* → *predict* → *act* loop (Figure 2). I reuse, rather than reinvent, the per-agent components reviewed in §2.1: a timestamped memory stream with periodic reflection, instruction-to-plan-to-command decomposition, and a growing library of reusable named skills. The single agent is a solved-enough substrate here; MineMate's contribution is the teaming layer above it.

### 4.2 From one agent to a team

Going from one agent to many adds a coordination layer. Inter-agent messages are typed *speech acts* (`request`, `inform`, `propose`, `agree`, `commit`) so commitments can be tracked rather than merely exchanged, and shared world knowledge is held in a text knowledge store after GITM [24], with a curator that reconciles conflicting reports into one canonical fact. Agents are partitioned into *task groups* (e.g. Miner, Farming), and a human can "own" a group so work splits across people and agents. By default, communication is many-to-many and peer-to-peer: any agent may address any other and form plans dynamically. As §5 shows, this freedom is what makes the team feel autonomous but is also the source of interference and talk-loops, which is why communication *structure* is a first-class axis (§3).

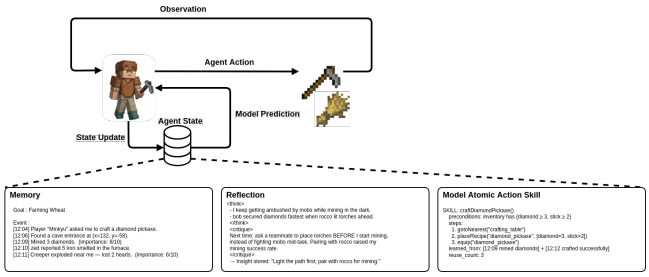


**Figure 2: Single-agent update loop. The agent perceives the world, updates a timestamped *Agent State*, predicts the next action, and acts. Three stores back the loop: a *Memory* stream of timestamped observations (after Generative Agents [13]), a *Reflection* buffer of think/critique insights (after Reflexion [17]), and a library of *Atomic Action Skills* saved from successful sequences (after Voyager [18]).**

### 4.3 Interaction channel: from in-game chat to Slack

In early pilot runs, users commanded agents through the in-game Minecraft chat, which scrolls quickly over the play surface and cannot be searched, so users had to hold the whole conversation in their head, a working-memory cost several found tiring. I therefore moved the dialogue to a Slack bridge: a persistent, scrollable history with named channels (e.g. `#general-talk`) that let users address and read agents per group rather than as one mixed stream. Slack also smooths *human-to-human* interaction, which pilot sessions flagged as mattering once more than one person is on the team: co-players coordinate with one another and with agent groups in the same place, outside the game view. The trade-off is that it biases input toward typed text, constraining the "just say it" voice style that felt natural in-game, which is why interaction and observation are treated as deliberate axes rather than fixed choices.

### 4.4 Observability asymmetry

The architecture makes observation asymmetric. In the game world it is *partial*: some agents see a region and others do not, as a human player would. Through text it is effectively *full*, since every reported state is visible in one place, but with a caveat that recurs throughout both studies, *a model does not reliably know its own state*: what an agent *says* is not the same as what it *does*. Observability is therefore the observation-modality axis of the design space (§3), not a fixed property of the system.

## 5 MAIN EXPERIMENT

I ran a single main experiment that instantiates the design space (§3). The agents were driven by two vision+language backbones used in parallel, Gemma and Kimi 2.5, on the MINDcraft setup. The goal is to localize, with real users, which configuration of the design space helps and how teaming scales with the agent count.

*Participants and procedure.* Four participants, recruited from UNIST friends and spanning a range of Minecraft experience, *shared a single world*. Each ~5-hour session opened with the design space explained, then ~1.5 hours of free play in which participants chose

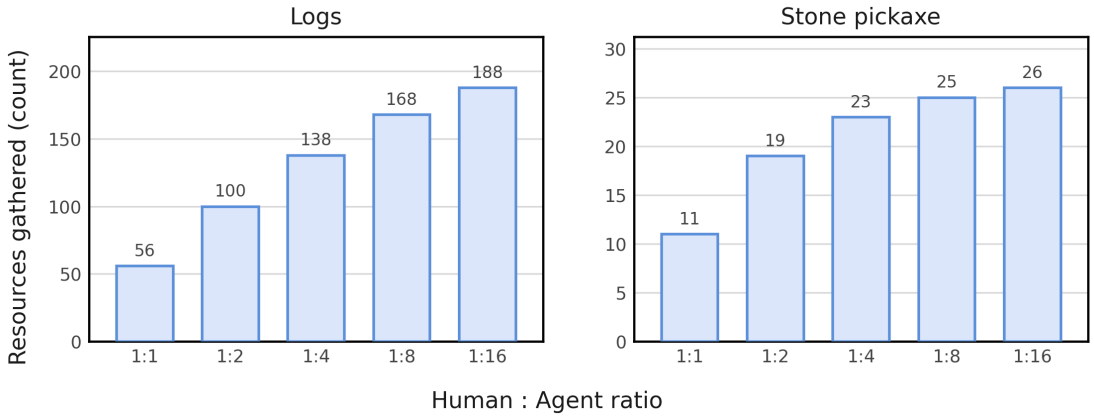


**Figure 3: Resource yield versus team size for the two measured subtasks, gathering *logs* and crafting a *stone pickaxe*, swept over Human:Agent ratios from 1:1 to 1:16 (low to high), with a freshly generated world per run. Both show *diminishing marginal returns*, but they saturate very differently. Log gathering is abundant and embarrassingly parallel, so its yield keeps climbing across the whole sweep. The stone pickaxe is a multi-step *collaborative* recipe (logs into sticks, stone into cobblestone, then combined), and inter-agent coordination is exactly where these agents are weakest, so it saturates almost immediately: extra agents stop helping past 1:4. Values are *per team* (one human and the agents they direct), averaged over the four teams; the yield axis is independent per panel.**

**Table 1: Main experiment at a glance, and the configuration each participant settled on during free play. Four UNIST friends; agents driven by Gemma and Kimi 2.5 in parallel on MINDCRAFT; ~5 h per session (~1.5 h free configuration play + five timed runs + setup/interview). Each 25 min run swept one Human:Agent ratio (1:1 to 1:16, low to high) on a fresh world: 15 min gathering logs, then 10 min crafting a stone pickaxe.**

| P | Interaction | Model comm. | Observation |
|---|---|---|---|
| P1 | Direct | Isolated | Text (Slack) |
| P2 | Centralized† | Decoupled | Text (Slack) |
| P3 | Centralized† | Decoupled | Text (Slack) |
| P4 | Centralized† | Decoupled | Text (Slack) |

† split into several small, separate groups, not one large coordinated team.

and switched design-space elements at will; the configuration each settled on (Table 1) is therefore a *revealed preference*, not an assignment. For the measured part, all teams were then *unified* to the most-preferred configuration, *Decoupled* groups with *Centralized* interaction, and ran five timed runs sweeping each team's Human:Agent ratio low to high, 1:1, 1:2, 1:4, 1:8, 1:16, on a freshly generated world per run; each 25-minute run gathered *logs* for 15 min then crafted a *stone pickaxe* for 10 min. Each participant commanded only their *own* team's coordinator and never another participant's agents, but agents could message and request work *across* team boundaries. Because the ratio was swept low to high, a within-session learning effect is confounded with team size, so I read the yield curves for *shape* (where they saturate) rather than absolute level.

*Scaling: gathering parallelizes, joint crafting does not.* Both tasks improve with diminishing returns (Figure 3), but log gathering keeps climbing across the whole sweep while the stone pickaxe flattens past 1:4. The takeaway is that team capability grows with the agent count only as far as a task decomposes into independent, parallel work; once a task needs agents to collaborate, extra agents stop helping, because inter-agent coordination is where these agents are weakest.

*Chosen configurations.* Table 1 lists what each participant settled on, and two patterns held across the group. First, as the agent count grew, managing every agent directly became uncomfortable: three of the four moved to *centralized* interaction, but centralized agents into several small, separate groups rather than one large team, which was harder to steer and slower to respond. Second, all four observed the team through *text* on the Slack bridge (§4.3) rather than the in-world view, since a per-group channel let them track many agents at once instead of chasing each through the world, the observation-modality preference the design space anticipated (§3). Across all four, participants rarely did the in-world work themselves; they spent their time *directing* agents and clearing *bottlenecks*, unblocking stuck agents and re-allocating tasks, so the human's role looked more like a coordinator than a player.

*Why decoupled, not free, communication.* On model communication the same three chose the *decoupled* regime, for a concrete reason: with free communication agents talked in circles and stalled, lost track of the goal, and, when one model held several tasks at once, did none properly. They still wanted *some* agency, so they let models share the resources they genuinely needed but never assigned one model two roles at once. The fourth participant went further to the *isolated* regime, reporting that in their experience

agent-to-agent collaboration tended to fail outright, so they cut cross-group communication and bridged the groups themselves.

## 6 DISCUSSION

*The bottleneck is coordination, not capability.* The clearest signal is that team yield grew with the agent count only as far as a task was *decomposable* into independent work: pure parallel gathering kept improving across the whole sweep, while the collaborative stone-pickaxe recipe stopped improving past 1:4 (Figure 3). The agents are competent individually; what fails is the *seam* between them. The same story appeared in how users configured the team, as they avoided free inter-agent communication precisely because agents talked in circles, lost the goal, and broke down when given two roles at once. This relocates the frontier for embodied LLM agents into HCI: the hard question is no longer "can the agent mine the block" but "how does a person keep a growing team aligned." My headline framing follows: embodied LLM agents are *strong as singletons but weak at collaboration.*

*Users want bounded agency, not full autonomy or full control.* Given a free choice, no participant centralized everything through one coordinator and none let the agents coordinate freely. They converged instead on a middle band, several small, *decoupled* groups that share only the resources they need and never carry two roles, with one user cutting inter-agent links entirely. The revealed preference is enough agency that the user is not micromanaging, but little enough that the fragile inter-agent layer cannot derail the task. Interfaces for agent teams should make that band easy to express, rather than offering only the extremes of full autonomy or hands-on control.

*Legibility runs through text, but text hides behavior.* All participants preferred text observation through Slack over the in-world view, because a per-group text channel let them track several agents at once instead of chasing each through the world. Yet text exposes what an agent *says*, not what it *does* (the observability asymmetry of §4.4); making agent *behavior* legible, not just its speech, is the open interface problem these teams raise.

*Emergent coordination cuts both ways.* The same emergence that makes these teams attractive is what makes them hard to govern. Prior many-agent work celebrates spontaneous structure, the self-organized roles of Project Sid [1] and the emergent gaming coordination of MindAgent [5], but in my sessions that same self-direction is what users fought: agents formed plans, reassigned work, and negotiated among themselves in ways that, left unmediated, drifted from the user's intent. The decoupled, single-role configuration users converged on is best read as a way to keep the useful part of emergence, agents still acting with initiative inside a narrow remit, while suppressing the part that derails the task, the open-ended cross-talk that loops or forgets the goal. Designing for agent teams is therefore less about maximizing autonomy than about choosing where, and how much, emergence is allowed.

*Design implications for agent-team interfaces.* Three implications follow for tools that put one person in charge of many agents. First, expose *bounded-agency* controls directly: rather than offering only full autonomy or per-agent micromanagement, let the user form small groups, grant each the resources it may share, and cap each agent at a single role, the very configuration participants assembled by hand here. Second, surface *behavior*, not just speech: because an agent's text report can diverge from what it does, the interface should let the user see and audit actions, not only read chat. Third, make task structure legible: since extra agents help on decomposable work but not on tightly coupled steps, an interface that flags which subtasks parallelize would show a user where adding agents pays off and where it will not, turning the diminishing-returns curve of Figure 3 into an actionable signal rather than a surprise discovered at scale.

*Limitations and future work.* This is a small, exploratory study, and it falls short of a controlled HCI experiment in ways worth stating plainly. Four participants shared one session, with uneven Minecraft skill and play strategies that I did not control, and the low-to-high ratio sweep confounds a learning effect with team size. The measured runs did fix one unified configuration, so the scaling curves are comparable, but the configuration *preferences* (Table 1) come from free play and are revealed, not controlled, choices. The results are therefore directional, surfacing which variables matter and how teaming scales in shape, not controlled effect estimates. Future work should scale the number of users, fix or factorize the configuration, counterbalance the ratio order, cross team composition with spatial role assignment (bounded, repetitive regions versus open, long-horizon ones), and restore a richer voice-in / text-out channel.

## 7 CONCLUSION

MineMate uses Minecraft as an affordable, controllable surrogate for the embodied human–agent teaming that real robots cannot yet support at scale. In a main experiment in which four people directed teams swept from 1:1 to 1:16 over two subtasks, the binding constraint was never the agents' raw capability but the coordination *between* capable agents and the people directing them: team yield rose with the agent count only for decomposable, parallel work and saturated as soon as a task required inter-agent collaboration, while users converged on small, decoupled groups observed through text. I organize these findings into a three-axis design space of user–model interaction, model communication, and observation, and argue that, as agentic systems move into the physical world, the central HCI problem is helping a person stay in control of a team of emergent, only-partially-legible teammates. The testbed and the design space are offered together as a reusable scaffold for that study; the project page collects the demonstration clips and the data behind the figures (https://mk040412.github.io/MineMate/).

# REFERENCES


[1] Altera.AL. 2024. Project Sid: Many-agent simulations toward AI civilization. *arXiv preprint arXiv:2411.00114* (2024).

[2] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI)*.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. 2024. $\pi_0$: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164* (2024).

[4] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

[5] Ran Gong, Qiuyuan Huang, Xiaojian Ma, Hoi Vo, Zane Durante, Yusuke Noda, Zilong Zheng, Song-Chun Zhu, Demetri Terzopoulos, Li Fei-Fei, and Jianfeng Gao. 2024. MindAgent: Emergent Gaming Interaction. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 3154–3183.

[6] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*. 8048–8057.

[7] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. 2019. MineRL: A Large-Scale Dataset of Minecraft Demonstrations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*.

[8] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In *International Conference on Learning Representations (ICLR)*.

[9] Eric Horvitz. 1999. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. 159–166.

[10] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. In *Proceedings of the 8th Conference on Robot Learning (CoRL) (Proceedings of Machine Learning Research, Vol. 270)*. PMLR, 2679–2713.

[11] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[12] Qian Long, Zhi Li, Ran Gong, Ying Nian Wu, Demetri Terzopoulos, and Xiaofeng Gao. 2024. TeamCraft: A Benchmark for Multi-Modal Multi-Agent Systems in Minecraft. *arXiv preprint arXiv:2412.05255* (2024).

[13] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[14] Chen Qian, Wei Liu, Nuo Chen, Yufan Dang, Jiahao Li, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*. 15174–15186.

[15] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2186–2188.

[16] Sarah Schömbs, Yan Zhang, Jorge Goncalves, and Wafa Johal. 2025. From Conversation to Orchestration: HCI Challenges and Opportunities in Interactive Multi-Agentic Systems. In *Proceedings of the 13th International Conference on Human-Agent Interaction (HAI)*. ACM, 158–168.

[17] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[18] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research (TMLR)* (2024).

[19] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2025. JARVIS-1: Open-World Multi-task Agents with Memory-Augmented Multimodal Language Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 47, 3 (2025), 1894–1907. https://doi.org/10.1109/TPAMI.2024.3511593

[20] Isadora White, Kolby Nottingham, Ayush Maniar, Max Robinson, Hansen Lillemark, Mehul Maheshwari, Lianhui Qin, and Prithviraj Ammanabrolu. 2025. Collaborating Action by Action: A Multi-agent LLM Framework for Embodied Reasoning. *arXiv preprint arXiv:2504.17950* (2025).

[21] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *Conference on Language Modeling (COLM)*.

[22] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2025. Large Language Model-Brained GUI Agents: A Survey. *Transactions on Machine Learning Research (TMLR)* (2025).

[23] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In *Handbook of Reinforcement Learning and Control*. Studies in Systems, Decision and Control, Vol. 325. Springer, 321–384.

[24] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. 2023. Ghost in the Minecraft: Generally Capable Agents for Open-World Environments via Large Language Models with Text-based Knowledge and Memory. *arXiv preprint arXiv:2305.17144* (2023).

[25] Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Yankai Chen, Chunyu Miao, Hoang Nguyen, Yue Zhou, Weizhi Zhang, Liancheng Fang, Langzhou He, Yangning Li, Dongyuan Li, Renhe Jiang, Xue Liu, and Philip S. Yu. 2026. LLM-Based Human-Agent Collaboration and Interaction Systems: A Survey. In *Findings of the Association for Computational Linguistics: ACL 2026*. Association for Computational Linguistics.