Communication to Completion: Modeling Collaborative Workflows with Intelligent Multi-Agent Communication

Yiming Lu[1,2*], **Xun Wang**[2], **Simin Ma**[2], **Shujian Liu**[2], **Sathish Reddy Indurthi**[2]
Song Wang[2], **Haoyun Deng**[2], **Fei Liu**[1], **Kaiqiang Song**[2]
[1]Emory University [2]Zoom Video Communications
{yiming.lu,fei.liu}@emory.edu
{xun.wang,kaiqiang.song}@zoom.us

Abstract

Teamwork in workspace for complex tasks requires diverse communication strategies, but current multi-agent LLM systems lack systematic frameworks for task oriented communication. We introduce **Communication to Completion (C2C)**, a scalable framework that addresses this gap through two key innovations: (1) the Alignment Factor (AF), a novel metric quantifying agent task alignment that directly impacts work efficiency, and (2) a Sequential Action Framework that integrates stepwise execution with intelligent communication decisions. C2C enables agents to make cost aware communication choices, dynamically improving task understanding through targeted interactions. We evaluated C2C on realistic coding workflows across three complexity tiers and team sizes from 5 to 17 agents, comparing against **no communication** and **fixed steps** baselines. The results show that C2C reduces the task completion time by about 40% with acceptable communication costs. The framework completes all tasks successfully in standard configurations and maintains effectiveness at scale. C2C establishes both a theoretical foundation for measuring communication effectiveness in multi-agent systems and a practical framework for complex collaborative tasks.

1 Introduction

Modern organizations run on teams: people plan, divide, and verify work under deadlines while juggling synchronous meetings, asynchronous chats/emails, and limited attention. Extending this paradigm to AI, coordinating teams of LLM agents to execute complex, long-horizon tasks (e.g., software development) remains a grand challenge (Park et al., 2023; Hong et al., 2024; Qian et al., 2023). Effective collaboration hinges on communication: too much creates coordination overhead; too little yields misalignment and rework.

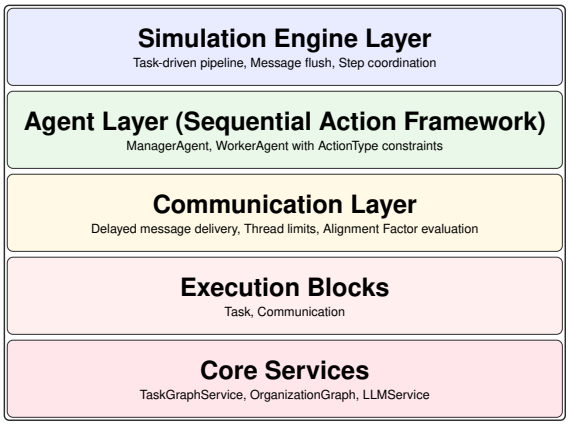


Figure 1: Communication to Completion architecture: a five-layer stack simulation engine

Recent multi-agent frameworks demonstrate that structured dialogues and role specialization can transform general purpose LLMs into cooperative problem solvers across software engineering, data analysis, and decision support (Wu et al., 2024; Hong et al., 2024; Qian et al., 2023; Li et al., 2023). Beyond mere tool use, conversation itself decomposes tasks, aligns partial knowledge, and arbitrates proposals, often outperforming single agent pipelines on complex problems (de Souza Santos and Ralph, 2022; Khan et al., 2024; Yao et al., 2023a; Besta et al., 2024). Yet many systems still schedule or trigger communication via fixed or purely reactive heuristics that ignore the evolving trade-off between coordination cost and task progress (Wu et al., 2024; Chen et al., 2023).

Meanwhile, empirical studies of software teams warn that coordination costs (especially meetings), can dominate delivery time in modern distributed work, with a post-2020 shift toward heavier synchronous load and changing patterns across chat, email, and meetings (Mok et al., 2023; Yang et al., 2022). These observations motivate treating communication as a first class resource to be sched-

[*]Work done during internship at Zoom

uled, routed, and evaluated rather than as a byproduct of agent reasoning, thus leading to a central open question: *how can agents learn to communicate strategically to maximize collaborative benefit while minimizing coordination costs?*

In this work, we address the above question by casting multi-agent collaboration as cost communication and optimizing it accordingly. Our contributions are:

- We introduce **Communication to Completion (C2C)**. A scalable framework models *when*, *whom*, and *how* to communicate as part of execution and instantiate it with a **Sequential Action Framework (SAF)** that synchronizes agent actions into discrete timesteps for deterministic, reproducible collaboration.
- We define **Alignment Factor (AF)**, a metric that quantifies each agent's task understanding and directly modulates work efficiency. Communications update AF, and AF in turn scales work rate, linking conversation to progress and yielding interpretable traces.
- We demonstrate through comprehensive experiments spanning complexities and team sizes, C2C lowers completion time and improves work efficiency compared to baselines.

2 Related Work

Multi-Agent Collaboration Frameworks LLM agents are increasingly organized as teams rather than monoliths. Social simulation work demonstrates emergent, human-like collective behavior (e.g., the "Smallville" agents of Park et al. (2023) and the recent large scale AgentSociety simulation (Piao et al., 2025)). For goal oriented collaboration—particularly software development—frameworks such as MetaGPT (Hong et al., 2024) and ChatDev (Qian et al., 2023) impose predefined roles and SOPs that mirror corporate workflows. AutoGen (Wu et al., 2024) offers a flexible substrate for composing conversable agents, while AgentVerse (Chen et al., 2023) and OpenAgents (Xie et al., 2023) provide extensible environments for building and deploying general purpose agents. Outside pure research prototypes, platforms like OpenHands operationalize multi/single agent development loops in realistic toolchains (Wang et al., 2025). In parallel, role-playing and deliberation structures further organize collaboration within conversations (Li et al., 2023; Yao et al., 2023a; Besta et al., 2024), and new evaluations emphasize interactive, multi-environment testing and repository-level search and planning (Liu et al., 2023a; Antoniades et al., 2024). Our work differs in elevating communication from a procedural step to an optimizable resource, explicitly linking when/what to communicate with collaborative progress via an alignment driven mechanism.

Communication in Multi-Agent Systems Communication has long been central in multi-agent learning. Classic MARL studies learned protocols end-to-end under partial observability (Foerster et al., 2016; Sukhbaatar et al., 2016). Recent advances focus on efficient or sparse communication to reduce overhead, e.g., decentralized scheduling and model-based message estimation (Han et al., 2023; Liu et al., 2023c; Kim et al., 2019; Singh et al., 2018). In LLM agent settings, natural-language messaging carries high-level, semantic content, and methods increasingly structure agent talk through reasoning–acting loops (ReAct) (Yao et al., 2023b), self improvement with textual feedback (Reflexion) (Shinn et al., 2023), and iterative self-refinement (Madaan et al., 2023). Multi-agent debate further uses conversational arbitration to improve correctness and planning (Du et al., 2023). Yet many systems still trigger communication on fixed or purely reactive heuristics. We instead propose an alignment driven policy that decides when, whom and how to communicate based on a measurable, task grounded alignment signal.

Quantifying Coordination and Alignment Coordination costs are a foundational concern in collaborative systems (Malone, 1988; Brooks Jr, 1995). However, evaluations of LLM multi agents often emphasize final outcomes (e.g., pass rates on benchmarks) over process efficiency. In software engineering, SWE-bench exposes the gap between realistic issue resolution and model capabilities (Jimenez et al., 2023), while SWE-agent (Yang et al., 2024) and OpenHands (Wang et al., 2025) reveal the importance of tool use and orchestration—but still provide limited real-time signals about team alignment. Complementary efforts argue for richer process traces and interactive environments to analyze coordination choices (Liu et al., 2023a; Antoniades et al., 2024), and work on LLM-as-judge warns that automated evaluators may introduce bias at the process level (Liu et al., 2023b; Chen et al., 2024; Tan et al., 2024). We contribute the Alignment Factor (AF) as a dynamic,

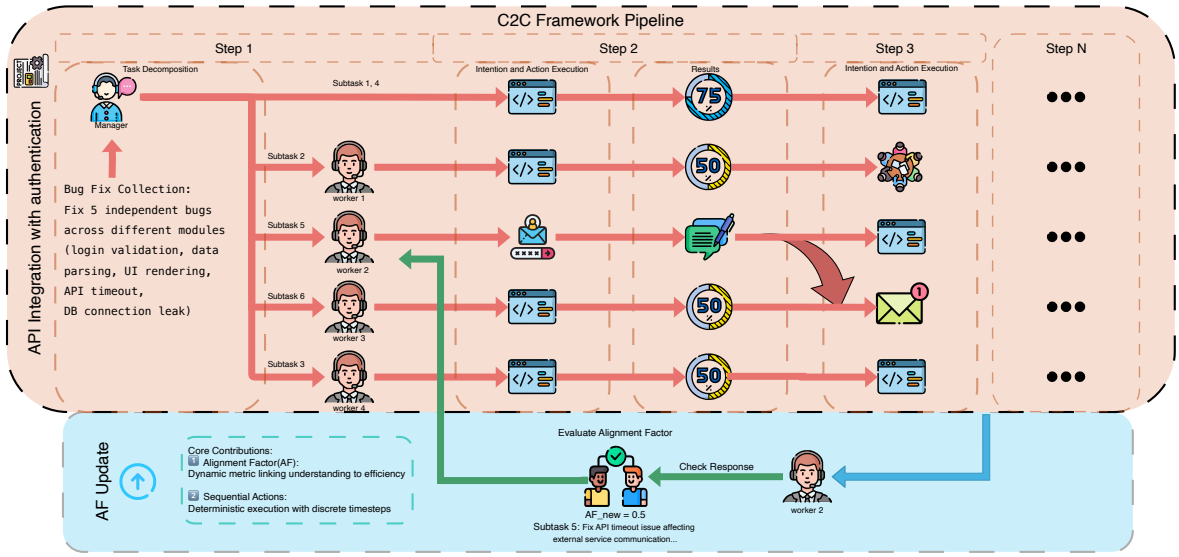


Figure 2: **C2C timeline on a software bug-fix task.** Step 1: the manager decomposes the task and assigns subtasks. Step 2: workers execute intentions and produce interim results. Communications update the *Alignment Factor (AF)*; higher AF increases work efficiency and reshapes subsequent actions (green feedback). Vertical dashed dividers denote discrete timesteps in the Sequential Action Framework; the cycle repeats until completion (Step N).

real-time proxy for shared task understanding that both modulates work efficiency and directly informs agents' communication decisions, thereby connecting conversational actions to tangible task outcomes.

3 C2C Framework

Communication to Completion is a time-stepped framework for deciding *when*, *whom*, and *how* to communicate in multi-agent collaboration. Figure 2 provides an overview of its mechanism using a software task. We organize the discussion as follows: §3.1 introduces the Sequential Action Framework that ensures deterministic execution, §3.2 describes the Alignment Factor mechanism for quantifying task understanding, §3.3 details how agents make communication decisions, §3.4 explains hierarchical task decomposition and tracking, and §3.5 presents the intention-based agent decision making that integrates these components.

3.1 Sequential Action Framework

Concurrent multi-agent execution often introduces temporal ambiguity. The *Sequential Action Framework* (SAF) constrains each agent to exactly one action per timestep, yielding deterministic state transitions and reproducible analysis.

Actions SAF defines four actions capturing common collaborative behavior that capture common collaborative behavior: *work* (task execution with time cost), *communicate* (compose/send a message), *reply* (respond to an incoming message), and *meeting* (synchronous group discussion). Each action is temporally bounded and commits a single transition.

Synchronized Timesteps All agents share a fixed temporal grid. At timestep t, agent i selects and executes a_i^t, then the system advances to $t{+}1$. Formally, all actions at t finish before any action at $t{+}1$ begins:

$$\forall i, j: \; a_i^t \text{ completes before } a_j^{t+1} \text{ starts.} \quad (1)$$

Forward-only Delivery. Messages sent at t are delivered at $t{+}1$, preventing instantaneous feedback loops and enforcing causal consistency. A communication buffer stages pending deliveries and flushes at timestep boundaries.

3.2 Alignment Factor

The *Alignment Factor* (AF) measures an agent's task-specific understanding and directly modulates effective productivity. Unlike static capability scores, AF evolves through communication. For agent i on task j, $AF_{i,j} \in [0.01, 1.00]$ denotes comprehension quality. Values near 0.01 indicate limited understanding; values near 1.00 indicate mastery. We initialize $AF_{i,j}{=}0.3$ to reflect the initial understanding.

When agent i receives reply about task j from agent k, AF is updated based on the communication's impact on task understanding. An LLM evaluates the message content, agent i's current task context, and the relevance of the received information to determine the new alignment:

$$AF_{i,j}^{\text{new}} = \min\big(1.0,\ AF_{i,j}^{\text{old}} + \Delta_{\text{eval}}\big), \qquad (2)$$

where $\Delta_{\text{eval}} \in [0, 0.5]$ is computed by the LLM based on: (i) how well the message addresses the agent's knowledge gaps, (ii) the relevance of the information to the specific task requirements, and (iii) the clarity and actionability of the guidance provided. Different message types naturally yield different impacts, help requests addressing critical blockers typically produce larger alignment gains.

If agent i spends h hours on task j, the effective progress is

$$\text{EffectiveProgress} \ = \ h \cdot AF_{i,j}, \qquad (3)$$

A low AF impedes an agent's progress, creating a natural incentive for it to seek clarification and coordination before investing substantial effort.

3.3 Agent Communication Decisions

Agents autonomously decide when, with whom, and how to communicate based on their current task state and situational awareness, without relying on fixed thresholds or rigid scheduling. For clarity, we decompose this decision process into four key components: initiating the communication, selecting the recipients, choosing the appropriate channel, and composing the message content.

Communication Initiation Agents may initiate communication in response to various task situations: encountering technical difficulties (HELP_REQUEST), facing unclear requirements (NEED_CLARIFICATION), identifying coordination needs with team members (MEETING_INVITE), or reaching milestones that warrant status updates (PROGRESS_UPDATE). The decision to communicate emerges from the agent's assessment of its current alignment and task complexity.

Recipient Selection When initiating communication, agents consider multiple factors in selecting recipients: skill relevance to the issue, existing task dependencies, and historical interaction. The selection process reflects realistic collaboration patterns observed in human teams.

Channel Selection Agents choose among three communication channels. *Chat* suits quick questions and simple clarifications with rapid response times. *Email* handles detailed explanations with longer response windows. *Meetings* address complex coordination needs requiring synchronous discussion among multiple participants.

Message Composition Communication content is generated based on the agent's current context, including task description, encountered difficulties, current alignment factor, and relevant technical details. Messages are composed to be informative and actionable, providing recipients with sufficient context to offer effective assistance.

3.4 Hierarchical Task Management

Complex tasks are decomposed by manager agents and tracked as a directed acyclic graph (DAG) of subtasks and dependencies.

The manager performs task decomposition in the beginning of the simulation, by analyzing the task requirements and proposing several subtasks according to team size and skills. Subtasks are connected through dependency edges. The manager tracks these dependencies and coordinates task assignments to ensure workers receive suitable subtasks. And the manager updates the task graph as workers complete subtasks, monitoring overall progress. Parent task progress is computed as a weighted average of subtask progress:

$$\text{ParentProgress} = \frac{\sum_{i \in \text{subtasks}} w_i \, P_i}{\sum_{i \in \text{subtasks}} w_i}, \qquad (4)$$

where w_i represents the effort estimate for subtask i. The parent task is marked complete only when all required subtasks reach done status, ensuring accurate project tracking.

3.5 Intention-Based Agent Decision Making

Each agent makes decisions through an intention-based mechanism that evaluates the current situation and generates contextually appropriate actions.

Context Formation At each timestep, the agent constructs a comprehensive context that includes: (i) currently assigned tasks and their completion status, (ii) alignment factors for each assignment indicating task understanding, (iii) recent communications including pending requests and received guidance, and (iv) team state including teammate skills and availability

Intention Given this context, the agent uses an LLM to carefully reason about the current situation and generate an intention, a high-level decision about what action to take next. The generated intention is then translated into a concrete action within the SAF framework:

- **Work**: Continue task execution with effectiveness modulated by current AF.
- **Help**: Compose the message identifying difficulties and required expertise (HELP_REQUEST).
- **Clarification**: Request additional task details or requirement explanations (NEED_CLARIFICATION).
- **Coordination**: Propose synchronous discussion for complex collaboration needs (MEETING_INVITE).
- **Reporting**: Share subtasks progress with manager (PROGRESS_REPORT).

When generating communication intentions, the agent also determines appropriate recipients and selects suitable channels (chat, email, meeting) based on urgency and complexity.

Adaptive Behavior This intention-based method enables agents to exhibit adaptive behavior without hard coded rules. The communication patterns emerge naturally from the agents' contextual reasoning rather than threshold triggers.

By integrating these components, C2C enables agents to collaborate strategically and adaptively based on actual task needs.

4 Experiments

4.1 Experimental Setup

To evaluate the performance of C2C on realistic tasks, we implement experiments on software engineering workflows across three complexity tiers: **Simple** (8 hours, basic SWE operations), **Medium** (24 hours, API integration with authentication), and **Complex** (40 hours, full stack user authentication system); see Appendix A for details. Each task requires diverse skills including front-end development, back-end systems, database management, security implementation, and testing. Tasks are decomposed into several subtasks according to task complexity and team size with explicit skill requirements and dependencies, mirroring real-world software development scenarios.

Team Configurations We systematically vary team sizes from 5 to 17 agents: 1M+4W (5 agents), 1M+8W (9 agents), and 1M+16W (17 agents), where M denotes manager and W denotes worker. Additionally, we evaluate multi-task scenarios with 1M+8W handling 2 concurrent tasks to assess workload distribution and task interference effects. Each agent is assigned 2–4 complementary skills from a skill pool, including skills like *frontend*, *backend*, and *database*.

Models and Implementation All agents are powered by GPT-4o with temperature 0.7 for human like decision making. The Sequential Action Framework operates with 0.25 hour time steps over a maximum of 160 simulation steps (40 hours). Communication costs are calculated using realistic time models: email drafting (9 minutes base + content length), chat messages (3 minutes base), meetings (30 minutes minimum + preparation time + number of participants). We implement thread-depth limits (maximum 3 reply rounds) to prevent infinite communication loops.

Baselines We compare C2C against two systematic baselines: (1) **No Communication**: agents work independently with task assignments but no communication, representing traditional parallel processing approaches; (2) **Fixed Steps**: agents communicate every 16 steps, simulating conventional project management practices. Both baselines use identical task decomposition and skill matching but lack C2C's intelligent communication strategies.

Metrics Following our evaluation framework, we report: **task completion rate** (percentage of tasks completed within time budget), **average completion time** (hours to finish successful tasks), **communication cost** (total time spent on communication activities), **alignment score** (average agent task alignment factor), and **efficiency** (ratio of productive work to total time investment). Each configuration is evaluated across all three task complexity tiers.

4.2 Main Results

As shown in Table 1, which details the results for a 1M+4W team, C2C consistently demonstrates superior performance in key efficiency metrics, although all methods achieved a 100% task completion rate. The primary advantage of C2C is most evident in task completion time. On Complex

Metric	Complexity	No Communication	Fixed Steps	C2C
Task Completion Rate (%)	Simple	100	100	100
	Medium	100	100	100
	Complex	100	100	100
Avg Completion Time (hours)	Simple	7	**5**	5.5
	Medium	20	14.75	**13**
	Complex	33.5	36.25	**24.75**
Communication Cost	Simple	–	2.03	1.94
	Medium	–	2.75	3.26
	Complex	–	8.12	7.02
Alignment Score (AF)	Simple	0.3	**0.55**	0.51
	Medium	0.3	0.59	**0.68**
	Complex	0.3	0.53	**0.55**
Efficiency	Simple	1.14	**1.6**	1.45
	Medium	1.20	1.63	**1.85**
	Complex	1.19	1.10	**1.62**

Table 1: Comparison of three policies for a 1M+4W team across task complexities (Simple=8h, Medium=24h, Complex=40h). We report **Task Completion Rate** (%), **Avg Completion Time** (hours), **Communication Cost** (total communication time), **Alignment Score** (mean AF over the run; higher indicates better shared understanding), and **Efficiency** = completed work / total time. C2C achieves the best time and efficiency on Medium and Complex with comparable communication cost. Bold marks the best value within each row; "-" denotes not applicable.

tasks, C2C finishes in **24.75 hours**, significantly faster than both No Communication (33.5 hours) and Fixed Steps (36.25 hours). This trend holds for Medium tasks, where C2C is also the fastest.

This time saving translates to higher efficiency. For Complex tasks, C2C's efficiency score is **1.62**, substantially outperforming Fixed Steps (1.10) and No Communication (1.19). Similarly, C2C achieves the highest alignment score on Medium (0.68) and Complex (0.55) tasks, confirming that its communication strategy effectively enhances agent understanding. While communication costs are comparable across strategies, C2C utilizes its communication budget more effectively to achieve better outcomes.

Figure 3 illustrates the dynamics of the alignment factor mechanism. Alignment scores start low as agents have minimal task understanding, then improve through strategic communications. Help requests and meetings yield the strongest alignment improvements, while progress updates are modest. The communication heatmap reveals the manager centric communication pattern.

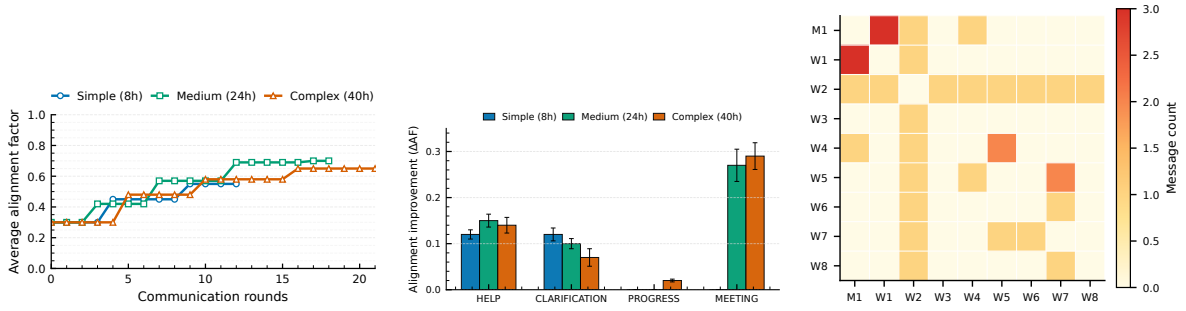
Configuration	Completion Time	Comm/Agent	Comm Cost	Speedup
1M + 4W	13h	3.1	2.75	1.54
1M + 8W	11.25h	2.1	3.78	1.78
1M + 16W	10.25h	2.6	5.12	**1.95**
1M+8W (2 tasks)	21h	2.3	4.64	1.35

Table 2: Scalability and multi-task analysis showing communication cost and speedup across team configurations. Multi-task performance demonstrates effective workload balancing and resource sharing capabilities. Speedup is calculated as the ratio of the completion time under the "No Communication" baseline to the completion time of each respective configuration.

4.3 Scalability and Multi-Task Analysis

Table 2 demonstrates C2C's effectiveness across team sizes and workload scenarios. The framework exhibits sub-linear scaling in communication cost: while team size increases to 3.4 times (5 to 17 agents), communication cost only increases 86%. This favorable scaling behavior stems from C2C's intelligent communication routing and the alignment factor mechanism, which prioritizes high value interactions over communications.

Speedup analysis reveals that performance steadily improves as the number of agents increases, with the largest configuration tested (1M+16W) achieving the highest speedup of



(a) Alignment evolution over rounds (b) Impact of communication type (c) Communication heatmap

Figure 3: Analysis of the Alignment Factor (AF) dynamics. (a) Average alignment improves over successive communication rounds across all task complexities. (b) High intent communications like MEETINGS and HELP requests yield the largest gains in alignment. (c) The communication heatmap reveals a manager centric coordination pattern, with the manager (M1) acting as the central hub.

$\mathbf{1.95}\times$. This highlights the effective task parallelization and specialized skill utilization in larger teams. However, the incremental gain in speedup lessens as the team grows, suggesting that simply adding more workers yields progressively smaller advantages.

The multi-task evaluation with 1M+8W reveals C2C's ability to handle concurrent workloads effectively. When processing 2 tasks simultaneously, completion time increases from 11.25h to 21h (an 87% increase), significantly better than naive linear scaling. Analysis of communication patterns shows that C2C naturally evolves hub-and-spoke topologies with managers as primary coordinators, avoiding the quadratic communication complexity that plagues peer-to-peer approaches. In multi-task scenarios, agents exhibit sophisticated context switching behavior, maintaining separate alignment factors per task and prioritizing communications based on overall workflow optimization.

4.4 Message Type and Decomposition

Message Type Analysis Table 3 details the impact of different message types on collaboration. **MEETING_INVITE** messages provide the highest alignment gains (+0.27), followed by **HELP_REQUEST** (+0.15), directly correlating with task success by resolving critical blockers. In contrast, **PROGRESS_UPDATE** messages maintain synchronization with a neutral (0) impact on alignment in this context. This analysis validates C2C's strategy of prioritizing communications that address specific knowledge gaps.

Task Decomposition Quality Effective collaboration begins with high quality task decomposition.

Message Type	Frequency	Avg Response Step	Impact on Alignment
HELP_REQUEST	**9**	3	+0.15
CLARIFICATION	0.3	2	+0.10
PROGRESS_UPDATE	0.3	5	0
MEETING_INVITE	1	7	+0.27
RESPONSE	8	–	–

Table 3: Analysis of message types in the 1M+8W configuration on Medium tasks, showing the frequency and impact of each communication type.

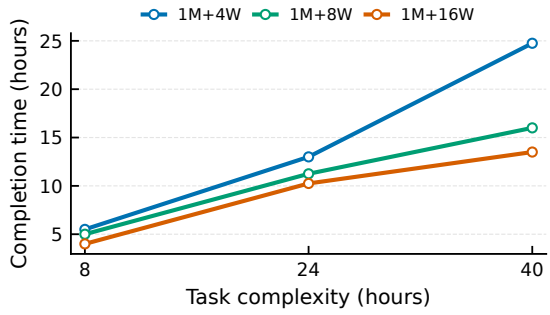
We compare C2C's adaptive decomposition strategy against manual, naive LLM, and hierarchical methods in Table 4. C2C's approach, which considers agent skills and workload during decomposition, achieves superior subtask clarity and higher AF. This leads to **14% increase** in worker utilization rate compared to naive LLM decomposition, demonstrating that context aware planning is critical for multi-agent efficiency.

Decomposition Method	Subtask Clarity	Alignment Factor
Manual	1.00	0.70
LLM-naive	0.72	0.58
Hierarchical	0.89	0.64
C2C Adaptive	**0.95**	**0.68**

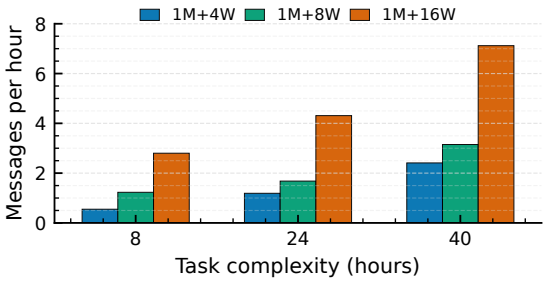
Table 4: Comparison of task decomposition quality. Subtask Clarity is rated by an LLM judge (1.0 is best). C2C's adaptive method approaches manual quality.

4.5 Communication Pattern Analysis

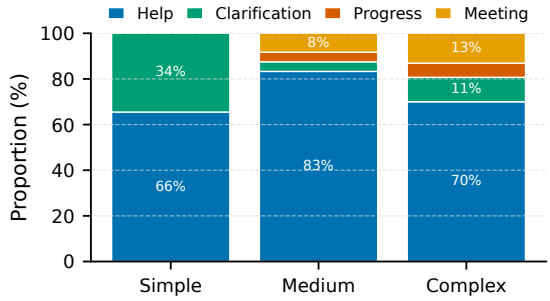
Figure 4 shows how communication varies with task complexity and team size (1M+4W, 1M+8W, 1M+16W). **(a)** Completion time increases with task complexity; adding workers shortens time but with diminishing returns. **(b)** Message intensity (messages/hour) rises with complexity and team size.



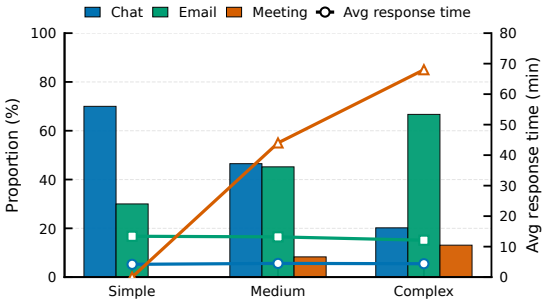
(a) Time vs. Complexity.



(b) Communication Freq. vs. Complexity.



(c) Message Type Distribution.



(d) Communication Method Analysis.

Figure 4: Effect of task complexity under C2C (8/24/40 hours) and team configuration (1M+4W, 1M+8W, 1M+16W). (a) Completion time increases with task complexity; larger worker pools shorten time with diminishing returns. (b) Messages per hour rise with both complexity and team size. (c) Help messages dominate across settings; clarification and meeting shares grow with complexity, and progress updates appear from medium upward. (d) Communication channels shift from chat (simple) toward email (complex), with a modest increase in meetings.

Message Types In **(c)**, simple tasks are dominated by *help* and *clarification* (about 66% and 34%, respectively). At medium complexity, *help* remains the majority (≈83%), while *meetings* appear (≈8%). For complex tasks, the mix diversifies: *help* ≈70%, and *meetings* ≈13% become dominant. C2C allocates most messages to high value help requests and escalates to meetings only when the expected coordination benefit exceeds the cost.

Channels and Latency Panel **(d)** indicates a shift from chat toward email as complexity grows, with a modest rise in meetings. The per channel response times for *chat* and *email* stay roughly flat across complexity levels (reduce slightly), whereas *meeting* latency increases for complex tasks.

These patterns align with the logic of C2C: as tasks become more complex, agents continue to seek help most of the time but increasingly use meetings, while clarification needs drop. The engine selects higher yield (though costlier) channels when needed, while larger teams reduce completion time without eliminating the upward pressure on communication volume with complexity.

5 Conclusion

In this paper, we present Communication to Completion, a multi-agent framework that treats communication as an optimizable resource and quantifies task understanding through the Alignment Factor within a Sequential Action Framework. By treating alignment as a quantifiable and dynamic variable, C2C enables agents to autonomously determine when and how to communicate.

Our experiments demonstrate that this alignment-driven approach adapts naturally to task complexity, with agents communicating more frequently and through richer channels when facing difficult problems while minimizing overhead on simpler tasks. Across diverse task complexities and team sizes, C2C consistently achieves high task completion while reducing completion time and improving efficiency relative to baselines. The framework produces interpretable patterns, including alignment factor trajectories, communication type distributions, and collaboration network structures, which bridge agent behavior and coordination theory to provide design insights for multi-agent systems.

Limitations

While the C2C framework demonstrates significant performance gains, we acknowledge several limitations that offer avenues for future research. The findings are derived from a controlled simulation, and the framework's performance in unpredictable, real-world environments is yet to be validated. The experiments were confined to the software engineering domain, so the C2C's effectiveness may not generalize to other collaborative fields. Furthermore, the Alignment Factor (AF) relies on an LLM's evaluation, which introduces potential subjectivity and bias.

References

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. 2024. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Frederick P Brooks Jr. 1995. *The mythical man-month: essays on software engineering*. Pearson Education.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, and 1 others. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2(4):6.

Ronnie E de Souza Santos and Paul Ralph. 2022. A grounded theory of coordination in remote-first and hybrid software teams. In *Proceedings of the 44th International Conference on Software Engineering*, pages 25–35.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29.

Shuai Han, Mehdi Dastani, and Shihan Wang. 2023. Model-based sparse communication in multi-agent reinforcement learning. In *Proceedings of the 2023 international conference on autonomous agents and multiagent systems*, pages 439–447. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. International Conference on Learning Representations, ICLR.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R Bowman, Tim Rocktäschel, and Ethan Perez. 2024. Debating with more persuasive llms leads to more truthful answers. *arXiv preprint arXiv:2402.06782*.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. 2019. Learning to schedule communication in multi-agent reinforcement learning. *arXiv preprint arXiv:1902.01554*.

Joel Z Leibo, Edgar A Dueñez-Guzman, Alexander Vezhnevets, John P Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charlie Beattie, Igor Mordatch, and Thore Graepel. 2021. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *International conference on machine learning*, pages 6187–6199. PMLR.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023a. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023b. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Zeyang Liu, Lipeng Wan, Xue Sui, Zhuoran Chen, Kewu Sun, and Xuguang Lan. 2023c. Deep hierarchical communication graph in multi-agent reinforcement learning. In *IJCAI*, pages 208–216.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon,

Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Thomas W Malone. 1988. Modeling coordination in organizations and markets. In *Readings in Distributed Artificial Intelligence*, pages 151–158. Elsevier.

Lillio Mok, Lu Sun, Shilad Sen, and Bahareh Sarrafzadeh. 2023. Challenging but connective: large-scale characteristics of synchronous collaboration across time zones. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Joon Sung Park, Joseph O'Brien, Carrie Jun 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*, pages 1–22.

Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, and 1 others. 2025. Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society. *arXiv preprint arXiv:2502.08691*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Stuart Russell and Peter Norvig. 2010. *Artificial Intelligence: A Modern Approach*, 3 edition. Prentice Hall.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2018. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*.

Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.

Sainbayar Sukhbaatar, Rob Fergus, and 1 others. 2016. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. 2024. Judgebench: A benchmark for evaluating llm-based judges. *arXiv preprint arXiv:2410.12784*.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, and 5 others. 2025. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, and 1 others. 2023. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Longqi Yang, David Holtz, Sonia Jaffe, Siddharth Suri, Shilpi Sinha, Jeffrey Weston, Connor Joyce, Neha Shah, Kevin Sherman, Brent Hecht, and 1 others. 2022. The effects of remote work on collaboration among information workers. *Nature human behaviour*, 6(1):43–54.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Ming Yin, Dinghan Shen, Silei Xu, Jianbing Han, Sixun Dong, Mian Zhang, Yebowen Hu, Shujian Liu, Simin Ma, Song Wang, and 1 others. 2025. Livemcp-101: Stress testing and diagnosing mcp-enabled agents on challenging queries. *arXiv preprint arXiv:2508.15760*.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. 2023. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2(5).

Kaiyan Zhang, Runze Liu, Xuekai Zhu, Kai Tian, Sihang Zeng, Guoli Jia, Yuchen Fan, Xingtai Lv, Yuxin Zuo, Che Jiang, Ziyang Liu, Jianyu Wang, Yuru Wang, Ruotong Zhao, Ermo Hua, Yibo Wang, Shijie

Wang, Junqi Gao, Xinwei Long, and 7 others. 2025. Marti: A framework for multi-agent llm systems reinforced training and inference.

A Tasks Details

This part provide example task prompts used in our simulations. Each prompt specifies the description, time budget, and required skills.

Simple Task

```
description: "Fix five independent bugs
    across modules: login validation,
    data parsing, UI rendering glitches,
    API timeout handling, and a database
    connection leak. No cross-
    dependencies."
hours: 8.0
skills: ["backend", "frontend", "database
    ", "api", "testing"]
```

Medium Task

```
description: "Integrate an external API
    with authentication, including token
    management and error/latency handling
    ; deliver minimal usage docs."
hours: 24.0
skills: ["backend", "api", "
    authentication", "oauth", "testing",
    "documentation"]
```

Hard Task

```
description: "Build a user authentication
     service covering registration, login
    , password reset, OAuth 2.0 sign-in,
    JWT issuance/refresh, session
    management, and security hardening."
hours: 40.0
skills: ["backend", "security", "database
    ", "oauth", "authentication", "
    frontend", "testing"]
```

B Implementation Details

We provide prompts used in our experiments in this section.

Decompose a task into subtasks

```
Decompose this task into subtasks:

Task: {task.description}
Estimated hours: {task.estimated_hours}
Required skills: {', '.join(task.required_skills)}

Team members:
{self._format_team(team)}
{manager_note}
Create {teamsize} subtasks that:
1. Can be worked on independently or with minimal dependencies
2. Match team members' skills
3. Are reasonably sized (total hours should be close to the original task's estimated hours)
4. Cover all aspects of the original task


Return JSON:
{{
    "subtasks": [
      {{
           "description": "Clear description of what needs to be done",
           "estimated_hours": number,
           "required_skills": ["skill1", "skill2"],
           "suggested_assignee": "team member_name (can be any team member including Manager)
               ",
           "dependencies": [] // indices of subtasks this depends on
       }}
    ],
    "decomposition_rationale": "Brief explanation of your decomposition strategy"
}}
```

Figure 5: Prompt for the manager to decompose a give task.

Intention generation

```
You are {context.agent.name}, a {context.agent.role.value}.

Current situation:
Tasks:
{self._format_tasks_for_prompt(context.current_tasks[:3], context)}
Note: Alignment affects work efficiency.(max: 1.0)
{message_info}
{last_action_info}

Analyze your situation and choose your primary intention for this step:

1. CONTINUE_TASK - Continue working on current tasks
2. CHECK_MESSAGES - Check and potentially respond to messages
3. REQUEST_HELP - Ask for other agents' help
4. NEED_CLARIFICATION - Need clarification on task requirements
5. REPORT_PROGRESS - Report progress to manager
6. SCHEDULE_MEETING - Schedule a meeting

IMPORTANT Decision Factors:
- If has MEETING_START: Almost always CHECK_MESSAGES (meeting is starting!)
- If has MEETING_INVITE: Strongly consider CHECK_MESSAGES (need to RSVP)
- If stuck on task for long: Consider REQUEST_HELP or NEED_CLARIFICATION
- Balance responsiveness with productivity

Return JSON: {{"intention": "INTENTION_NAME", "reasoning": "explanation"}}
```

Figure 6: Prompt agents use to generate their intentions at each step.

Decompose a task into subtasks

```
You are evaluating how much a received message helps an worker understand their task better.

Task Information:
- Task ID: {task_id}
- Description: {task.description}
- Required Skills: {', '.join(task.required_skills) if task.required_skills else 'Not
    specified'}
- Current Progress: {task.actual_hours:.1f}/{task.estimated_hours:.1f} hours
- Current Alignment Factor: {current_alignment:.2f} ({current_alignment*100:.0f}% efficiency)

Message Type: {message.message_type.value if hasattr(message, 'message_type') else 'Unknown'}
From Agent: {message.from_agent_id}

{'Original Request (sent by me):' if original_request else 'Context:'}
{original_request.content if original_request else 'This is a proactive message or the
    original request is not available.'}

Reply Received:
{message.content}

Alignment Factor Guidelines:
- Range: 0.01 (1% efficiency) to 1.0 (100% efficiency)
- Current value: {current_alignment:.2f}
- Alignment can increase OR decrease based on communication quality
- Clear and helpful communication should improve understanding
- Confusing, contradictory, or misleading information may reduce understanding
- Consider the overall impact on task clarity and execution confidence

Consider:
1. How directly the reply addresses the original request/confusion
2. How actionable and specific the information is
3. Whether critical blockers were resolved
4. The completeness of the response
5. Diminishing returns (harder to improve as alignment approaches 1.0)

Return a JSON response:
{{
    "new_alignment_factor": <float between 0.01 and 1.0>,
    "change": <float, positive for increase, negative for decrease>,
    "reasoning": "<brief explanation of why this change in understanding>"
}}
```

Figure 7: Prompt for updating alignment factor.