Before Humans Join the Team: Diagnosing Coordination Failures in Healthcare Robot Team Simulation

Yuanchen Bai[1,2], Zijian Ding[3], Shaoyue Wen[4], Xiang Chang[1] and Angelique Taylor[1,2]

***Abstract*— As humans move toward collaborating with coordinated robot teams, understanding how these teams coordinate and fail is essential for building trust and ensuring safety. However, exposing human collaborators to coordination failures during early-stage development is costly and risky, particularly in high-stakes domains such as healthcare. We adopt an agent-simulation approach in which all team roles, including the supervisory manager, are instantiated as LLM agents, allowing us to diagnose coordination failures before humans join the team. Using a controllable healthcare scenario, we conduct two studies with different hierarchical configurations to analyze coordination behaviors and failure patterns. Our findings reveal that team structure, rather than contextual knowledge or model capability, constitutes the primary bottleneck for coordination, and expose a tension between reasoning autonomy and system stability. By surfacing these failures in simulation, we prepare the groundwork for safe human integration. These findings inform the design of resilient robot teams with implications for process-level evaluation, transparent coordination protocols, and structured human integration.**

I. Introduction

Human–robot interaction is increasingly envisioned to move beyond one-to-one interaction with individual robots toward collaboration with coordinated robot teams that assume complementary roles within shared workflows [1]. Large language model (LLM)-based multi-agent systems (MAS) provide a promising technical foundation for such robot teams, where multiple agents can support robots from capability, interaction, and implementation aspects. Specifically, MAS has emerged as an effective paradigm for solving more diverse and complex tasks (e.g., code generation [2] and debate [3]) compared with single-agent approaches [4]. From an interaction perspective, natural language provides an intuitive interface for coordinating robot teams, making such systems more accessible to both technical and non-technical users, including those without prior robot programming experience [5]. For implementation, researchers have developed multi-agent frameworks, such as AutoGen [6] and CrewAI [7], to ease the creation of MAS.

However, alongside the growing potential of robot teams, it is critical to consider the risks of system failure. Failures in robotic systems, and how they are handled and communicated to human collaborators, can significantly impact both task completion and human trust [8], [9], [10]. While prior work has examined failures in human–robot interaction, it has largely focused on single-robot settings (e.g., [9]). As a result, how coordination breakdowns emerge within robot teams, and what they imply for human oversight and intervention, remains underexplored.

Studying such failures in multi-agent robotic systems (MARS) introduces additional challenges. In high-stakes domains such as healthcare, physical constraints (e.g., limited robots, hardware bottlenecks, and high operational costs) make failures costly and demand efficient resource allocation [11]. Coupled with strict safety and reliability requirements, these challenges call for robust coordination structures that are resilient to failure (e.g., hierarchical structure [12]), rather than ad hoc or unstructured ones. This raises a critical question: whether MARS, built on MAS frameworks originally designed for virtual tasks, can meet the demands of real-world deployment.

However, existing analyses of MAS coordination patterns fall short in capturing real-world complexities. For example, [13] identifies 14 failure modes across three categories, but the analysis is based on virtual task benchmarks such as math problem solving. Prior evaluations focus on task outcomes, lacking finer-grained, process-level assessments [14]. In addition, reasoning capability, an important factor shaping agent behavior, has been primarily examined at the single-agent level (e.g., [15]), with limited understanding of its impact on team-level coordination. Together, these gaps motivate our investigation into MARS coordination patterns and failure modes, toward better supporting humans who build and work alongside robot teams.

To address the absence of existing benchmarks for evaluating MARS under real-world constraints, we construct a custom, controllable scenario capable of systematically injecting critical challenges and boundary conditions—such as team-level recovery logic and hierarchical role interpretation. Among potential domains, healthcare stands out: in high-stakes tasks like emergency room onboarding, robots need to operate under resource constraints, clearly defined roles, and low fault tolerance. Building on this setting, we examine the performance of hierarchical MARS, built upon current state-of-the-art multi-agent frameworks, and analyze the coordination patterns that emerge across contextually grounded scenarios. Importantly, our goal is not to compare frameworks. Rather, we use different system configurations as controlled probes to analyze coordination behaviors and failure patterns.

Following the generative agent simulation paradigm [16],

[1]Yuanchen Bai, Xiang Chang, and Angelique Taylor are with Cornell Tech, New York, USA yb299@cornell.edu, xc529@cornell.edu, amt298@cornell.edu

[2]Yuanchen Bai and Angelique Taylor are also with the Department of Information Science, Cornell University, Ithaca, USA

[3]Zijian Ding is with University of Maryland, College Park ding@umd.edu

[4]Shaoyue Wen is with Imperial College London jw7525@ic.ac.uk

all roles in the robot team, including the supervisory manager that would ultimately be filled by a human healthcare worker, are instantiated as LLM agents. This fully simulated approach provides a controlled testbed for characterizing coordination dynamics as a prerequisite to hybrid human-agent deployment. Our analysis is conducted at the agent-agent level and positions humans as integral to the broader interaction loop, as system designers, collaborators, and ultimate decision-makers to whom failures may be escalated. By characterizing how failures emerge, propagate, and are resolved within robot teams before humans are introduced into the loop, our work provides a foundation for designing more effective human–robot coordination, including how failures should be managed and communicated in collaboration with humans. This "diagnose before deploy" approach ensures that coordination protocols are robust and well-understood prior to human integration, reducing the risk of exposing human collaborators to preventable system failures. In this way, we contribute toward realizing human–robot symbiosis with AI in real-world, high-stakes environments.

Our contributions include:

- **Contributing Factors to Coordination Failures in Hierarchical MARS:** We identify coordination failures in hierarchical MARS and investigate their dependencies on contextual knowledge, system structure, and underlying model reasoning capability. Our findings reveal that while sufficient contextual knowledge is necessary, system structure remains the bottleneck for robust coordination, and different reasoning capabilities give rise to distinct failure profiles.
- **Reasoning Capabilities and Coordination Trade-offs:** We find that strong reasoning models exhibit more advanced planning and team orchestration in our specific scenario, but also introduce more diverse failure patterns due to their reasoning initiatives. Although non-reasoning models show fewer failure patterns in our scenario, this stems not from stronger problem-solving capabilities but from a lack of deliberate reasoning that limits their autonomy and adaptability.
- **Design Implications for Resilient Robot Teams:** Drawing from our empirical findings, we identify three design principles for building robot teams that humans can effectively supervise and collaborate with: process-level evaluation as a foundation for surfacing otherwise invisible coordination failures, transparent coordination protocols that expose team state on demand, and structured human integration that treats human roles within the hierarchy as deliberate design choices.

II. Related Work

A MAS consists of multiple agents that collaborate to achieve a common goal [4]. Hierarchical MAS, noted for their resilience to failure [12], are promising for scaling to larger and more layered agent teams for tasks of higher complexity. While some recent MAS are considered as "hierarchical", they often simplify hierarchy into rigid task-handoffs, overlooking the adaptive, bidirectional structures seen in real organizations of agents. This suggests a revisit to what hierarchy entails and whether incorporating these elements could enhance coordination.

Recent work provides engineering scaffolds to support the exploration of custom MAS. Frameworks such as Microsoft AutoGen [6], CrewAI [7], and LangGraph [17] represent attempts in this direction, providing modular agent construction and communication mechanisms that enable developers to flexibly configure models, tools, and interaction processes. These frameworks have demonstrated promising adaptability in many tasks, including crime trend analysis [18], paper reviews [19], and engineering material model construction [20]. However, once deployed in real-world physical scenarios, their originally language-driven, loosely-structured designs expose vulnerabilities due to limited physical resources and demanding delegation and report-back mechanisms.

To advance MAS applications in real environments, researchers have conducted a systematic analysis of collaborative failure modes such as agent authority overreach, role responsibility conflicts, tool invocation confusion, and feedback chain disruption [13]. Consequently, some research attempts to improve system robustness from the perspective of model capabilities, such as introducing reinforcement learning mechanisms [21], embedding causal or symbolic reasoning modules [22], and modeling temporal dependency structures [23]. Other work emphasizes comprehensive modeling of knowledge structures and improving external environment perception, in an attempt to improve system control over task contexts and information flow [24].

However, even with the above enhancement mechanisms, once collaboration enters high-risk, low-tolerance real-world physical environments, the originally language-driven, loosely-structured MAS architectures still struggle to effectively address failure modes caused by their structural deficiencies [25], [13]. Therefore, we focus on a high-risk, low-tolerance task—medical robotic collaboration—to systematically analyze the failure modes exposed by current mainstream multi-agent frameworks in this scenario, and propose a protocol-constrained MAS design approach to improve system task control, collaborative transparency, and structural fault tolerance.

III. Methodology

We designed a controlled test case in a healthcare setting that simulates real-world complexity, serving as a testbed to examine how hierarchical MARS systems operate under high-stakes conditions. Rather than comparing frameworks, we use different system configurations as probes to examine how three factors (i.e., contextual knowledge, communication structure, and reasoning capability) shape coordination, with the goal of informing human oversight and transparency in robot team deployment.

All roles in the robot team, including the supervisory manager, are instantiated as LLM agents rather than human operators. Following the generative agent simulation paradigm [16], this design choice is deliberate: by using

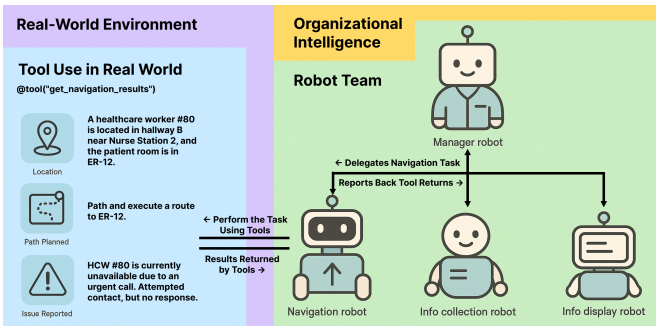

Fig. 1. Overview of our agent-simulated hierarchical MARS for emergency room onboarding. In this scenario, a patient arrives and a healthcare worker (HCW) must be located, credentialed, and assigned. A manager robot (r_m) coordinates three subordinate robots: navigation (r_n) locates and guides the HCW to the patient room, information collection (r_c) gathers HCW credentials and specialty data, and information display (r_d) presents team composition and generates a layout plan. Each subordinate uses a dedicated tool simulating its onboard subsystem, interprets the tool's output, and reports back to the manager. All roles are instantiated as LLM agents to diagnose coordination failures before human healthcare workers join the team.

TABLE I

MARS ROLES AND CORRESPONDING TOOLS.

Robot Role	**Robot Tool (Simulated Subsystem)**
r_n: Locates HCWs and guides them to assigned rooms.	u_n: Simulates internal navigation systems including location tracking, path planning, and staff communication.
r_c: Gathers HCW credentials and specialty data.	u_c: Simulates the onboard interface to collect structured identity and specialty information.
r_d: Presents data and generates layout plan.	u_d: Simulates internal systems to query the institutional database to retrieve and display team roles and composition details.
r_m: Orchestrates the team.	No assigned tools beyond built-in coordination functions (e.g., delegation).

agents to surface coordination failures first, we avoid exposing human healthcare workers to preventable breakdowns during early-stage system development. This allows us to systematically diagnose and address failure modes before humans join the team, establishing a foundation for subsequent hybrid human-agent integration (see Section VI-C).

To illustrate our stepwise exploration of the three factors (See Figure 2), we define experimental settings of the studies as a configuration tuple (κ, σ, ω), where $\kappa \in \{0, 1\}$, $\sigma \in \{0, 1\}$, $\omega \in \{\text{GPT-4o}, \text{o3}\}$. Here, $\kappa = 1$ indicates the inclusion of contextual and procedural knowledge, while $\kappa = 0$ corresponds to its absence. $\sigma = 1$ denotes an enhanced communication structure, and $\sigma = 0$ reflects its absence. ω specifies the underlying model, either GPT-4o-2024-08-06 [26] or o3-2025-04-16 [27], respectively representing non-reasoning and reasoning models.

a) MARS Framework, Robots, Tools and Tasks Setup.: MARS comprises three core components—**robots role** (R), **tasks** (T), and **tools or utilities** (U). While we retain the term "tool" for consistency with LLM-agent frameworks terminology, these tools simulate the subsystems of robots in MARS. Let $R = \{r_m, r_n, r_c, r_d\}$ denote the set of four robots: a manager (r_m) and three subordinates—navigation (r_n), information collection (r_c), and information display (r_d) robots. Each subordinate robot is uniquely equipped with a corresponding tool from the tool set $U = \{u_n, u_c, u_d\}$. We define a one-to-one mapping $\psi : \{r_n, r_c, r_d\} \to U$ such that: $\psi(r_n) = u_n, \psi(r_c) = u_c, \psi(r_d) = u_d$. The manager robot r_m is not assigned any task-execution tools. Instead, it relies solely on built-in coordination functions provided by the framework (e.g., delegation). This reflects its intended role as a high-level planner and leader, rather than an executor of low-level tasks handled by subordinate robots.

The overall task set is denoted as $T = \{\tau_n, \tau_c, \tau_d, \tau_{\text{ref}}\}$. T reflects a minimal but representative coordination workflow adapted from acute-care onboarding. It comprises three execution tasks—navigation (τ_n), information collection (τ_c), and information display (τ_d)—which are expected to be completed by their corresponding subordinate robots, and a reflection task (τ_{ref}) (i.e. to reflect on the overall process and summarize outcomes and lessons learned), which is expected to be handled solely by the manager. We impose a strict precedence relation $\tau_n \prec \tau_c \prec \tau_d \prec \tau_{\text{ref}}$, meaning that each downstream task may start only after its immediate predecessor has been marked *Success* or its failure has been resolved by the manager. This structure reflects the high-stakes nature of real-world settings such as healthcare, where downstream actions (e.g., data interpretation or decision-making) must wait until upstream requirements—such as staff arrival or identity confirmation—are satisfied. Each robot uses and can only use its own tool to complete its assigned task, and then reports the result to the manager. The manager is responsible for validating the task completion outcomes and determining follow-up actions, such as retry or escalation. Table I shows the description of robots' roles and tools.

b) Test Case Design.: We design a test case $\Phi = (O, \mathcal{P}, \{\delta_j\})$, comprising a predefined observation O, prompt pack $\mathcal{P}$, and scripted tool returns δ_j (See Table II). O encodes situational context across all tasks, including observable cues and ambient constraints that agents may use to infer what needs to be done. In practice, O simulates the environment as perceived by robots (e.g., whether upstream failure is resolved) which serve to trigger appropriate tasks and ground their execution. Thus, O plays a dual role: it provides contextual justification for why a task becomes relevant and offers cues needed to reason about how to execute it. $\mathcal{P}$ includes task descriptions, robot roles and any other contextual knowledge. δ_j is a structured, tool-specific output (e.g., a planned avigation path) returned by a robot subsystem. These outputs require *further interpretation* by the robot to assess whether the task has succeeded or failed. When a failure is inferred, the robot must *escalate* the issue by reporting it to the manager for high-level coordination or recovery. This setup offers a controlled yet realistic environment for eliciting agent behaviors in the face of both expected and unexpected outcomes.

To characterize the complexity of MARS coordination,

TABLE II

OBSERVATIONS, CORRESPONDING TASKS, AND EXPECTED OUTCOMES FOR OUR MARS STUDIES.

Observation o_n: Patient Arrival but HCW Unavailable	
Corresponding Task	**Expected Outcome**
τ_n: Navigate the healthcare worker to the assigned patient room	Detect and escalate task failure due to HCW unavailability; perform failure handling
Observation o_c: HCW Reassigned & Collection Begins	
τ_c: Collect identity and specialty data from the newly assigned HCW #90	Recognize that upstream issue has been resolved; successfully retrieve HCW information without issue
Observation o_d: Team Info Collected & Layout Updated	
τ_d: Get updated data and generate a visual layout plan	Display correct team information as prompted and produce a layout plan reflecting current assignments
Observation o_{ref}: Post-Task Reflection Report	
τ_{ref}: Generate a post-hoc summary of team performance	Produce an accurate report summarizing outcomes, reasoning, and lessons learned across prior tasks

we present a structured decision loop to illustrate how robot behavior at each step depends on multiple factors: At each time step t, a robot r selects an action a_t using a policy π_ω instantiated by the underlying model ω, conditioned on the current observation by the robot (e.g., via sensors) O_t, cumulative interaction history H_t, task assigned τ, and prompt pack $\mathcal{P}$: $a_t = \pi_\omega(r, \tau, O_t, H_t, \mathcal{P})$. The action a_t, once executed, updates the environment and leads to new states for future decision steps: $(O_t, H_t) \xrightarrow{a_t} (O_{t+1}, H_{t+1})$.

c) Challenges in High-Stakes Real-World Tasks with Robot Team.: As no established metrics exist for hierarchical MARS, we begin by identifying seven dimensions where real-world constraints introduce distinct operational considerations in MARS, forming the basis for our evaluation criteria and test case design. These criteria reflect coordination challenges that—while some may potentially present in other MAS contexts—have not been systematically examined. In our high-stakes healthcare setting, such challenges become especially pronounced: *1) Agent Characteristics* shift from ephemeral modules to persistent, embodied team members requiring stable identity and accountability. *2) Agent Configuration* becomes role-based, with agents managing a coherent cluster of functions over time. *3) Role Boundaries & Constraints* are tightly scoped due to hardware and authority control. *4) Traceability & Accountability* is heightened, as failures need to be linked to specific agents and actions. *5) Consequence of Upstream Failures* increases, since downstream tasks frequently depend on upstream success without fallback mechanisms. *6) Tool Access & Modularity* is shaped by how tools are embedded within individual robot systems, making them less interchangeable and more tightly bound to agent roles compared to abstract, API-like tools. *7) Definition of Success* becomes holistic—dependent not only on task completion but also on correct sequencing, reporting, and compliance with role expectations.

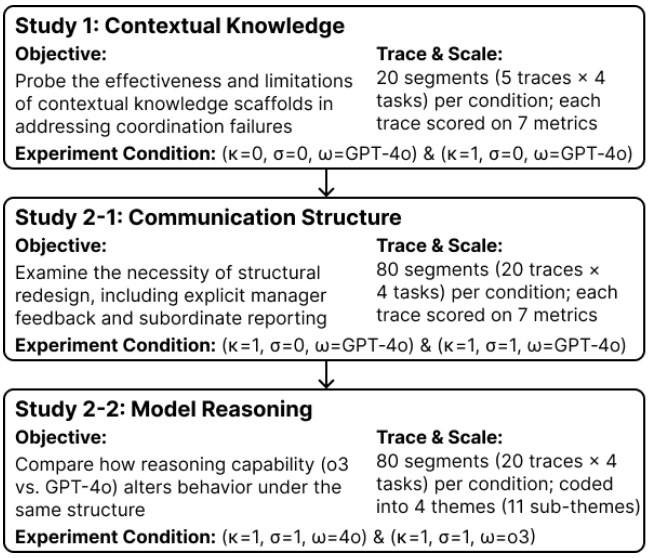

Fig. 2. Study Overview.

IV. STUDY 1: EVALUATION OF HIERARCHICAL MARS COORDINATION

Study 1 serves as an initial diagnostic to screen for typical coordination failures in hierarchical MARS. By providing rich contextual and procedural knowledge ($\kappa = 0 \rightarrow 1, \sigma = 0, \omega =$ GPT-4o), we examine which failures can be addressed through knowledge alone and which persist—thereby revealing potential structural bottlenecks for further investigation (see Figure 2). For evaluation, we ran 5 traces (a trace refers to a full run of the four tasks) per condition.

A. Experiment Setup

1) Hierarchical Structure Setup.: We built on the CrewAI [7] framework using its hierarchical mode–described in the documentation as a structure that "simulates traditional organizational hierarchies for efficient task delegation and execution". CrewAI also features a Knowledge Base (KB) that provides agents with "a reference library they can consult while working." CrewAI is well-suited for our research because these features make it a natural starting point to probe context-grounded behaviors and to identify which coordination challenges can, or cannot, be resolved through contextual knowledge alone.

2) Knowledge Base as Contextual Intervention.: We developed a KB with contextual or procedural knowledge as a shared resource analogous to organizational documentation that ground MARS team behavior and decision-making. The KB defines five knowledge key points, including 1) *tool access rules* through a tool-robot mapping and real-world functions to prevent tool misuse, 2) *role-specific responsibilities* to ensure that robots adhere to defined scope of responsibilities, 3) *task success and failure criteria* to help MARS determine whether to proceed, retry, or escalate, reducing false completions, 4) *environmental cue grounding* to enable MARS to learn how to interpret real-world triggers (e.g., ID scans) for initiating appropriate actions, and 5)

task execution and recovery workflow for clear procedural steps, including escalation paths, enabling MARS to adapt effectively to failure.

a) Metrics.: Our evaluation goes beyond task outcomes to capture process-level dynamics, particularly within the hierarchical structure. We assess the performance of MARS at both the manager and subordinate levels. All metrics are evaluated using a rubric-based scheme with three levels: 0 (criterion not met), 0.5 (partially met), and 1 (fully met). Full task-specific rubrics are provided in the Supplementary Materials.

At the **manager level**, we consider four metrics:

1) Delegation Accuracy (M_1): Whether r_m delegates tasks to the correct robots, based on their role and tool access.
2) Task-Completion Judgment (M_2): Whether the manager correctly assesses the success or failure of each task.
3) Issue Handling (M_3): Whether r_m detects and responds to reported issues in a timely manner.
4) Reflection Quality (M_4): Whether r_m reflects on the task outcomes and the relevant lessons learned captured.

At the **subordinate robot level**, we track three metrics:

1) Tool Usage (M_5): whether the agent uses the correct and accessible tool for the assigned task.
2) Local Reasoning (M_6): whether the agent correctly executes its assigned responsibility and properly interprets the tool's output based on the prompt.
3) Report Compliance (M_7): whether the agent correctly reports the task execution result back to r_m.

b) Scoring.: Let Λ_τ denote the metric set applicable for a task τ and Δ_{trace} denote the set of task–metric pairs present in a trace, where τ indexes a task and k indexes an evaluation metric: $\Delta_{\text{trace}} = \big\{ (\tau, k) \mid \tau \in \text{trace},\ M_k \in \Lambda_\tau \big\}$. Each pair $(\tau, k) \in \Delta_{\text{trace}}$ is independently scored $s_{\text{trace},(\tau,k)} = \rho\big(\text{output}_{\text{trace},\tau},\, M_k\big)$, where $\rho(\cdot, M_k)$ maps the model output to $\{0, 0.5, 1\}$ under metric M_k via the rubric discussed previously. The normalized success rate for a trace is the arithmetic mean of these scores: $\text{SR}_{\text{trace}} = \text{mean}_{(\tau,k)\in\Delta_{\text{trace}}}\, s_{\text{trace},(\tau,k)}$.

B. Results

Table III shows the performance of MARS with and without a KB. A detailed KB leads to an increase in overall success rate—from an average of 45.29% to 72.94%. Overall, KB helps increase performance across most of the seven metrics. Five metrics showed apparent improvement in mean score: delegation accuracy and reflection quality at the manager level, and tool usage, local reasoning, and report compliance at the subordinate level. In contrast, the KB has a minor impact on task completion and issue handling measures. Task completion judgment, achieved near-perfect accuracy with or without the KB, indicating this ability was already strong. Issue handling indicates even with detailed failure handling instructions and manager role-based emphasis (e.g., reinforcing that the manager should respond to reported issues), proactive failure handling remains lacking.

TABLE III
COMPARISON OF AVERAGE PERFORMANCE METRICS WITH AND WITHOUT A KNOWLEDGE BASE (KB).

Metric	κ=0	κ=1
Avg Success Rate (%)	45.29	**72.94**
Delegation Accuracy	0.33	**0.73**
Task Completion	0.93	**0.97**
Issue Handling	0.00	0.00
Reflection Quality	0.30	**0.80**
Tool Usage	0.33	**0.67**
Local Reasoning	0.40	**0.73**
Report Compliance	0.47	**0.77**

Our failure mode analysis reveals that five critical failure modes persist even with a detailed KB. Table IV summarizes these failure modes and reports their frequency across the 10 traces (n/10). Across the 10 traces, eight traces exhibit *hierarchical role misalignment*, where the manager completes tasks intended for its subordinates while delegating tasks that fall under its own leadership responsibilities; all 10 traces contain at least one *tool access violation*, where a tool is used by a robot without the designated permission; and all traces show *lack of in-time handling of failure reports*, in which the manager fails to provide timely alternative solutions or escalate reported problems. Four traces display *noncompliance with prescribed workflows*, where agents either interpret instructions differently or ignore the given prompts; and two traces reveal *bypassing or false reporting of task completion*, where the manager claims a task is complete without actually performing the required actions. Each of these failure types corresponds to at least one KB section that has provided detailed guidelines. Notably, even when an extensive failure recovery protocol is provided, the team consistently fail to detect, escalate, or recover from critical errors. This suggests that the bottleneck does not lie in information availability, but rather in structural limitations that inhibit timely communication and intervention.

V. STUDY 2: STRUCTURAL REDESIGN & MODEL COMPARISON

Study 2 extends the contextual knowledge focus of Study 1 by investigating two additional factors: communication structure and model reasoning. In Study 2-1, we address failure-handling gaps by adding explicit bidirectional communication between manager and subordinates. In Study 2-2, we examine how reasoning capacity influences coordination under the improved structure. CrewAI's hierarchical mode helps surface coordination challenges but lacks transparency and control, prompting our transition to AutoGen, which offers a more customizable design space.

A. Experiment Setup

1) 2-1: Hierarchical Structure Redesign.: We implement a hierarchical structure using AutoGen's [6] "Selec-

TABLE IV
STUDY 1 SUMMARY OF FAILURE MODES AND EXAMPLES.

Failure Mode	Observed Example
Hierarchical role misalignment (8/10)	r_m completes the τ_c without delegating to the r_c
Tool access violations (10/10)	r_m uses the u_c, which should only be accessible to the r_c
Lack of in-time handling of failure reports (10/10)	When the r_n reports an issue (e.g., HCW unavailable), r_m fails to offer alternative solutions or escalate the issue
Noncompliance with prescribed workflows (4/10)	r_m pre-fetches display information and gives it as context to r_d, which then redundantly uses the tool to retrieve the same data again
Bypassing or false reporting of task completion (2/10)	r_m claims the τ_m is complete without actually generating a report (e.g., "Action: None (compiling the final report)")

torGroupChat", where the selector decides which agent is assigned to a specific task. We implement two improvements to the communication structure: (1) *Enabling proactive manager feedback:* To ensure that r_m actively monitors progress, we force r_m to provide timely feedback after each task execution, via a selector function. (2) *Enabling subordinate-level interpretation and report-back:* To allow subordinate agents to reflect on the outcomes of their tool usage (i.e., outputs returned from robot subsystems), we activate the 'reflect_on_tool_use' setting. This enables subordinates to further interpret whether the tool return indicates task success or failure, and to compose a report-back message to r_m. For evaluation, we ran 80 segments (20 traces, 4 tasks per trace) using GPT-4o and re-applied the seven evaluation metrics from Study 1.

2) 2-2: Model Reasoning Comparison.: To analyze how model reasoning influences coordination, we conducted an additional 80 segments (20 traces, 4 tasks per trace) using a strong-reasoning model (o3), compared to GPT-4o. Both of the models are released by OpenAI. Combined with the expanded communication structure introduced in Study 2-1, this setup led to more diverse and complex coordination behaviors. To capture these nuanced patterns, we moved beyond the discrete 0/0.5/1 scoring rubric and adopted the Grounded Theory approach [28], a qualitative method that facilitates theory development from empirical data. This methodology has also been used in recent MAS studies that analyze model traces (e.g. [13]). The first author developed the initial set of codes. To ensure reliability, the codes were collaboratively reviewed by four authors—all with experience in MAS-who iteratively resolved inconsistencies and refined the coding scheme until consensus was reached.

B. Results

1) Structural Redesign Effectiveness for Failure Handling.: The average success rate is 88.97%. We observe strong performance across all seven metrics: delegation accuracy 88%, task completion 88%, issue handling 90%, reflection quality 95%, tool usage 90%, local reasoning 86%, and report compliance 90%. 18/20 GPT-4o traces achieve consistent scores of 1 or 0.5 across all dimensions while in 2/20, the manager fails to delegate any task and hallucinates the entire workflow. Notably, issue handling, which was completely absent in Study 1, now shows marked improvement, with r_m proactively generating alternative plans or escalating unresolved issues to human supervisors. In our setup, human escalation refers to invoking a higher-level external entity (beyond the robot team) when internal resolution proves insufficient. While we acknowledge that some behavioral differences may stem from the underlying framework, the marked improvement in failure handling can be largely attributed to our structural intervention. The observed behavioral improvements, i.e., managers providing real-time feedback and subordinates proactively reassessing outcomes and reporting anomalies, closely align with the bidirectional communication mechanisms introduced in our design. This suggests that addressing structural bottlenecks, such as communication flow, is essential for resolving persistent coordination failures like failure handling.

2) Reasoning Trade-offs.: We identify four major themes in MARS coordination patterns, each comprising several sub-themes (Table V). To contextualize these sub-themes, we annotate each with '✓' or '✗' to indicate whether its implications are positive or negative within our test scenario. We also report the frequency of each sub-theme across 20 traces for both GPT-4o and o3. We find distinct behavioral profiles which underscore trade-offs between reasoning and non-reasoning models:

1) Planning Granularity & Execution Alignment: o3 demonstrates fine-grained, step-by-step planning (1.1) in all 20 traces, often incorporating time thresholds and conditional logic (e.g., if-else), whereas GPT-4o generates only high-level, general plans, highlighting o3's greater initiative in decomposing tasks. o3 also proactively anticipates downstream actions (1.2) in 6 traces, compared to only 1 for GPT-4o, further underscoring its planning ability. However, this strength comes at a cost: o3 deviates from prompt instructions (1.3) in 14 traces, significantly more than GPT-4o (4 traces), reflecting o3's greater tendency to override expected procedures with its own internal logic.

2) Task & Organizational Role Interpretation: o3 exhibits stronger awareness of team roles, initiating cross-role coordination (2.1) in 10 traces versus 1 for GPT-4o, indicating o3's ability to coordinate the robot team to improve task execution. Both models reject tasks outside their scope of responsibility (2.2), but o3 does so slightly more often (5 traces) than GPT-4o (4 traces). o3 triggers human intervention (2.3) in 13 traces, compared to 5 for GPT-4o. When neither model escalates, o3 tends to attempt new solutions, whereas GPT-4o often stalls in unproductive self-reflection.

3) Communication Robustness & Format Compliance: In 9 traces, o3 ignores explicit instructions from the manager and refuses to adjust its output accordingly (3.1). In 11 traces,

o3 produces outputs that deviate entirely from the expected schema (3.2). While GPT-4o occasionally produces minor formatting mismatches, it does not exhibit such schema-breaking behavior. However, o3 demonstrates stronger auditing behavior (3.3): in the 11 traces where formatting errors occur, the manager actively verifies output completeness and explicitly flags missing or malformed fields-a level of diligence absent in GPT-4o. This shows that while reasoning enables more rigorous self-auditing, its occasional refusal to comply with feedback limits the system's ability to recover from detectable failures.

4) Task Termination & Verification: In 20 traces, o3 repeatedly re-executes tasks without providing justification (4.1)—such as generating multiple reflection reports even after a complete one has already been produced—compared to just 1 such instance in GPT-4o. In 11 traces, o3 engages in elaborate domain reasoning to resolve issues; however, this reasoning is often unverified by actual tool returns, leading to unverifiable or factually inaccurate assertions of success (4.2). In contrast, such behavior occurs in 2 traces for GPT-4o, indicating that while elaborate reasoning enables stronger problem-solving ability, it can also increase the risk of ungrounded execution.

VI. Discussion & Future Work

A. Technical Evaluation of Hierarchical MARS

Through two studies using a custom test case in a healthcare MARS scenario, we investigated coordination failures and behavioral trade-offs across non-reasoning and reasoning models—highlighting a deeper tension between autonomy and stability in deploying MARS in real-world settings. **Coordination Failures:** Though contextual knowledge is necessary to improve procedural execution, structure is the bottleneck for performance. It is essential to ensure that agents are not only given clear guidance on what tasks to perform and how, but also structurally enabled to carry out intended behaviors. **Reasoning Trade-off:** Strong reasoning does not guarantee stability of coordination behaviors. o3 demonstrates strong problem-solving capability in orchestrating the team and generating detailed plans, but can become trapped by its own reasoning logic—such as repeatedly requesting information explicitly marked as unnecessary in the prompts, or generating redundant reflection reports after a sufficient one has already been generated. These overthinking behaviors align with prior observations [14] that reasoning models can continue exploring alternatives even after reaching correct solutions. In contrast, GPT-4o, exhibits shallower reasoning than o3, but can still behave unexpectedly. For instance, it asks the information collection robot to address a navigation failure, resembling a desperate attempt to address a perceived impasse without a grounded strategy. Both models exhibit instability—o3 due to overthinking, and GPT-4o due to lack of deliberative reasoning. This suggests that instability does not stem from how much reasoning occurs, but from whether the reasoning style can be properly understood, constrained, aligned, and grounded. Deploying models—regardless of their level of reasoning—to operate with a degree of autonomy requires systematically understanding and managing the challenges of stability.

B. Build and Collaborate with Resilient Robot Team

Our findings point toward a broader design challenge: robot teams in high-stakes environments must be legible not only to each other, but to the humans who supervise and collaborate with them. To build and work with a resilient robot team requires deliberate design across three aspects. **1) Process-Level Evaluation.** Outcome-level metrics are insufficient for understanding robot team behavior-failures such as silent escalation breakdowns and unverified task completion are only detectable through fine-grained, trace-level inspection. Building human-compatible robot teams requires strict process-level evaluation before deployment. **2) Transparent Coordination Protocols.** Beyond evaluation, deployed robot teams must be designed to expose their internal coordination state on demand, for example, structured failure report templates that encode what failed, which agent, and what recovery was attempted, and process-level update mechanisms that allow external observers to query team state without interrupting execution. **3) Structured Human Integration.** Human roles within robot teams must be specified with the same rigor as robot roles: at which hierarchical layer does a human operator sit, under what conditions should the team surface a decision to a human, and what information does that human need to intervene effectively? These are coordination design choices that determine whether human oversight is genuinely integrated or merely reactive.

C. From Agent Simulation to Human-Agent Teams

All team roles in our studies, including the supervisory manager, are played by LLM agents [16]. This fully simulated setup enables controlled analysis of coordination patterns as a necessary first step before deploying robot teams alongside human collaborators. The failures we identify (hierarchical role misalignment, escalation breakdowns, reasoning-compliance tension) are challenges that human supervisors will inevitably encounter.

Our findings inform a hybrid architecture in which human and agent roles are structurally interchangeable. For instance, the manager could be replaced by a human charge nurse who receives failure reports and makes escalation decisions, while subordinate robots handle execution. Future work will investigate hybrid human-robot teams where human healthcare workers selectively assume roles within the hierarchy, enabling us to study how coordination dynamics shift when humans and agents collaborate as interchangeable teammates.

TABLE V

STUDY 2 THEME-LEVEL ANALYSIS WITH TRACE COUNTS FOR GPT-4O AND O3. BOLD = MORE DESIRABLE.

Sub-theme	Description	4o	o3
Theme 1: Planning & Execution Alignment			
1.1 Step-by-Step Planning ✓	Multi-layered conditional guidance with time thresholds and if-else logic	0	**20**
1.2 Downstream Anticipation ✓	Anticipate and initiate downstream tasks without prompting	1	**6**
1.3 Prompt Deviation ✗	Ignore explicit instructions in favor of internal logic	**4**	14
Theme 2: Role Interpretation			
2.1 Cross-role Collab. ✓	Invoke cross-role resources to orchestrate collaboration	1	**10**
2.2 Task Rejection ✓	Clarify role scope and refuse out-of-scope tasks	4	**5**
2.3 Human Escalation ✓	Escalate to human when internal resolution fails	5	**13**
Theme 3: Communication & Compliance			
3.1 Refusal to Coord. ✗	Refuse manager feedback, turning recoverable errors fatal	**0**	9
3.2 Missing Output ✗	Correct tool use but fail to produce structured output	**0**	11
3.3 Manager Auditing ✓	Detect missing fields and flag output errors	**0**	11
Theme 4: Termination & Verification			
4.1 Unjustified Repeat ✗	Re-execute tasks without justification	**1**	20
4.2 Unverified Inference ✗	Elaborate reasoning unverified by tool returns	**2**	11

REFERENCES

[1] Y. Bai, R. Han, N. Parikh, W. Ju, and A. Taylor, "Towards considerate embodied ai: Co-designing situated multi-site healthcare robots from abstract concepts to high-fidelity prototypes," *arXiv preprint arXiv:2602.03054*, 2026.

[2] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun. ChatDev: Communicative Agents for Software Development. [Online]. Available: http://arxiv.org/abs/2307.07924

[3] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving Factuality and Reasoning in Language Models through Multiagent Debate," May 2023.

[4] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang, "A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges," *Vicinagearth*, vol. 1, no. 1, p. 9, 2024.

[5] S. Lu, J. Berger, and J. Schilp, "Extracting robotic task plan from natural language instruction using bert and syntactic dependency parser," in *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2023, pp. 1794–1799.

[6] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," Oct. 2023, arXiv:2308.08155 [cs]. [Online]. Available: http://arxiv.org/abs/2308.08155

[7] CrewAI, "Crewai," https://www.crewai.com/, 2025, accessed: 2025-05-02.

[8] S. Honig and T. Oron-Gilad, "Understanding and resolving failures in human-robot interaction: Literature review and model development," *Frontiers in psychology*, vol. 9, p. 861, 2018.

[9] G. LeMasurier, C. Tagliamonte, J. Breen, D. Maccaline, and H. A. Yanco, "Templated vs. generative: Explaining robot failures," in *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN)*. IEEE, 2024, pp. 1346–1353.

[10] B. Dossett, J. Sharma, J. Gregory, K. Haring, and C. Reardon, "Trust dynamics in augmented reality-mediated human-robot teams: Impact of performance, feedback, and error severity," in *2025 34th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2025, pp. 2487–2494.

[11] A. Taylor, S. Matsumoto, and L. D. Riek, "Situating robots in the emergency department," in *AAAI Spring Symposium on Applied AI in Healthcare: Safety, Community, and the Environment*, 2020.

[12] J.-t. Huang, J. Zhou, T. Jin, X. Zhou, Z. Chen, W. Wang, Y. Yuan, M. Sap, and M. R. Lyu, "On the resilience of multi-agent systems with malicious agents," *arXiv preprint arXiv:2408.00989*, 2024.

[13] M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, D. Klein, K. Ramchandran, M. Zaharia, J. E. Gonzalez, and I. Stoica, "Why Do Multi-Agent LLM Systems Fail?" Apr. 2025.

[14] P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar, "The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity," *arXiv preprint arXiv:2506.06941*, 2025.

[15] R. Liu, J. Geng, A. J. Wu, I. Sucholutsky, T. Lombrozo, and T. L. Griffiths, "Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse," *arXiv preprint arXiv:2410.21333*, 2024.

[16] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 2023.

[17] LangChain, "Langgraph: Multi-agent workflows," https://blog.langchain.com/langgraph-multi-agent-workflows/, 2025, accessed: 2025-08-01.

[18] S. K. Fatima, T. Zubair, N. Ahmed, and A. Khan, "Autogen driven multi agent framework for iterative crime data analysis and prediction," 2025. [Online]. Available: https://arxiv.org/abs/2506.11475

[19] C. Li, X. Hu, M. Xu, K. Li, Y. Zhang, and X. Cheng, "Can large language models be trusted paper reviewers? a feasibility study," 2025. [Online]. Available: https://arxiv.org/abs/2506.17311

[20] C. Tian and Y. Zhang, "Optimizing collaboration of llm based agents for finite element analysis," 2024. [Online]. Available: https://arxiv.org/abs/2408.13406

[21] Y. Yuan and T. Xie, "Reinforce llm reasoning through multi-agent reflection," 2025. [Online]. Available: https://arxiv.org/abs/2506.08379

[22] Z. Wan, Y. Li, X. Wen, Y. Song, H. Wang, L. Yang, M. Schmidt, J. Wang, W. Zhang, S. Hu, and Y. Wen, "Rema: Learning to meta-think for llms with multi-agent reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2503.09501

[23] Z. Wang, K. Wang, Q. Wang, P. Zhang, L. Li, Z. Yang, X. Jin, K. Yu, M. N. Nguyen, L. Liu, E. Gottlieb, Y. Lu, K. Cho, J. Wu, L. Fei-Fei, L. Wang, Y. Choi, and M. Li, "Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2504.20073

[24] N. Krishnan, "Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications," 2025. [Online]. Available: https://arxiv.org/abs/2504.21030

[25] P. He, Y. Xing, S. Dong, J. Li, Z. Dai, X. Tang, H. Liu, H. Xu, Z. Xiang, C. C. Aggarwal, and H. Liu, "Comprehensive vulnerability analysis is necessary for trustworthy llm-mas," 2025. [Online]. Available: https://arxiv.org/abs/2506.01245

[26] OpenAI, "Gpt-4o system card," https://openai.com/index/gpt-4o-system-card/, 2024, accessed: 2025-08-01.

[27] ——, "Gpt-4o system card," https://openai.com/index/introducing-o3-and-o4-mini/, 2025, accessed: 2025-08-01.

[28] B. Glaser and A. Strauss, *Discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.