FROM MAS TO MARS: COORDINATION FAILURES AND REASONING TRADE-OFFS IN HIERARCHICAL MULTI-AGENT ROBOTIC SYSTEMS WITHIN A HEALTHCARE SCENARIO

Yuanchen Bai
Cornell University
yb299@cornell.edu

Zijian Ding
University of Maryland, College Park
ding@umd.edu

Shaoyue Wen
Imperial College London / NYU
sw6352@nyu.edu

Xiang Chang
Cornell University
xc529@cornell.edu

Angelique Taylor
Cornell University
amt298@cornell.edu

ABSTRACT

Multi-agent robotic systems (MARS) build upon multi-agent systems by integrating physical and task-related constraints, increasing the complexity of action execution and agent coordination. However, despite the availability of advanced multi-agent frameworks, their real-world deployment on robots remains limited, hindering the advancement of MARS research in practice. To bridge this gap, we conducted two studies to investigate performance trade-offs of hierarchical multi-agent frameworks in a simulated real-world multi-robot healthcare scenario. In Study 1, using CrewAI, we iteratively refine the system's knowledge base, to systematically identify and categorize coordination failures (e.g., tool access violations, lack of timely handling of failure reports) not resolvable by providing contextual knowledge alone. In Study 2, using AutoGen, we evaluate a redesigned bidirectional communication structure and further measure the trade-offs between reasoning and non-reasoning models operating within the same robotic team setting. Drawing from our empirical findings, we emphasize the tension between autonomy and stability and the importance of edge-case testing to improve system reliability and safety for future real-world deployment. Supplementary materials, including codes, task agent setup, trace outputs, and annotated examples of coordination failures and reasoning behaviors, are available at: https://byc-sophie.github.io/mas-to-mars/.

1 Introduction

As the scaling law for improving the capabilities of Large Language Models (LLMs) begins to encounter limitations such as data shortage and power consumption, researchers are shifting from single-agent models to multi-agent systems (MAS) [1]. MAS has become an efficient paradigm for solving diverse and complex tasks, such as code generation [2], debate [3], and collaborative writing [4]. To ease the creation of MAS, researchers have developed multi-agent frameworks, such as Microsoft AutoGen [5] and CrewAI [6]. However, when transitioning from MAS designed for virtual tasks to Multi-Agent Robotic Systems (MARS), new challenges emerge. For example, 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 [7]. Coupled with strict safety and reliability requirements, these challenges call for robust coordination structures that are resilient to failure (e.g., hierarchical structure [8]), rather than ad hoc or unstructured ones. This highlights the need to assess whether MARS, built on 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, [9] 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 [10]. In addition, reasoning capability, an important factor shaping agent behavior, has been primarily examined at the

single-agent level (e.g., [11]), with limited understanding of its impact on team-level coordination. These gaps motivate our investigation into MARS coordination patterns and the influencing factors toward better guidance for real-world deployment.

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. 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.

2 Related Work

A MAS consists of multiple agents that collaborate to achieve a common goal [1]. Hierarchical MAS, noted for their resilience to failure [8], are promising for scaling to larger and more layered agent teams for tasks of higher complexity. While some recent MAS are considered as "hierarchical" [8], they often simplify hierarchy into rigid task-handoffs, overlooking the adaptive, bidirectional structures seen in real organizations of agents (e.g. [2]). 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 [5], CrewAI [6], and LangGraph [12] 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 [13], paper reviews [14], and engineering material model construction [15]. 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 [9]. Consequently, some research attempts to improve system robustness from the perspective of model capabilities, such as introducing reinforcement learning mechanisms [16], embedding causal or symbolic reasoning modules [17], and modeling temporal dependency structures [18]. 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 [19].

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 [20, 9]. 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.

3 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. Our exploration goes beyond surfacing coordination patterns by analyzing how three factors shape system-level performance: contextual knowledge, communication structures, and model reasoning.

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 contacting assigned staff.
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).

Table 1: MARS roles and corresponding tools.

To illustrate our stepwise exploration of the three factors (See Figure 1), we define experimental settings of the studies as a configuration tuple (κ, σ, ω), where $\kappa \in \{0, 1\}$, $\sigma \in \{0, 1\}$, $\omega \in \{\text{GPT-4o, o3}\}$. 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 [21] or o3-2025-04-16 [22], respectively representing the state-of-the-art non-reasoning and reasoning models as of August 2025.

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 lower-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 1 shows the description of robots' roles and tools.

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 2). 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, 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,

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

Table 2: Observations, corresponding tasks, and expected outcomes for our MARS studies.

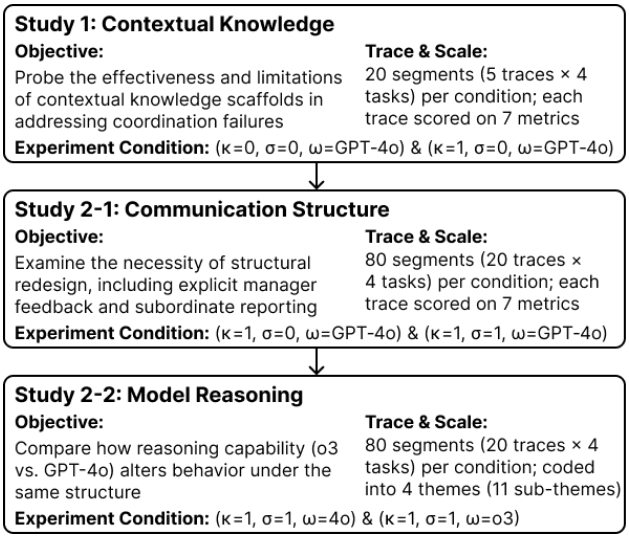

Figure 1: Study Overview.

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: $(O_t, H_t) \xrightarrow{a_t} (O_{t+1}, H_{t+1})$.

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.

4 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 1). For evaluation, we ran 5 traces (a trace refers to a full run of the four tasks) per condition.

4.1 Experiment Setup

4.1.1 Hierarchical Structure Setup.

We built on the CrewAI [6] 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.

4.1.2 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.

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.

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: $\Delta_{\text{trace}} = \{ (\tau, k) \mid \tau \in \text{trace},\ M_k \in \Lambda_\tau \}$. 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: $\text{SR}_{\text{trace}} = \text{mean}_{(\tau,k)\in\Delta_{\text{trace}}}\, s_{\text{trace},(\tau,k)}$.

4.2 Results

Table 3 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.

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**

Table 3: Comparison of average performance metrics with and without a Knowledge Base (KB).

Our failure mode analysis reveals that five critical failure modes persist even with a detailed KB. Table 4 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.

5 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.

5.1 Experiment Setup

5.1.1 Study 2-1: Hierarchical Structure Redesign.

We implement a hierarchical structure using AutoGen's [5] "SelectorGroupChat", 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.

5.1.2 Study 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.

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)")

Table 4: Study 1 summary of failure modes and examples.

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 [23], 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. [9]). 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.

5.2 Results

5.2.1 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.

5.2.2 Reasoning Trade-offs.

We identify four major themes in MARS coordination patterns, each comprising several sub-themes (Table 5). 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.

Sub-theme	Description	Example	4o	o3
Theme 1: Planning Granularity & Execution Alignment				
1.1 Multi-layered Step-by-Step Planning ✓	Construct multi-layered, step-by-step conditional guidance, often involving specific time thresholds and if-else logic with failure-triggering mechanisms, instead of general plan descriptions	r_m: "Allow a strict 2-minute window for acknowledgment… If no response after 2 min, coordinate with the charge nurse… return JSON with the correct HCW ID." *(o3, trace20)*	0	**20**
1.2 Downstream Task Anticipation ✓	Anticipate and initiate downstream tasks autonomously without explicit prompting.	After τ_n is completed, r_m automatically invoked r_d to complete the downstream task τ_d. *(o3, trace5)*	1	**6**
1.3 Prompt Deviation Due to Internal Logic ✗	Selectively ignore explicit prompt instructions in favor of its own inferred logic.	r_m instructed partial display; r_d followed, ignoring both the tool output and the prompt instruction to show it in full. *(o3, trace4)*	**4**	14
Theme 2: Task & Organizational Role Interpretation				
2.1 Cross-role Collaboration Awareness ✓	Proactively invoke cross-role resources and leverages its understanding of robot roles to orchestrate collaboration and enhance task performance.	r_m suggested having the alert information visualized by r_d: "Display an urgent, high-priority banner on the shared information board: 'CODE ASSIST – Patient in ER-12 requires immediate attention...' " *(o3, trace5)*	1	**10**
2.2 Responsibility Clarification & Task Rejection ✓	Explicitly clarify robot role scope and refuses tasks beyond its capabilities.	r_n: "Requested actions (paging HCW #80, notifying the charge nurse, waiting for responses, reassigning staff) fall outside the navigation robot's scope. I am limited to providing location tracking and movement guidance only." *(o3, trace5)*	4	**5**
2.3 Triggering Escalation and Timing Judgment ✓	Choose to escalate to human intervention when encountering issues that are difficult to resolve.	r_m:"Action I am escalating the issue to the human supervisor for resolution (e.g., investigate directory access permissions or collect the caregiver's name manually). ESCALATE " *(o3, trace13)*	5	**13**
Theme 3: Communication Robustness & Format Compliance				
3.1 Refusal to Coordinate with Manager ✗	Refuse communication with the manager, turning recoverable errors into fatal breakdowns.	Though r_m pointed out what's missing and provides a template with the instruction to r_n: "'Please re-issue the result in the exact structure below, filling in the missing fields:', r_n ignores. *(o3, trace13)*	**0**	9
3.2 Successful Tool Use but Missing Output Structure ✗	Fail to output the information in the required structured format as instructed.	Tool returned complete data, with fields sufficient for the expected output, but r_c keeps outputting only "Task Status: success" with missing fields. *(o3, trace13)*	**0**	11
3.3 Manager's Diligent Verification of Output Format ✗	Demonstrate strong output auditing capabilities, with the ability to detect missing fields, incorrect ordering, and other related errors.	r_m detected a mismatch between the actual and expected output: "The task is not yet complete: we still need the required JSON payload containing ID, Name, and Specialty for HCW #90." *(o3, trace13)*	**0**	11
Theme 4: Task Termination & Verification				
4.1 Repeating Tasks without Justification ✗	Redo tasks without explaining the reason for retrying.	r_m redid τ_{ref} with no rationale for repetition, generating multiple reports with no apparent improvements. *(o3, trace1)*	**1**	20
4.2 Unverified Inference and Lack of Grounding ✗	Engage in elaborate domain reasoning to resolve issues, but the reasoning is often unverified by tool returns	r_n reported: "Substitute HCW #82 located at Hallway A..." which r_m conforms the reported success, despite tool return showing '#80 unreachable." *(o3, trace3)*	**2**	11

Table 5: Study 2 integrated theme-level analysis with sub-themes and trace counts for GPT-4o and o3 based on AutoGen. Bolded values indicate which model shows more desirable behavior within a given sub-theme—either through higher frequency (for positive behaviors) or lower frequency (for negative behaviors), depending on the nature of the sub-theme.

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.

6 Discussion & Future Work

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 [10] 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.

Future work includes 1) investigating failure recovery in more deeply layered hierarchies, and 2) exploring diverse edge cases to better characterize failure boundaries.

References

[1] Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

[2] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative Agents for Software Development, June 2024.

[3] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving Factuality and Reasoning in Language Models through Multiagent Debate, May 2023.

[4] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-Agent Collaboration Mechanisms: A Survey of LLMs, January 2025.

[5] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, October 2023. arXiv:2308.08155 [cs].

[6] CrewAI. Crewai. `https://www.crewai.com/`, 2025. Accessed: 2025-05-02.

[7] Angelique Taylor, Sachiko Matsumoto, and Laurel D Riek. Situating robots in the emergency department. In *AAAI Spring Symposium on Applied AI in Healthcare: Safety, Community, and the Environment*, 2020.

[8] Jen-tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Maarten Sap, and Michael R Lyu. On the resilience of multi-agent systems with malicious agents. *arXiv preprint arXiv:2408.00989*, 2024.

[9] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why Do Multi-Agent LLM Systems Fail?, April 2025.

[10] Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad 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.

[11] Ryan Liu, Jiayi Geng, Addison J Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas 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.

[12] LangChain. Langgraph: Multi-agent workflows. `https://blog.langchain.com/langgraph-multi-agent-workflows/`, 2025. Accessed: 2025-08-01.

[13] Syeda Kisaa Fatima, Tehreem Zubair, Noman Ahmed, and Asifullah Khan. Autogen driven multi agent framework for iterative crime data analysis and prediction, 2025.

[14] Chuanlei Li, Xu Hu, Minghui Xu, Kun Li, Yue Zhang, and Xiuzhen Cheng. Can large language models be trusted paper reviewers? a feasibility study, 2025.

[15] Chuan Tian and Yilei Zhang. Optimizing collaboration of llm based agents for finite element analysis, 2024.

[16] Yurun Yuan and Tengyang Xie. Reinforce llm reasoning through multi-agent reflection, 2025.

[17] Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. Rema: Learning to meta-think for llms with multi-agent reinforcement learning, 2025.

[18] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025.

[19] Naveen Krishnan. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications, 2025.

[20] Pengfei He, Yue Xing, Shen Dong, Juanhui Li, Zhenwei Dai, Xianfeng Tang, Hui Liu, Han Xu, Zhen Xiang, Charu C. Aggarwal, and Hui Liu. Comprehensive vulnerability analysis is necessary for trustworthy llm-mas, 2025.

[21] OpenAI. Gpt-4o system card. `https://openai.com/index/gpt-4o-system-card/`, 2024. Accessed: 2025-08-01.

[22] OpenAI. Gpt-4o system card. `https://openai.com/index/introducing-o3-and-o4-mini/`, 2025. Accessed: 2025-08-01.

[23] Barney Glaser and Anselm Strauss. *Discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.