1. Problem

# 1. Problem Analysis

Cegid’s multi-agent chat platform (built on AG2’s Group Chat pattern) currently faces significant challenges with **complex user requests**.

The **AG2 Group Chat pattern** allows multiple LLM-driven agents to converse freely, but it lacks structure for coordinating **multi-step workflows**, **aggregating data** from multiple APIs, and handling **errors** reliably.

The shortcomings can be summarized as follows:

## 1.1 Multi-step Workflows:

Group Chat orchestrates agents via free-form conversation, without an explicit plan for multi-step tasks. This means there is no guaranteed stepwise execution order – agents speak opportunistically, guided only by a GroupChatManager’s next-speaker selection. As a result, complex workflows can derail or loop because the system doesn’t enforce a structured sequence of actions. In a business scenario (e.g. an ERP query that requires subsequent approval then an update), the free-form approach may skip or repeat steps, leading to incomplete transactions or requiring repeated user prompts.

## 1.2 Cross-API Data Aggregation:

In the current pattern, each agent may call a different business API (ERP, CRM, supply chain, etc.) and report its result in the chat. However, there is no dedicated mechanism to **aggregate these outputs** into a coherent answer. The conversation pattern relies on agents themselves to integrate information, which is unreliable. For example, one agent might retrieve inventory data from an ERP, and another fetch customer data from CRM, but unless an agent is explicitly prompted to combine them, the user sees disjointed answers. Structured orchestration frameworks (like LangGraph) explicitly model an “aggregator” step for this purpose, whereas a pure group chat does not. The result is a poor user experience: users must manually piece together information from multiple responses instead of receiving a unified insight.

## 1.3 Error Handling:

The Group Chat pattern lacks robust error-catching or recovery flows. If an agent encounters an exception (e.g. an API call fails or returns invalid data), the default behavior is often to output the error as a message or fall silent. There is no built-in retry logic or alternate path on failure. In practice, a failed API call can halt the workflow without resolution, forcing the user to intervene. By contrast, more structured agent frameworks incorporate error-handling strategies – for instance, a LangGraph workflow can branch to a recovery node on error, and OpenAI’s Swarm will catch a function error and allow the conversation to continue rather than hard-stop. The absence of such mechanisms in the current system means **failed operations lead to dead-ends or confusing outputs**, degrading user trust. For example, if a “Create order” agent crashes due to a missing field, the system may not gracefully ask for clarification or try a fallback, leaving the user with an unhelpful error message.

## **1.4 Impact on Business Operations:**

These limitations directly affect performance and user experience across Cegid’s use cases:

### 1.4.1 ERP queries

In **ERP queries** (e.g. financial reports or inventory checks), multi-step requests (such as “fetch Q4 revenue and then forecast next quarter”) may not execute correctly – the agents might only do the first part or produce separate answers. The user experiences frustration when a single query requires multiple clarifications. High-value workflows like financial close processes involve many dependent steps, which the current system struggles to maintain in order.

### 1.4.2 CRM updates

In **CRM updates** (e.g. finding a client record then updating their status), a free-form agent conversation might retrieve the data but fail to execute the update if not explicitly prompted in sequence. Errors (like insufficient permissions or validation failures) are not handled gracefully, so the user might get a generic failure and need to retry manually. This slows down CRM tasks and reduces confidence in the assistant.

### 1.4.3 Supply chain workflows

In **supply chain workflows**, such as checking stock across warehouses and placing a replenishment order, the need to combine data from multiple APIs is common. Without proper aggregation, the assistant might return inventory levels from one system and separately confirm order placement, but not correlate them (e.g. ensure that the lowest-stock item was ordered). Any API hiccup (network timeout or data format issue) could terminate the entire chain. This leads to inefficiencies and potential business risks (like missing a restock alert) because the AI agent cannot reliably carry out the full workflow end-to-end.

### **1.4.4 Summary**

The table below summarizes these limitations and their impact:

| **Challenge** | **Issue with Group Chat Pattern** | **Impact on Business Use Cases** |
| --- | --- | --- |
| **Multi-step workflows** | Lacks an explicit step-by-step execution plan, relying on unguided agent dialog. Complex, branched tasks are hard to manage in a free-form chat. | Multi-step processes (ERP approvals, multi-stage supply orders) may execute incorrectly or not at all, requiring repeated user intervention and reducing reliability. |
| **Data aggregation across APIs** | No built-in mechanism to combine outputs from multiple agents/APIs; each agent’s result remains separate by default. The conversation doesn’t enforce a final aggregation step. | Users receive fragmented information (e.g. separate ERP and CRM outputs) instead of a unified answer. This increases cognitive load and chance of error when making decisions based on partial data. |
| **Error handling** | Errors/exceptions are not caught systematically. A failed API call or tool use can halt the conversation without recovery. No automatic retries or fallback logic in free chat. | Failed operations lead to abrupt stops or cryptic error messages. Users must manually diagnose or retry, hurting productivity and trust. In critical workflows (finance, procurement), lack of error recovery can cause process delays or data inconsistency. |

Overall, the current AG2 Group Chat approach, while flexible for simple Q&A, is **ill-suited for complex, multi-step business operations**. It negatively impacts performance (through inefficiencies and repeated attempts) and user experience (through incomplete or confusing interactions), highlighting the need for a more structured and intelligent orchestration solution.

2. SOTA

# SOTA

2.1 Context

## 2.1 Framing the Problem in the Broader Context

[**2.1 Framing the Problem in the Broader Context 1**](#_7mqy29ul8sxu)

[2.1.1 Introduction 1](#_n1yavz2a2euh)

[2.1.1 Multi-Agent Systems as a Path to AGI 1](#_ih998segr3fx)

[2.1.2 Major AI Players and Complexity Challenges 1](#_l5ke746ryocj)

[● OpenAI 2](#_9e8ket1xh6nl)

[● Anthropic 2](#_h55upvuf9xly)

[● Others (Progression Toward AGI) 2](#_h7ui1px02xpc)

[2.2.3 Summary 3](#_6a85ya9w9fz)

### 2.1.1 Introduction

The challenges Cegid faces are symptomatic of the broader evolution in **LLMs and multi-agent systems** as we push toward more **agentic AI** and, ultimately, AGI.

Early LLM applications were often single-turn or single-agent, handling one query at a time.

However, enterprise tasks (like those in finance, HR, and retail) demand *composed actions* – chains of reasoning, tool usage, and decisions.

The industry recognizes that enabling AI to handle such complexity autonomously is a key stepping stone toward more general intelligence ([OpenAI’s Level-3: Agents | by Nicholas Domnisch | EE Solutions](https://medium.com/ee-solutions/openai-level-3-agents-agi-275e1615d5dc#:~:text=their%20intended%20use%20case%20and,%E2%80%9D%20%28Analytics%20India%20Magazine)).

### 2.1.2 Multi-Agent Systems as a Path to AGI

Leading AI researchers view multi-agent collaboration and orchestration as crucial for improving reasoning and achieving more generalized problem-solving. OpenAI, for example, has explicitly stated that *“multi-agent is a path to even better AI reasoning.”* ([OpenAI’s Level-3: Agents | by Nicholas Domnisch | EE Solutions](https://medium.com/ee-solutions/openai-level-3-agents-agi-275e1615d5dc#:~:text=their%20intended%20use%20case%20and,%E2%80%9D%20%28Analytics%20India%20Magazine)) In OpenAI’s framework for progress toward AGI, a critical milestone (“Level 3”) involves agents that can autonomously plan and execute tasks with minimal human guidance ([OpenAI’s Level-3: Agents | by Nicholas Domnisch | EE Solutions](https://medium.com/ee-solutions/openai-level-3-agents-agi-275e1615d5dc#:~:text=The%20rapid%20advancement%20of%20artificial,WallStreetPit)) ([OpenAI’s Level-3: Agents | by Nicholas Domnisch | EE Solutions](https://medium.com/ee-solutions/openai-level-3-agents-agi-275e1615d5dc#:~:text=OpenAI%E2%80%99s%20AI%20development%20framework%20consists,%E2%80%9D%20%28%2017%20Analytics%20India)). This implies moving beyond single-step Q&A to systems of agents that **coordinate on complex objectives**. The difficulties Cegid’s platform encounters (maintaining context over multiple steps, integrating diverse information sources, handling unexpected outcomes) are exactly the challenges being addressed to reach that next level of AI capability.

### 2.1.3 Major AI Players and Complexity Challenges

Top AI companies are rapidly innovating to handle multi-step reasoning and action:

#### OpenAI

Recently introduced the *O1* model series (and subsequent *O3*), which incorporate *agentic reasoning loops*. Unlike traditional LLMs that produce an answer in one go, O1 can iteratively break down a problem, step through a solution, and even call tools/functions in the process. This “step-by-step” internal workflow significantly improves its ability to tackle complex prompts. OpenAI’s O1 Pro mode uses these loops to work through prompts systematically, and the upcoming O3 extends these capabilities further with impressive results in complex task benchmarks. In parallel, OpenAI has experimented with a multi-agent orchestration framework called **Swarm**, which introduces structured concepts like *routines* and *handoffs* to coordinate agents (ensuring one agent’s output feeds correctly to the next) ([Introducing OpenAI's Swarm: A New Open-Source Multi-Agent ...](https://www.kommunicate.io/blog/openai-swarm/#:~:text=Introducing%20OpenAI%27s%20Swarm%3A%20A%20New,for%20orchestrating%20multiple%20AI%20agents)) ([Building Intelligent Multi-Agent Systems with OpenAI's Swarm - DZone](https://dzone.com/articles/building-intelligent-multi-agent-systems-with-swarm#:~:text=DZone%20dzone,agentic%20AI%20in%20this%20guide)). These developments indicate OpenAI’s multi-pronged approach: more *intelligent models (O1/O3)* and better *orchestration frameworks (Swarm)* to manage complexity.

#### Anthropic

Anthropic’s Claude models have been designed with a focus on safety and long-form reasoning. The latest Claude versions (e.g. Claude 3.x “Sonnet”) boast very large context windows and strong performance on reasoning tasks ([Claude's agentic future and the current state of the frontier models](https://www.interconnects.ai/p/claudes-agency#:~:text=Claude%27s%20agentic%20future%20and%20the,use%20chat%20models)). Moreover, Anthropic has endowed Claude with the ability to perform actions – for example, Claude can invoke a “computer use” mode to execute commands or use tools as part of its reasoning. This is analogous to an agent calling an API or running a script when needed. By enabling tool use and extended “thinking,” Anthropic is addressing the need for multi-step task handling within a single agent. Claude’s development shows the trend of **integrating reasoning and action** directly into the model’s capability, which is another approach to solving complex workflows.

#### Others (Progression Toward AGI)

Google’s upcoming Gemini model (e.g. *Gemini 2.0* with a “Flash Thinking” mode) and startups like Adept and DeepMind are also pursuing the vision of models that can plan and execute multi-step strategies. Techniques such as **Chain-of-Thought prompting**, **ReAct (Reason+Act)**, and **Self-Refinement** emerged in research and have been quickly adopted into these production models ([LLMs Evolve with Agentic Workflows, Enabling Autonomous Reasoning and Collaboration](https://www.deeplearning.ai/the-batch/llms-evolve-with-agentic-workflows-enabling-autonomous-reasoning-and-collaboration/#:~:text=,model%20to%20act%2C%20evaluate%2C%20reflect)). The idea is to get LLMs not just to answer, but to **think aloud, make decisions, and correct themselves**, much like an agent in a workflow. This broader context underscores that Cegid’s platform is part of a general movement: turning powerful LLMs into **reliable cognitive agents** that can manage elaborate tasks.

### 2.2.3 Summary

In summary, the difficulties with the current system (like multi-step orchestration and error handling) are *not unique*. They reflect the frontier problems AI companies are actively solving on the road to AGI. Both improved model capabilities (as seen in O1, O3, Claude, etc.) and improved multi-agent frameworks (AutoGen/AG2, LangChain’s LangGraph, OpenAI’s Swarm, etc.) are converging to handle such complexity. Cegid’s platform stands to benefit directly from these advances by adopting the latest techniques in multi-agent coordination and reasoning.

2.2 Multi-Agent

## 2.2 Advanced Multi-Agent Reasoning and Coordination

[**2.2 Advanced Multi-Agent Reasoning and Coordination 1**](#_1gntz5t0exz3)

[2.2.1 AG2 Reasoning Agent (Advanced Reasoner) 1](#_srzn9hhox2ei)

[2.2.2 Central Orchestrator Agents: 1](#_2fapkkz29hfw)

[2.2.3 Coordination Strategies 2](#_laqjpvswnx02)

### 2.2.1 AG2 Reasoning Agent (Advanced Reasoner)

The AG2 framework itself has evolved beyond the basic Group Chat.

One of the flagship advancements is the **ReasoningAgent** introduced in AG2 v0.5+, which serves as an “agentic brain” for complex tasks.

The ReasoningAgent implements a ***Tree-of-Thoughts*** approach: it **explores multiple reasoning paths in parallel and uses a grader (evaluation) agent to select the best outcome** ([ReasoningAgent - Tree of Thoughts with Beam Search in AG2 - DEV Community](https://dev.to/ag2ai/reasoningagent-tree-of-thoughts-with-beam-search-in-ag2-3ki1#:~:text=,for%20DPO%20and%20PPO%20training)).

In practice, this means instead of answering directly, the agent can consider several possible solution steps, simulate outcomes, prune bad paths, and refine its approach – much like a human brainstorming and evaluating different strategies.

The underlying algorithms include **Beam Search** (keeping a fixed number of best hypotheses at each step) ([ReasoningAgent - Tree of Thoughts with Beam Search in AG2 - DEV Community](https://dev.to/ag2ai/reasoningagent-tree-of-thoughts-with-beam-search-in-ag2-3ki1#:~:text=,for%20DPO%20and%20PPO%20training)), and options for **Monte Carlo Tree Search (MCTS)** and **LATS (Language Agent Tree Search)** for even more powerful search-based reasoning ([Qingyun Wu's Post - LinkedIn](https://www.linkedin.com/posts/qingyun-wu-183019a6_here-comes-ag2-reasoning-agent-httpslnkdin-activity-7290048934922645505-dk7q#:~:text=Qingyun%20Wu%27s%20Post%20,agent%20and%20Deepseek%20reasoning)) ([Language Agent Tree Search - AG2](https://ag2.airt.ai/0.8.4/docs/use-cases/notebooks/notebooks/lats_search#:~:text=Language%20Agent%20Tree%20Search%20,time%20compute)).

For example, with MCTS or LATS, the agent can incorporate reflection and backtracking: it generates possible next actions, reflects on their results (using a self-critique or external feedback), and backpropagates scores to decide the next exploration step ([Language Agent Tree Search - AG2](https://ag2.airt.ai/0.8.4/docs/use-cases/notebooks/notebooks/lats_search#:~:text=Language%20Agent%20Tree%20Search%20,time%20compute)) ([Language Agent Tree Search - AG2](https://ag2.airt.ai/0.8.4/docs/use-cases/notebooks/notebooks/lats_search#:~:text=2,trajectories%20based%20on%20the%20outcomes)).

This capability is particularly relevant for multi-step business queries – the ReasoningAgent can essentially **plan out a sequence of actions** by internally simulating different multi-step solutions (e.g. “First do X, then Y, if failure do Z”) before committing to a final answer. It’s a significant improvement in reasoning depth and reliability, and is seen as an *“agentic alternative”* to relying purely on a single large model like OpenAI’s O1.

### 2.2.2 Central Orchestrator Agents:

Beyond individual reasoning, the trend is toward specialized coordinator agents that manage other agents. AG2 has introduced concepts like the **CaptainAgent** (sometimes called a “Manager” or “Planner” agent) which is essentially a top-level orchestrator ([Reasoner Agents | Driving controlled autonomy... | by RAJIB DEB | Medium](https://medium.com/@rajib76.gcp/from-rule-aware-to-context-aware-systems-88c747986d88#:~:text=AG2%E2%80%99s%20Captain%20Agent%3A%20A%20Practical,Example)).

This agent can interpret a high-level user goal and **automatically break it into subtasks, then delegate those to expert agents**, overseeing the whole process ([Reasoner Agents | Driving controlled autonomy... | by RAJIB DEB | Medium](https://medium.com/@rajib76.gcp/from-rule-aware-to-context-aware-systems-88c747986d88#:~:text=An%20instantiation%20of%20such%20agent,complex%20systems%2C%20the%20Captain%20Agent)).

It acts as a project manager for the AI team, ensuring each step is completed and integrating the results.

This aligns with Marvin Minsky’s “Society of Mind” theory – individual agents handle small tasks, while a **central reasoning agent (the “builder” or captain) organizes them into a coherent strategy** ([Reasoner Agents | Driving controlled autonomy... | by RAJIB DEB | Medium](https://medium.com/@rajib76.gcp/from-rule-aware-to-context-aware-systems-88c747986d88#:~:text=Now%20that%20we%E2%80%99ve%20established%20the,concept%20of%20the%20Builder%20Agent)) ([Reasoner Agents | Driving controlled autonomy... | by RAJIB DEB | Medium](https://medium.com/@rajib76.gcp/from-rule-aware-to-context-aware-systems-88c747986d88#:~:text=Key%20Advantages%20of%20Reasoner%20Agents%3A)).

The advantage is twofold: (1) **Structured Planning** – the system does not rely on emergent conversation alone, but on an explicit plan devised by the reasoning agent; (2) **Controlled Autonomy** – constraints and safety checks can be applied at the planner level (e.g. the CaptainAgent can decide not to execute a risky action, or to involve a human if needed) ([Reasoner Agents | Driving controlled autonomy... | by RAJIB DEB | Medium](https://medium.com/@rajib76.gcp/from-rule-aware-to-context-aware-systems-88c747986d88#:~:text=Instead%20of%20dispersing%20constraints%20across,enhances%20system%20transparency%20and%20safety)) ([Reasoner Agents | Driving controlled autonomy... | by RAJIB DEB | Medium](https://medium.com/@rajib76.gcp/from-rule-aware-to-context-aware-systems-88c747986d88#:~:text=%2A%20can%20interpret%20high,ensure%20safety%20and%20goal%20alignment)).

State-of-art frameworks often implement this pattern under various names (e.g. a “Supervisor” agent in LangGraph, or a “Crew” lead in CrewAI).

The key point is that **multi-agent systems benefit from a hierarchical structure**: a reasoner/planner agent at the top and specialist agents for each tool or domain beneath.

### 2.2.3 Coordination Strategies

Research has also produced techniques for agent coordination such as agent voting or debate (having multiple agents propose solutions and reconcile) and self-consistency checks.

For example, an agent might produce multiple answers and then another agent (or the same agent in a different mode) evaluates which answer is best – this was inspired by *self-consistency decoding* in LLMs ([LLMs Evolve with Agentic Workflows, Enabling Autonomous Reasoning and Collaboration](https://www.deeplearning.ai/the-batch/llms-evolve-with-agentic-workflows-enabling-autonomous-reasoning-and-collaboration/#:~:text=,model%20to%20act%2C%20evaluate%2C%20reflect)).

AG2’s ReasoningAgent actually uses a form of this: the “thinker” and “grader” roles within it can be seen as two agents (generator and evaluator) working together ([ReasoningAgent - Tree of Thoughts with Beam Search in AG2 - DEV Community](https://dev.to/ag2ai/reasoningagent-tree-of-thoughts-with-beam-search-in-ag2-3ki1#:~:text=,for%20DPO%20and%20PPO%20training)).

Such patterns echo human collaborative problem-solving and are becoming standard in advanced agent systems to improve result quality and reliability.

2.3 LLMs

## 2.3 State-of-the-Art LLMs for Reasoning and Planning

### **OpenAI O1 and O3 Models**

OpenAI’s *O1* (introduced in late 2024) is a cutting-edge large language model specifically geared towards agentic behavior.

It features an internal mechanism to **work through tasks step by step**, essentially performing chain-of-thought internally.

O1 can plan intermediate steps, call functions (via OpenAI’s function calling interface), and adjust its approach based on tool outputs – all within one prompt/response cycle.

This greatly improves its ability to handle multi-step queries without external orchestration.

O1 was followed by *O3*, an even more advanced model with extended context and refined reasoning loops.

Early reports indicate O3 can solve more complex problems and make fewer reasoning errors, effectively extending what O1 started.

These models are relevant to Cegid’s platform because they reduce the “prompt engineering” burden – an O1-powered agent can on its own break down a command like *“Find the top 5 customers by revenue and update their status to VIP in CRM”* into a sequence of actions, whereas earlier models (GPT-3/4) might require explicit step-by-step prompting or an external planner.

However, using O1/O3 alone in a multi-agent system still benefits from oversight (to handle inter-agent coordination), but they bring much stronger built-in reasoning to each agent.

### **Anthropic Claude**

Anthropic’s latest *Claude* models (e.g. Claude 3.x series) are also at the forefront of reasoning.

Claude is known for a very large context window (allowing it to consider long dialogues or many documents at once) and for a training focus on harmlessness and helpfulness.

In terms of planning, Claude has demonstrated the ability to produce detailed step-by-step plans when asked, and with the newest updates, it can even execute those plans by controlling tools.

As noted earlier, Claude 3.5 “Sonnet” introduced a feature where the model can output special commands to, say, browse the web or use a terminal. In a multi-agent context, this means a single Claude agent can act somewhat autonomously, deciding when to call an API tool.

Claude’s strength is often in **robust natural language understanding** and maintaining coherent narratives (useful for summarizing aggregated data or explaining results), which could complement OpenAI models.

**Other Notable Models:**

The landscape also includes **DeepSeek’s R1** and **Google’s Gemini**, which have been mentioned as incorporating similar agentic reasoning capabilities. These models are built to handle *decision-making and tool use* as part of their core functionality. For instance, Google’s Gemini (successor to PaLM/LaMDA family) is expected to integrate strong planning abilities possibly influenced by DeepMind’s work in reinforcement learning. While Cegid’s focus will likely be on OpenAI or Anthropic due to Azure integration, it’s good to note that the capabilities (iterative reasoning, large context, tool integration) are becoming standard in top-tier LLMs by 2025.

In summary, the state-of-art LLMs today are far more capable for reasoning and planning than previous generations. Models like O1/O3 and Claude can serve as the “brains” of agents, able to interpret complex instructions, break them into steps, and even execute or call tools in mid-thought. Leveraging these in Cegid’s platform (via Azure OpenAI service or Anthropic’s API) will significantly boost the agents’ ability to handle the multi-step workflows and decision-heavy tasks required.