Emergency Response: Cooperation and Coordination Mechanisms in Multi-Agent Systems

Your Name

December 1, 2024

Contents

1 Introduction

In this report, we present the proposed cooperation and coordination mechanisms for the CrewAI emergency response problem. The mechanisms are structured into three main components:

1. Process Definition for individual crews.
2. Pydantic Outputs for structured data handling.
3. Agent Interaction between different crews.

2 Process Definition

2.1 Public Communication Crew Sequential Process

The Public Communication Crew operates within a structured sequential process to ensure efficient and accurate communication of fire incident reports to the public. Each task is assigned to a specific agent with well-defined responsibilities, as detailed below:

1. **Receive Report:** The *Communication Operator* obtains the fire incident report in Markdown format. This serves as the starting point for the process and can filter any information that is not relevant for this crew.

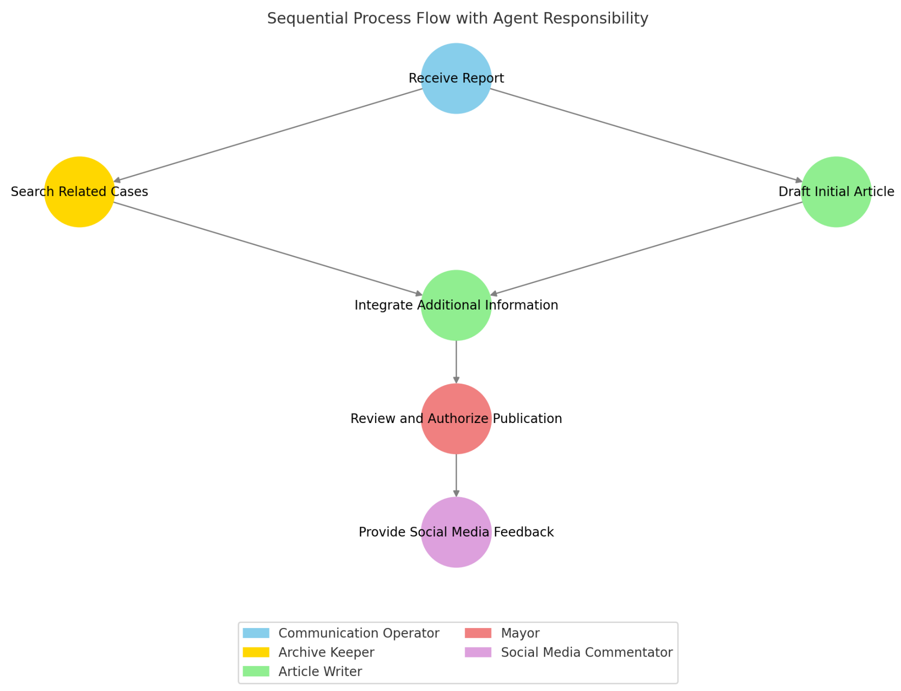
2. **Search Related Cases:** The *Archive Keeper* searches for past incidents with similar locations or fire types. This task depends on the completion of the *Receive Report* task.

3. **Draft Initial Article:** The *Article Writer* drafts an initial article based on the current report. This task also depends on the completion of the *Receive Report* task.

4. **Integrate Additional Information:** The *Article Writer* integrates insights from related cases into the draft. This task requires the completion of both the *Search Related Cases* and *Draft Initial Article* tasks.

5. **Review and Authorize Publication:** The *Mayor* reviews the article and either authorizes publication or provides feedback for revisions. This task depends on the completion of the *Integrate Additional Information* task.

6. **Provide Social Media Feedback:** The *Social Media Commentator* critiques the emergency response in a humorous yet constructive manner. This task depends on the approval of the article by the *Mayor*.



Sequential Process Flow with Agent Responsibility
Receive Report
Search Related Cases
Draft Initial Article
Integrate Additional Information
Review and Authorize Publication
Provide Social Media Feedback
Communication Operator
Archive Keeper
Article Writer
Mayor
Social Media Commentator

Figure 1: Sequential Process Flow of the Public Communication Crew with Agent Responsibilities

Task Dependencies The sequential process relies on strict task dependencies to ensure an organized workflow:

- *Search Related Cases* and *Draft Initial Article* can be executed in parallel but both depend on *Receive Report*.
- *Integrate Additional Information* requires the completion of both *Search Related Cases* and *Draft Initial Article*.
- *Review and Authorize Publication* depends on *Integrate Additional Information*.
- *Provide Social Media Feedback* requires article approval from the *Mayor*.

The visual representation in Figure 1 highlights these dependencies and assigns colors to denote the responsible agents, ensuring clarity and accountability.

3 Pydantic Outputs

Structured outputs are essential for ensuring clarity and consistency in task execution. Below are listed the Pydantic models used in the system.

3.1 Pydantic Outputs for Public Communication Crew

Structured outputs are crucial for ensuring clarity, consistency, and seamless integration across tasks. Below are the Pydantic models designed for the tasks in the Public Communication Crew process:

3.1.1 Receive Report Task Output

```python
from pydantic import BaseModel

class ReceiveReportOutput(BaseModel):
    report_id: str
    location: str
    fire_type: str
    timestamp: str
    markdown_content: str
```

Listing 1: Pydantic model for Receive Report Task Output

3.1.2 Search Related Cases Task Output

```python
from pydantic import BaseModel
from typing import List

class RelatedCase(BaseModel):
    case_id: str
    location: str
    fire_type: str
    summary: str

class SearchRelatedCasesOutput(BaseModel):
    related_cases: List[RelatedCase]
    total_cases: int
```

Listing 2: Pydantic model for Search Related Cases Task Output

3.1.3 Draft Initial Article Task Output

```python
from pydantic import BaseModel

class DraftArticleOutput(BaseModel):
    title: str
    draft: str
    author: str
```

Listing 3: Pydantic model for Draft Initial Article Task Output

3.1.4 Integrate Additional Information Task Output

```python
from pydantic import BaseModel

class IntegratedArticleOutput(BaseModel):
    draft: str
    integrated_sources: list[str]
```

Listing 4: Pydantic model for Integrate Additional Information Task Output

3.1.5 Review and Authorize Publication Task Output

```python
from pydantic import BaseModel

class ReviewOutput(BaseModel):
    approved: bool
    comments: str
    draft: str
```

Listing 5: Pydantic model for Review and Authorize Publication Task Output

3.1.6 Provide Social Media Feedback Task Output

```python
from pydantic import BaseModel

class SocialMediaFeedbackOutput(BaseModel):
    feedback: str
    draft: str
    approved: bool
    comments: str
```

Listing 6: Pydantic model for Provide Social Media Feedback Task Output

Summary of Outputs

- **Receive Report Task Output:** Captures the initial fire incident report relevant details.
- **Search Related Cases Task Output:** Retrieves relevant historical cases for contextualization.
- **Draft Initial Article Task Output:** Records the initial draft content.
- **Integrate Additional Information Task Output:** Updates the draft with integrated sources and revisions.
- **Review and Authorize Publication Task Output:** Specifies the review status and comments from the Mayor.
- **Provide Social Media Feedback Task Output:** Details feedback posted on social media platforms, he can critize the mayor's decission.

4 Agent Interaction

4.1 Interaction Flow

Interaction between crews is designed using a flow-based approach:

- The Search and Rescue Crew provides victim data to the Medical Response Crew.
- A **Router** determines the priority of medical cases based on data received.

4.2 Router Implementation

The router ensures tasks are efficiently allocated:

```
def router(victim_data):
    if victim_data["priority"] == 1:
        return "Critical Response Team"
    else:
        return "General Response Team"
```
Listing 7: Router Implementation for Task Allocation

4.3 Communication Mechanism

Agents communicate using a message-passing protocol to ensure scalability. An example interaction is illustrated in Figure **??**.

5 Conclusion

This report outlines the proposed cooperation and coordination mechanisms for the CrewAI MAS. Future work includes extending the interaction model to include additional agent types and testing the scalability of the system.