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

Sheena Maria Lang, Antonio Lobo Santos, Zachary Parent,
María del Carmen Ramírez Trujillo and Bruno Sánchez Gómez

December 2, 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

1. **Receive and Assess Call.** The *Emergency Call Agent* receives incoming calls and collects relevant details about the incident. The information that this agent receives answers the following six questions and is saved in a report:
   - Is it an indoor or outdoor fire? The answer will be either *outdoor* or *indoor*.
   - Where is it? The location is received as coordinates (x, y).
   - Is anyone inside or trapped? The answer will be an integer number M representing the number of trapped people. If $M > 0$, rescues are needed, and the *Notification Agent* will detail that to the Fire Fighters Crew.
   - Are there hazards? The answer will be a boolean: *True* (yes) or *False* (no). Examples of hazards could include gas cylinders, chemicals, explosions, etc.
   - How big is the fire? The fire will be classified as either *large* (e.g., spreading rapidly), *medium* (e.g., smoke visible), or *small* (e.g., small flame), based on the assessment by the *Emergency Call Agent*.
   - Is anyone injured? How badly? The answer will be a tuple containing an integer N representing the number of injured people and another tuple listing the risk level of each person. They could be classified based on their injuries as *high risk*, *risk*, or *out of risk*. If there are no victims, then $N = 0$ and the second element of the tuple will be an empty list.
2. **Assign Fire Priority Level.** The *Emergency Call Agent* categorizes the incident and sends initial notifications to the *Notification Agent*. The emergency can be classified into the following levels:
   - **Classifying the Fire Based on the Answers:** The fire is classified into three levels:
     - **High Priority:**
       * Indoor fire or outdoor fire with significant hazards (e.g., chemicals, gas).
       * $M > 0$ (people are trapped inside).
       * Large fire spreading rapidly.
       * High-risk injuries or multiple victims.

       Fires meeting these criteria require immediate and large-scale responses to prevent further harm or fatalities.

- **Medium Priority:**
  - Outdoor fire with no hazards or small to medium size fire.
  - No trapped people, but some individuals at risk or with moderate injuries.
  - Medium-risk injuries or a small number of victims with manageable injuries.

  Fires that are moderately dangerous, but not life-threatening, or where injuries are serious but not critical, fall into this category. These require a quick response but lack the urgency of high-priority fires.
- **Low Priority:**
  - Outdoor fire with no hazards and small size.
  - No trapped people or minimal injuries.
  - Low-risk injuries (minor burns or cuts).

  These fires are less dangerous, with no immediate risk to life or property. A single fire engine may be sufficient to manage the situation.

3. **Notify Other Crews.** The *Notification Agent* receives the report and fire classification, then communicates the information to the appropriate crews (Medical Services Crew and Fire Fighters Crew):

- Information provided to the Fire Fighters:
  - Location (x, y)
  - Priority level (*low*, *medium*, or *high*)
  - Number of trapped people M
- Information provided to the Medical Services Crew:
  - Location (x, y)
  - Number of injured people N
  - A list of the risk levels of each injured person (*out of risk*, *risk*, or *high risk*)

2.1 Firefighter Agent Crew

The Firefighter Agent Crew operates within a structured **sequential process** to ensure effective and coordinated response to fire emergencies. Each task is assigned to a specific agent with well-defined responsibilities, as detailed below:

1. **Receive Fire Report:** The *Fire Chief* receives a fire report from the Emergency Service Operator. This serves as the starting point of the process, containing critical information about the location and severity of the fire.

2. **Analyze Fire Report:** The *Fire Chief* analyzes the report to extract key details, such as the type of fire, potential hazards, and necessary resources.

3. **Give Instructions:** The *Fire Chief* relays instructions based on the analysis to the other agents in the crew.

4. **Gather Materials:** The *Equipment Technician* uses the provided instructions to determine which materials are required, such as fire hoses, extinguishers, and personal protective equipment, and ensures they are packed and ready.

5. **Determine Fastest Route to Fire:** The *Fire Chief* calculates the fastest route to the fire location using real-time mapping tools.
6. **Determine Fire Attack Strategy:** Upon arrival, the *Fire Combatants* assess the fire scene and coordinate with each other to devise the most effective strategy for extinguishing the fire.
7. **Combat Fire:** The *Fire Combatants* use the gathered materials and implement the attack strategy to extinguish the fire.
8. **Notify Medical Services If Needed:** If any injuries or health risks arise at the scene, the *Fire Combatants* immediately notify medical services.
9. **Report Back to Emergency Central:** The *Fire Chief* provides updates to the Emergency Service Operator about the situation and the crew's progress, ensuring that central command remains informed.

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

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

- *Analyze Fire Report* depends on the completion of *Receive Fire Report*.
- *Give Instructions* depends on *Analyze Fire Report*.
- *Gather Materials* depends on *Give Instructions*.
- *Determine Fastest Route to Fire* can proceed in parallel with *Gather Materials* but depends on *Analyze Fire Report*.
- *Determine Fire Attack Strategy* and *Combat Fire* depend on the crew's arrival at the fire location.
- *Notify Medical Services If Needed* can occur independently based on the situation at the scene.
- *Report Back to Emergency Central* depends on the crew's progress during and after extinguishing the fire.

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

2.2 Medical Services Crew

The Medical Services Crew operates follows a **sequential** task structure to plan the treatment and evacuation of injured people from the emergency site. The tasks included within the Medical Services are:

1. **Receive Report:** The *Medical Services Operator* receives the fire incident report, and parses key information, such as the location, the number of injured, and the severity of injuries.

2. **Rank Hospitals:** The *Hospital Coordinator* ranks the city's hospitals based on distance to the emergency location.

3. **Allocate Hospital Resources:** The *Hospital Coordinator* assesses the available resources (beds, ambulances, paramedics) at the hospitals, and allocates their resources according to the needs of the emergency.

4. **Deploy Paramedics:** The *Paramedics* are deployed to the place of the incident, reporting an estimation of the time of arrival and a list of the medical procedures that will have to be performed.

5. **Report Medical Response:** The *Medical Services Operator* reports back a comprehensive summary of the response plan.

Task Dependencies The sequential nature of the process requires to establish task dependencies to define the crew's workflow:

- The *Rank Hospitals* task depends on the completion of the *Recieve Report* task.
- The *Allocate Hospital Resources* task depends on the completion of *Rank Hospitals*.
- The *Deploy Paramedics* task depends on the completion of *Allocate Hospital Resources*.
- The *Report Medical Response* task depends on the completion of *Deploy Paramedics*.

The task dependencies and agents who perform each task can be observed in Figure 2.

2.3 Public Communication Crew

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.

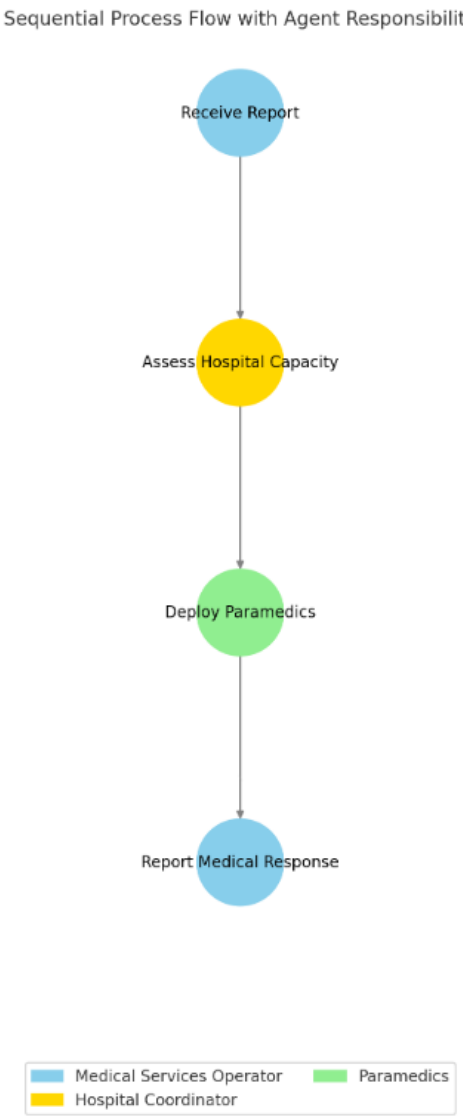

Figure 2: Sequential Process Flow of the Medical Services Crew with Agent Responsibilities

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*.

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*.

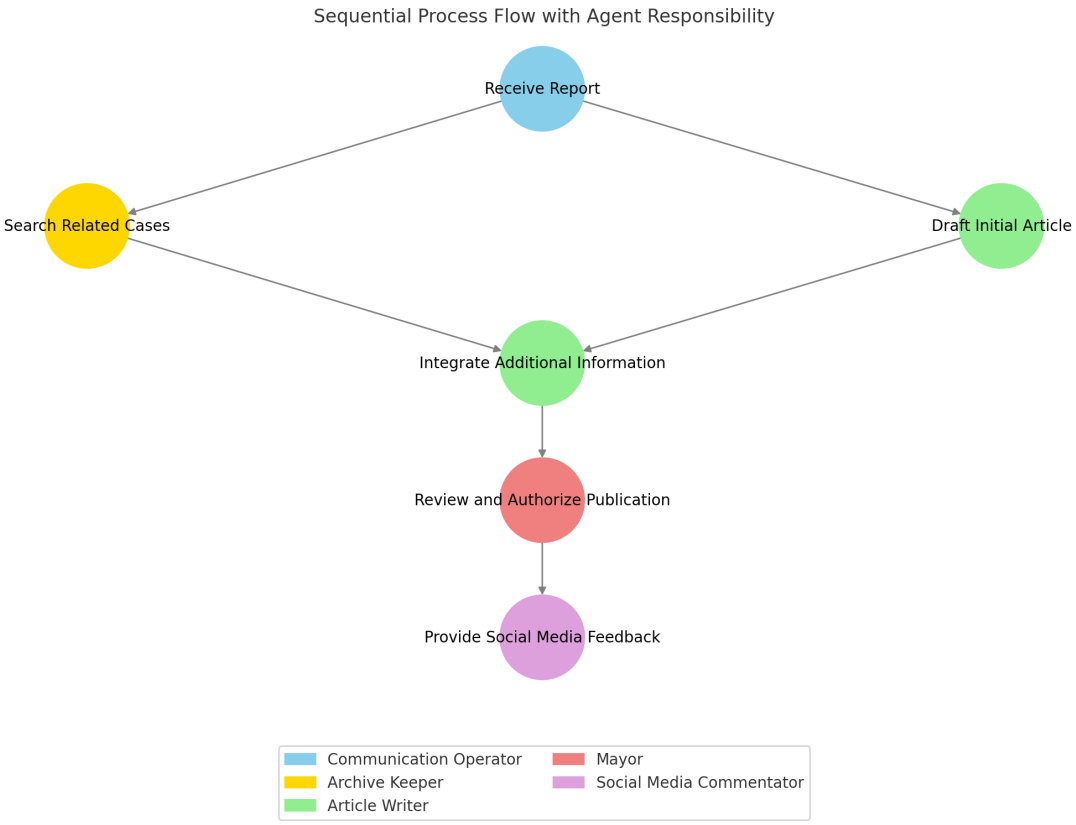

Figure 3: Sequential Process Flow of the Public Communication Crew with Agent Responsibilities

- *Review and Authorize Publication* depends on *Integrate Additional Information*.
- *Provide Social Media Feedback* requires article approval from the *Mayor*.

The visual representation in Figure 3 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.

4 Emergency Services Crew

Structured outputs ensure accurate information handling and effective communication within the Emergency Services Crew. Below are the Pydantic models designed for each task's output.

4.1 Receive and Assess Call Task Output

```
class CallAssessmentOutput(BaseModel):
    fire_type: str  # 'indoor' or 'outdoor'
```

```
    location: Tuple[float, float]  # (x, y) coordinates
    trapped_people: int  # M (number of trapped individuals)
    hazards_present: bool  # True or False
    fire_size: str  # 'small', 'medium', or 'large'
    injuries: Tuple[int, List[str]]  # (N, [risk levels])
    timestamp: str
```

4.2 Assign Fire Priority Level Task Output

```
class FirePriorityOutput(BaseModel):
    priority_level: str  # 'low', 'medium', or 'high'
    timestamp: str
```

4.3 Notify Other Crews Task Output

```
class CrewNotificationOutput(BaseModel):
    fire_fighters_info: dict  # {'location': (x, y), 'priority': str, 'trapped': int}
    medical_services_info: dict  # {'location': (x, y), 'injured_count': int, 'injury
    notification_status: bool
    timestamp: str
```

This format ensures the clarity and consistency required for seamless integration within the system.

4.4 Firefighter Agent Crew

Structured outputs ensure effective communication and accountability among team members in the Firefighter Agent Crew. Below are the Pydantic models designed to encapsulate outputs for each task in the firefighting process:

4.4.1 Receive Fire Report Task Output

```python
from pydantic import BaseModel

class FireReportOutput(BaseModel):
    report_id: str
    location: str
    severity: str
    hazards: list[str]
    timestamp: str
```

Listing 1: Pydantic model for Receive Fire Report Task Output

4.4.2 Analyze Fire Report Task Output

```python
from pydantic import BaseModel

class AnalyzeReportOutput(BaseModel):
    key_details: str
```

```
    risks_identified: list[str]
    resource_recommendations: list[str]
```

Listing 2: Pydantic model for Analyze Fire Report Task Output

4.4.3 Give Instructions Task Output

```
from pydantic import BaseModel

class InstructionsOutput(BaseModel):
    instructions: str
    assigned_roles: dict
    additional_notes: str
```

Listing 3: Pydantic model for Give Instructions Task Output

4.4.4 Gather Materials Task Output

```
from pydantic import BaseModel

class MaterialsOutput(BaseModel):
    materials_list: list[str]
    readiness_status: str
```

Listing 4: Pydantic model for Gather Materials Task Output

4.4.5 Determine Fastest Route Task Output

```
from pydantic import BaseModel

class RouteOutput(BaseModel):
    fastest_route: str
    estimated_arrival_time: str
    alternate_routes: list[str]
```

Listing 5: Pydantic model for Determine Fastest Route Task Output

4.4.6 Determine Fire Attack Strategy Task Output

```
from pydantic import BaseModel

class FireAttackStrategyOutput(BaseModel):
    strategy: str
    assigned_zones: dict
    anticipated_risks: list[str]
```

Listing 6: Pydantic model for Determine Fire Attack Strategy Task Output

4.4.7 Combat Fire Task Output

```python
from pydantic import BaseModel

class CombatFireOutput(BaseModel):
    progress_status: str
    resources_used: list[str]
    obstacles_encountered: list[str]
```

Listing 7: Pydantic model for Combat Fire Task Output

4.4.8 Notify Medical Services Task Output

```python
from pydantic import BaseModel

class MedicalNotificationOutput(BaseModel):
    medical_notified: bool
    reason: str
    injured_personnel: int
```

Listing 8: Pydantic model for Notify Medical Services Task Output

4.4.9 Report Back to Emergency Central Task Output

```python
from pydantic import BaseModel

class EmergencyReportOutput(BaseModel):
    situation_summary: str
    success_status: bool
    time_to_containment: str
    post_incident_notes: str
```

Listing 9: Pydantic model for Report Back to Emergency Central Task Output

Summary of Outputs

- **Receive Fire Report Task Output:** Captures critical details from the initial fire report, including location, severity, and hazards.
- **Analyze Fire Report Task Output:** Summarizes the analysis of the fire report with key risks and recommended resources.
- **Give Instructions Task Output:** Outlines the instructions, role assignments, and additional guidance provided to the crew.
- **Gather Materials Task Output:** Details the materials prepared and their readiness status.
- **Determine Fastest Route Task Output:** Specifies the optimal route to the fire and alternative options.

- **Determine Fire Attack Strategy Task Output:** Documents the firefighting strategy, assigned zones, and risk assessments.
- **Combat Fire Task Output:** Tracks firefighting progress, resources used, and encountered challenges.
- **Notify Medical Services Task Output:** Records the details of medical notifications made and injuries encountered.
- **Report Back to Emergency Central Task Output:** Summarizes the incident status, success metrics, and lessons learned post-response.

4.5 Medical Services Crew

Structured outputs ensure consistency and facilitate effective collaboration among agents within the Medical Services Crew. Below are the Pydantic models for each task's output:

4.5.1 Receive Report Task Output

```python
from pydantic import BaseModel

class ReceiveReportOutput(BaseModel):
    report_id: str
    location: str
    number_of_injured: int
    severity_levels: list[str]
    timestamp: str
```

Listing 10: Pydantic model for Receive Report Task Output

4.5.2 Rank Hospitals Task Output

```python
from pydantic import BaseModel

class Hospital(BaseModel):
    hospital_id: str
    location: str
    available_beds: int
    available_ambulances: int
    available_paramedics: int

class RankHospitalsOutput(BaseModel):
    ranked_hospitals: List[Hospital]
    timestamp: str
```

Listing 11: Pydantic model for Rank Hospitals Task Output

4.5.3 Allocate Hospital Resources Task Output

```python
from pydantic import BaseModel

class HospitalResources(BaseModel):
    hospital_id: str
    beds_provided: int
    ambulances_dispatched: int
    paramedics_deployed: int

class AllocateHospitalResourcesOutput(BaseModel):
    hospital_resource_allocation = List[HospitalResources]
    timestamp: str
```

Listing 12: Pydantic model for Allocate Hospital Resources Task Output

4.5.4 Deploy Paramedics Task Output

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

class MedicalProcedure(BaseModel):
    procedure_name: str
    priority_level: str
    requires_ambulance_transport: bool

class DeployParamedicsOutput(BaseModel):
    paramedics_deployed: int
    estimated_arrival_time: str
    procedures: List[MedicalProcedure]
```

Listing 13: Pydantic model for Deploy Paramedics Task Output

4.5.5 Report Medical Response Task Output

```python
from pydantic import BaseModel

class MedicalResponseReportOutput(BaseModel):
    summary: str
    timestamp: str
```

Listing 14: Pydantic model for Report Medical Response Task Output

Summary of Outputs

- **Receive Report Task Output:** Captures the key details of the fire incident, including injury data.
- **Rank Hospitals Task Output:** Ranks the available hospital based on distance to the emergency site.

- **Allocate Hospital Resources Task Output:** Summarizes the resources provided by each hospital for emergency medical care.
- **Deploy Paramedics Task Output:** Reports the deployment plan and medical procedures to be performed.
- **Report Medical Response Task Output:** Provides an overall response plan.

4.6 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:

4.6.1 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 15: Pydantic model for Receive Report Task Output

4.6.2 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 16: Pydantic model for Search Related Cases Task Output

4.6.3 Draft Initial Article Task Output

```python
from pydantic import BaseModel

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

```
    draft: str
    author: str
```

Listing 17: Pydantic model for Draft Initial Article Task Output

4.6.4 Integrate Additional Information Task Output

```
from pydantic import BaseModel

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

Listing 18: Pydantic model for Integrate Additional Information Task Output

4.6.5 Review and Authorize Publication Task Output

```
from pydantic import BaseModel

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

Listing 19: Pydantic model for Review and Authorize Publication Task Output

4.6.6 Provide Social Media Feedback Task Output

```
from pydantic import BaseModel

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

Listing 20: 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.

5 Agent Interaction

5.1 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.

5.2 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 21: Router Implementation for Task Allocation

5.3 Communication Mechanism

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

6 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.