Sistema Multi-Agente per la Gestione delle Emergenze Urbane

Alessio Erasmo, Christian Felicione

10 febbraio 2026

1 Introduzione

Il *CityEmergencyManagementSystem* è un sistema multi-agente sviluppato in DALI secondo la metodologia GAIA per la gestione coordinata di emergenze urbane quali incendi e incidenti. Il sistema coordina automaticamente unità di soccorso attraverso quattro ruoli principali: Dispatcher, DroneScout, AmbulanceUnit e FireRescueUnit.

2 Architettura

Il Dispatcher coordina centralmente il sistema ricevendo segnalazioni e orchestrando le unità operative. I DroneScout sorvegliano il territorio identificando emergenze autonomamente. Le AmbulanceUnit gestiscono il soccorso sanitario mentre le FireRescueUnit intervengono sugli incendi. Gli obiettivi sono salvare vite umane, contenere situazioni di pericolo e garantire risposte coordinate prevenendo escalation.

3 Dinamica Operativa

Le emergenze raggiungono il Dispatcher tramite chiamate esterne o rilevazioni automatiche dei droni. Quando un drone avvista un'emergenza, ne comunica immediatamente localizzazione e tipologia. Se mancano informazioni, il Dispatcher richiede una ricognizione specifica.

Una volta classificata l'emergenza, il Dispatcher assegna le risorse appropriate: AmbulanceUnit per emergenze sanitarie, FireRescueUnit per incendi. Le unità possono accettare confermando la presa in carico o rifiutare specificando il motivo. Al termine dell'intervento comunicano la conclusione delle operazioni, liberando le risorse.

4 Caratterizzazione degli Agenti

Agente	**Comportamento**
Dispatcher	Reattivo: orchestra operazioni senza iniziativa autonoma.
DroneScout	Reattivo a richieste; proattivo nel pattugliamento e gestione batteria.
AmbulanceUnit	Reattivo a dispatch; proattivo nel richiedere supporto antincendio.
FireRescueUnit	Reattivo a dispatch; proattivo nel richiedere supporto sanitario.

Tabella 1: Comportamenti degli agenti

5 Eventi e Azioni

Il sistema opera secondo un'architettura event-driven. Il Dispatcher gestisce `call_emergency` e `report_emergency`. I droni monitorano `spot_fire` e `spot_accident` oltre al livello di batteria. Le unità operative rispondono a eventi di dispatch gestendo il proprio stato tramite eventi interni come `accept_dispatch`, `refuse_dispatch` e `do_rescue`.

Le azioni specifiche includono `rescue_people` per le ambulanze e `turn_off_fire` per le unità antincendio. Entrambe notificano il completamento tramite `report(emergency_retired)`.

6 Sequence Diagram

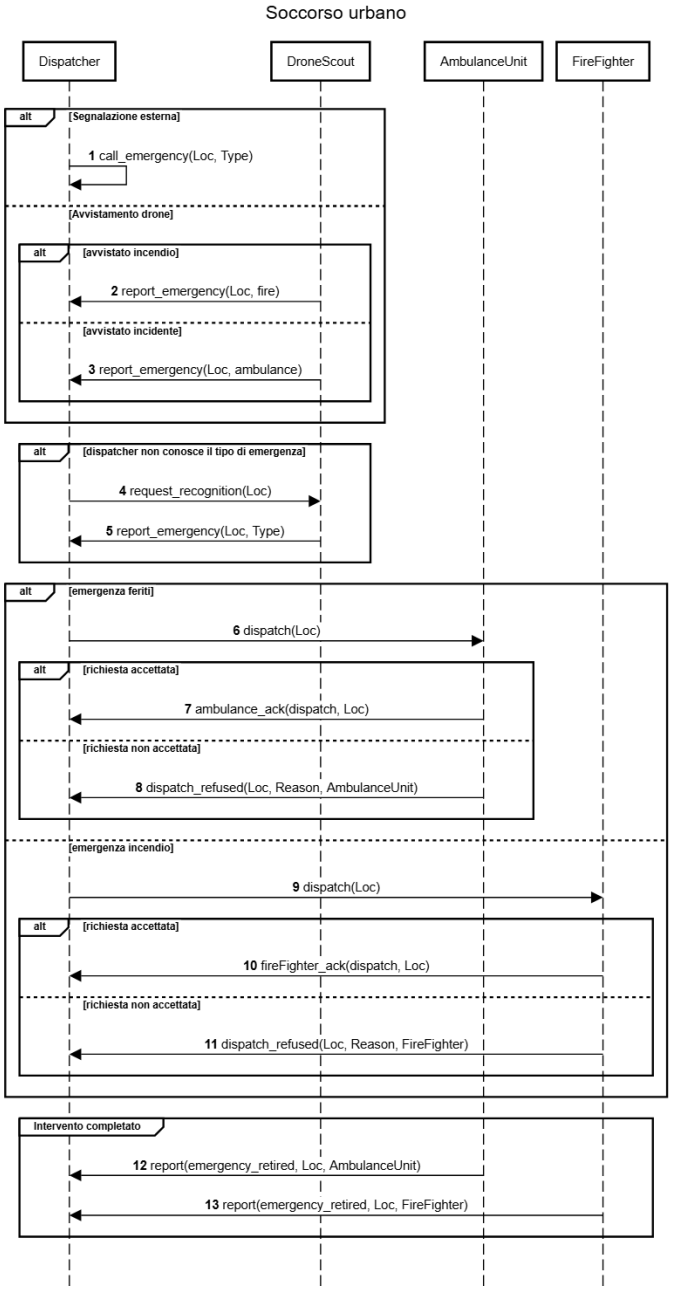

Figura 1: Sequence diagram per la gestione di un'emergenza

7 Class Diagram

Di seguito è riportato il class diagram del sistema multi-agente, che illustra le classi principali, i loro attributi e metodi, nonché le relazioni tra di esse. Il diagramma evidenzia la struttura modulare del sistema, con una chiara separazione dei ruoli e delle responsabilità di ciascun agente.

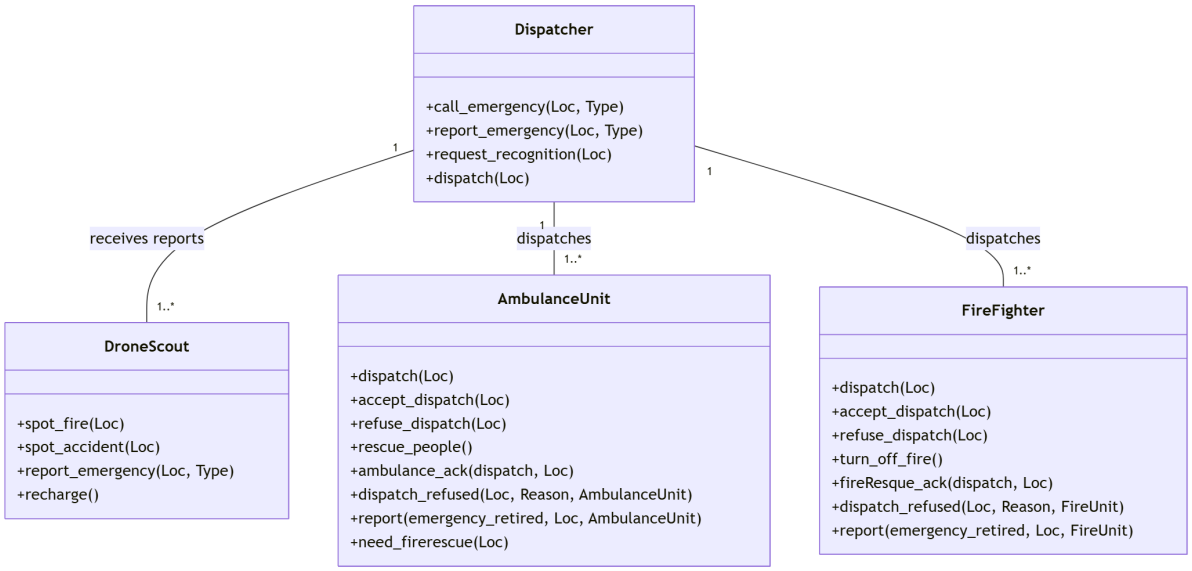

Figura 2: Class diagram del sistema multi-agente

Ogni classe, eccetto il dispatcher, dispone di più istanze, rappresentando le diverse unità operative e droni presenti nella città. Il dispatcher funge da coordinatore centrale, gestendo le comunicazioni e le assegnazioni di compiti tra le varie unità.

8 Esempio di Esecuzione

Dopo l'avvio del sistema, il Dispatcher si attiva e invia un messaggio di `activate` al DroneScout. Il drone inizia a pattugliare l'area, monitorando costantemente per emergenze. Se il drone rileva un'emergenza, invia immediatamente un messaggio al Dispatcher con la posizione. Il Dispatcher quindi classifica l'emergenza e invia un messaggio di `dispatch` all'opportuna unità, che se libera accetta l'incarico e si dirige verso il luogo selezionato per intervenire.

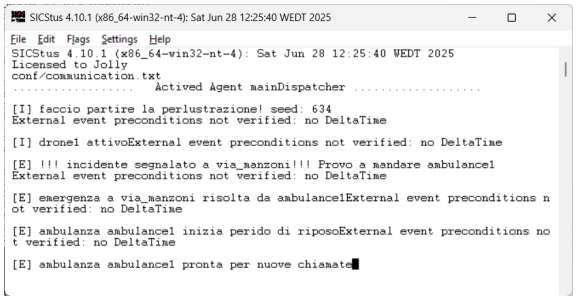

Figura 3: Ciclo di controllo del dispatcher

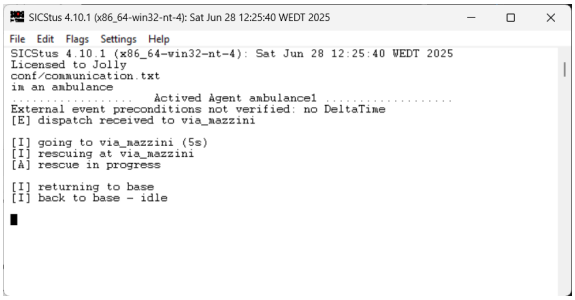

Figura 4: Gestione dell'emergenza da parte dell'ambulanza

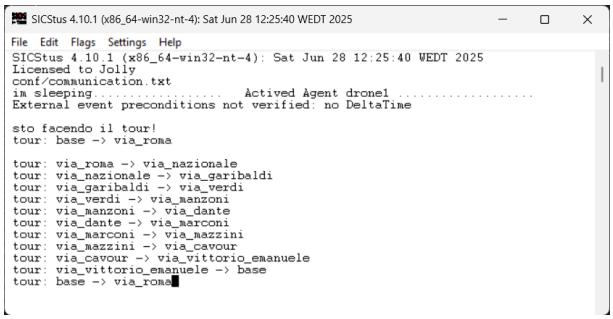

Figura 5: Drone in pattuglia

9 Messaggi di Test Manuali

Il sistema DALI permette di testare le interazioni tra agenti inviando messaggi manuali tramite l'interfaccia "New message". Questa funzionalità è particolarmente utile per verificare il comportamento del sistema in scenari controllati.

9.1 Trigger di Emergenze Esterne

Per simulare una chiamata di emergenza al Dispatcher:

Emergenza sanitaria in Via Roma:

```
To: mainDispatcher
From: user
Message: send_message(call_emergency(via_roma, ambulance), user).
```

Emergenza incendio in Via Nazionale:

```
To: mainDispatcher
From: user
Message: send_message(call_emergency(via_nazionale, fire), user).
```

Emergenza generica in Via Garibaldi:

```
To: mainDispatcher
From: user
Message: send_message(call_emergency(via_garibaldi, null), user).
```

9.2 Simulazione di Eventi Rilevati dai Droni

Per testare la capacità dei droni di rilevare emergenze autonomamente:

Rilevamento incendio:

```
To: drone1
From: environment
Message: send_message(spot_fire, environment).
```

Rilevamento incidente:

```
To: drone1
From: environment
Message: send_message(spot_accident, environment).
```

10 Installazione

Il sistema richiede **SICStus Prolog 4.x** e i file del progetto DALI MAS. Le procedure di installazione differiscono tra Linux e Windows.

10.1 Requisiti Comuni

- **SICStus Prolog 4.x** (licenza di valutazione sufficiente)
- File del progetto **DALI MAS**
- **tmux** (solo Linux, richiesto dagli script di avvio)

10.2 Installazione su Linux

1. Installazione di SICStus Prolog Scaricare SICStus dal sito ufficiale e richiedere una **licenza di valutazione**.

2. Installazione di tmux Su Debian/Ubuntu:

```
sudo apt update
sudo apt install -y tmux
```

Per distribuzioni Fedora/RHEL:

```
sudo dnf install -y tmux
```

3. Clonazione dei repository Clonare entrambi i repository necessari:

```
git clone https://github.com/AAAI-DISIM-UnivAQ/DALI.git
git clone https://github.com/christianfe/DALI-Exam-2026.git
```

4. Copia della cartella `src/` di DALI nel progetto Dalla cartella genitore che contiene entrambi i repository:

```
cp -r DALI/src DALI-Exam-2026/src
```

5. Configurazione del PATH Verificare se `sicstus` è già nel PATH:

```
which sicstus
```

Se non presente, cercare il percorso:

```
sudo find /usr/local -maxdepth 4 -type f -name sicstus 2>/dev/null
sudo find /opt -maxdepth 5 -type f -name sicstus 2>/dev/null
```

Aggiungere SICStus al PATH modificando `~/.bashrc`:

```
echo 'export PATH="$PATH:/usr/local/sicstus4.10.1/bin"' >> ~/.bashrc
source ~/.bashrc
```

6. Verifica dell'installazione Verificare che tmux e SICStus siano correttamente installati:

```
tmux -V
```

```
sicstus --version
```

7. Avvio del sistema Spostarsi nella cartella del progetto e avviare il MAS:

```
cd ~/DALI-Exam-2026
chmod +x startmas.sh
./startmas.sh
```

10.3 Installazione su Windows

1. Installazione di SICStus Prolog Scaricare e installare SICStus dal sito ufficiale, quindi richiedere e attivare una **licenza di valutazione**.

2. Clonazione dei repository Aprire il **Prompt dei comandi** (o PowerShell) e posizionarsi nella cartella desiderata (ad esempio `C:\projects`):

```
git clone https://github.com/AAAI-DISIM-UnivAQ/DALI.git
git clone https://github.com/christianfe/DALI-Exam-2026.git
```

3. Copia della cartella `src\` di DALI nel progetto .

```
rmdir /S /Q DALI-Exam-2026\src
xcopy /E /I DALI\src DALI-Exam-2026\src
```

4. Configurazione di startmas.bat Modificare il file `startmas.bat` indicando il percorso della cartella `bin` di SICStus:

```
set sicstus_home=C:\Program Files\SICStus Prolog VC16 4.10.1\bin
```

Verificare che la directory esista e contenga `sicstus.exe`.

5. Avvio del sistema Eseguire `startmas.bat` con doppio clic o da `cmd.exe` per avviare il MAS.

11 Conclusioni

L'architettura modulare garantisce scalabilità e manutenibilità. La combinazione di comportamenti reattivi e proattivi consente risposte flessibili a scenari complessi, mentre il coordinamento centralizzato assicura coerenza globale. L'implementazione in DALI sfrutta le capacità del linguaggio per eventi, comunicazione inter-agente e ragionamento ibrido, traducendo direttamente i costrutti della metodologia GAIA.