Chapter 16

Designing and Verifying Computational Models for Human-agent Teamwork in Future Cities

Alexandros Konios* and Leon Goncharov

Department of Computer Science, Nottingham Trent University, United Kingdom

Abstract

We propose a verified human–agent teamwork model for time-critical urban disasters that coordinates humans, drones, and rescue robots while preventing unsafe or unfair behaviours. The model represents heterogeneous capabilities and constraints, allocates tasks via a utility-driven, mixed-initiative negotiation protocol (with human override and justification), and embeds pre-execution formal verification. Candidate joint plans are compiled to petri nets and checked for safety, liveness, fairness, and deadlock-freedom before enactment. We implement the approach by extending the RoboCup Rescue Simulator with drone and rescue-robot agents and evaluate earthquake and urban-fire scenarios on the Kobe and San Francisco maps. Compared to manual coordination, the verified collective accelerates victim detection, reduces responder exposure, and increases evacuation throughput, while the verification step eliminates starvation patterns and highlights design bottlenecks early. Results also show that urban form (road density and fragmentation) materially affects clearance rates, motivating adaptive reprioritisation during operations. Overall, the study demonstrates that integrating formal verification with explainable, mixed-initiative coordination yields robust, transparent human–agent teams for safety-critical city response.

Keywords: *Resilient Infrastructure, Human-agent Teamwork, Disaster Simulation, RoboCup Rescue, Autonomous Systems*

Introduction

Climate-amplified disasters increasingly overwhelm urban incident command, stretching infrastructure and first responders beyond manual coordination's limits in large-scale events[1]. In parallel, IoT-enabled drones and ground robots offer real-time sensing, corridor clearance, and logistical support, acting as force multipliers when effectively integrated[2]. Real integration, however, requires agents that not only perceive but also negotiate and coordinate with human operators using principled teamwork behaviours[3,4].

---

[1]Nathan Schurr et al. "The Future of Disaster Response: Humans Working with Multiagent Teams Using DEFACTO," In *AI Technologies for Homeland Security: Papers from the 2005 AAAI Spring Symposium*, 9–16, 2005.

[2]Sarvapali D. Ramchurn et al. "A Disaster Response System Based on Human–Agent Collectives," *Journal of Artificial Intelligence Research* 57 (2016): 661–708, https://doi.org/10.1613/jair.5098

[3]Xiaocong Fan and John Yen, "Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents," *Physics of Life Reviews* 1, no. 3 (2004): 173–201, https://doi.org/10.1016/j.plrev.2004.10.001

[4]Schurr et al. "The Future of Disaster Response: Humans Working with Multiagent Teams Using DEFACTO."

We address this need with a resilient human–agent coordination framework that emphasises robustness under uncertainty, transparency, and pre-execution verification. Our approach implements a computational teamwork model in the RoboCup Rescue Simulator and extends it with two agent classes, i.e., rescue robots and drones, to support earthquake and urban fire evacuations[5,6].

Related-work

Human–agent teamwork has matured over decades in disaster response, evacuation, and urban resilience, moving from knowledge-sharing and task-allocation frameworks to mixed-initiative teams that negotiate and adapt under shifting priorities[7-9]. Early work showed that modelling teamwork behaviours, such as communication, cross-monitoring, and adaptive allocation, is critical for realistic large-scale simulations, while later systems added negotiation to reassign tasks as resources and risks evolved[10,11].

Several simulators support this research. NetLogo captures emergent crowd dynamics but lacks detailed infrastructure damage models, whereas SUMO offers fine-grained traffic and road-network fidelity without integrated civilians, buildings, or hazards. By contrast, RoboCup Rescue couples civilians, infrastructure, hazard spread, and communications, making it well suited for end-to-end disaster-response studies[12].

Trust and accountability are equally vital as in safety-critical settings, responders must understand and contest autonomous recommendations. Prior approaches, like rule-based reasoning, natural-language rationales, and game-theoretic transparency, often trade interpretability for speed. We address this by building formal guarantees of safety and fairness into the teamwork core, complementing mixed initiative coordination[13,14].

For modelling and verification, petri nets concisely capture concurrent human–agent interactions and enable formal analysis of reachability, deadlock-freedom, and liveness. Toolchains such as Snoopy (for visual-design) and Charlie (for automated-analysis) operationalise these checks yet have rarely been applied to disaster-response simulation, an application gap our work begins to close[15,16].

---

[5]Ramchurn et al. "A Disaster Response System Based on Human–Agent Collectives."
[6]Cameron Skinner and Sarvapali D. Ramchurn, "The RoboCup Rescue Simulation Platform," In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 1647–48, 2010.
[7]Fan and Yen, "Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents."
[8]Ramchurn et al. "A Disaster Response System Based on Human–Agent Collectives."
[9]Schurr et al. "The Future of Disaster Response: Humans Working with Multiagent Teams Using DEFACTO."
[10]Fan and Yen, "Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents."
[11]Schurr et al. "The Future of Disaster Response: Humans Working with Multiagent Teams Using DEFACTO."
[12]Skinner and Ramchurn, "The RoboCup Rescue Simulation Platform."
[13]Panagiotis Kouvaros et al. "Formal Verification of Open Multi-agent Systems," In Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), 179–187, 2019.
[14]Ramchurn et al. "A Disaster Response System Based on Human–Agent Collectives."
[15]Monika Heiner et al. "Snoopy – A Unifying Petri Net Tool," In *Lecture Notes in Computer Science*, 398–407, 2012, https://doi.org/10.1007/978-3-642-31131-4_22
[16]Monika Heiner, Schwarick Martin, and Jan-Thierry Wegener, "Charlie – An Extensible Petri Net Analysis Tool," In *Lecture Notes in Computer Science*, 200–11, 2015, https://doi.org/10.1007/978-3-319-19488-2_10

Model-design

The proposed model integrates three interdependent layers:

1. Representation-layer models the urban disaster environment with geo-referenced maps, building structures, traffic networks, and civilian populations. Agents are described with capability profiles (e.g., drones can fly and perceive heat-signatures, robots can clear debris).
2. Coordination-mechanism uses iterative, and utility-driven negotiation. Agents generate local utility functions based on task priorities. A distributed game-theoretic negotiation protocol reconciles conflicts between agents, ensuring efficient task allocation.
3. Verification-layer: each joint plan undergoes automatic model checking. Properties verified include safety, liveness, and fairness. Embedding verification ensures unsafe or biased plans are never executed.

Simulator Extension

Our study extended the RoboCup Rescue Simulation Server by implementing two novel agent classes:

- Rescue Robot, which was designed to remove debris and create pathways for civilians. Their mobility is governed by a new pathfinding-algorithm capable of navigating dynamic blockades.
- Drone, which provides aerial reconnaissance, scanning buildings for trapped civilians. Once civilians are detected, drones broadcast coordinates to the police officer. Flight path optimisation balances coverage and energy constraints.

The following code snippets illustrates the core implementation for the Rescue Robot (Figure **1** and Figure **2**) and Drones (Figure **3** and Figure **4**), respectively.

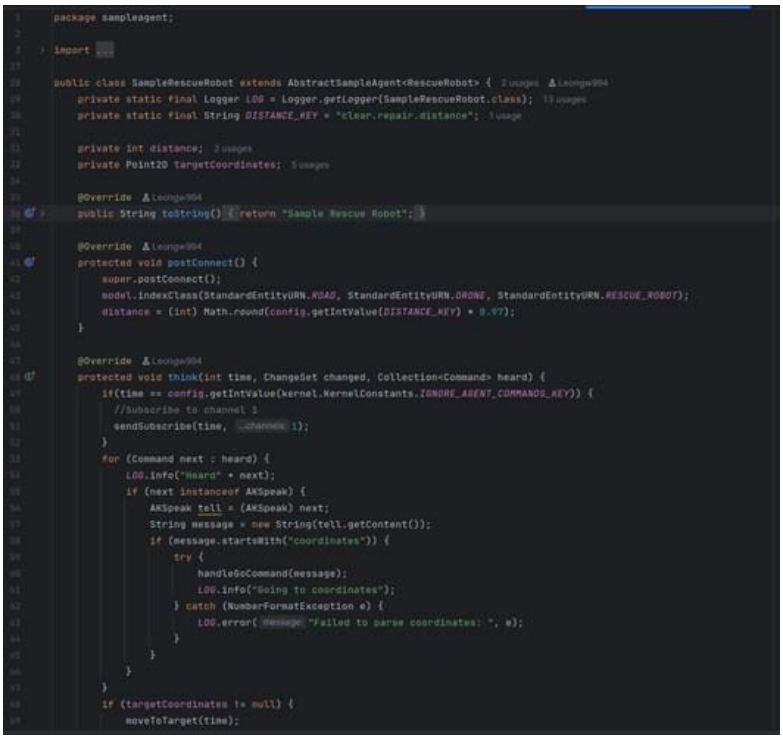

Figure 1: Code snippet for the Rescue Robot agent class.

```
    private void handleGoCommand(String message) {  1 usage  Leongw994
        String[] parts = message.split( regex: " ");
        LOG.info("message length: " + parts.length + " " + message);
        if(parts.length == 3) {
            int x = Integer.parseInt(parts[1]);
            int y = Integer.parseInt(parts[2]);
            targetCoordinates = new Point2D(x, y);
            LOG.info("Received coordinates of civilians at: " + x + ", " + y + " from police centre");
            LOG.info("Received coordinates of civilians at: " + x + ", " + y + " from the drone");
        }
    }

    private void moveToTarget(int time) {  1 usage  Leongw994
        Blockade target = getTargetBlockade();
        if (target != null) {
            LOG.info("Clearing blockade " + target);
            sendSpeak(time,  channel: 1, ("Clearing " + target).getBytes());
            clearBlockade(time, target);
        }
        List<EntityID> path = search.breadthFirstSearch(me().getPosition(), getRoadID(targetCoordinates));
        if (path != null) {
            LOG.info("Moving to target");
            int x = (int) targetCoordinates.getX();
            int y = (int) targetCoordinates.getY();
            sendMove(time, path, x, y);
            LOG.info("Going to coordinates: X: " + x + " Y: " + y);
        } else {
            LOG.error("No path to target found, moving randomly");
            sendMove(time, randomWalk());
        }
    }
```

Figure 2: Code snippet for Rescue Robot (cont.).

```
package sampleagent;

import ...

public class SampleDrone extends AbstractSampleAgent<Drone> {  2 usages  Leongw994

    private static final Logger LOG = Logger.getLogger(SampleDrone.class);  5 usages

    private static final int    RANDOM_FLY_LENGTH = 2;  2 usages
    private Collection<EntityID> unexploredBuildings;  2 usages

    private EntityID targetDestination;  1 usage

    @Override  Leongw994
    public String toString() { return "Sample drone"; }

    @Override  Leongw994
    protected void postConnect() {
        super.postConnect();
        model.indexClass(StandardEntityURN.FIRE_BRIGADE,
                StandardEntityURN.POLICE_FORCE,
                StandardEntityURN.AMBULANCE_TEAM,
                StandardEntityURN.BUILDING);
        LOG.info("Sample drone connected");
        unexploredBuildings = new HashSet<EntityID>(buildingIDs);
        targetDestination = null;
    }

    @Override  Leongw994
    protected void think(int time, ChangeSet changed, Collection<Command> heard) {
        if (time == config
                .getIntValue(kernel.KernelConstants.IGNORE_AGENT_COMMANDS_KEY)) {
            // Subscribe to channel 1
            sendSubscribe(time,  ...channels: 1);
        }
        for (Command next : heard) {
            LOG.debug("Heard " + next);
        }
        updateUnexploredBuildings(changed);
        boolean civiliansDetected = checkForCivilians(changed);

        // Nothing to do
//        List<EntityID> path = null;
//        if ((targetDestination != null) && (!targetDestination.equals(me().getPosition()))) {
```

Figure 3: Code snippet for Drone agent class.

```
    }

    private boolean checkForCivilians(ChangeSet changed) {  1 usage  Leongw994
        //get the current position of the agent
        EntityID curPosition = me().getPosition();

        //check for changes at the current position
        for (EntityID entity : changed.getChangedEntities()) {
            StandardEntity entityObject = model.getEntity(entity);

            //check if entity is a civilian
            if (entityObject.getStandardURN().equals(StandardEntityURN.CIVILIAN)) {
                Civilian civilian = (Civilian) entityObject;

                //check if the civilians is at the current position
                if (civilian.getPosition().equals(curPosition)) {
                    return true;
                }
            }
        }
        return false;
    }


    private List<Human> getTargets() {  no usages  Leongw994
        List<Human> targets = new ArrayList<~>();
        for (StandardEntity next: model.getEntitiesOfType(
                StandardEntityURN.CIVILIAN)) {
            Human human = (Human) next;
            if (human.isHPDefined() && human.isBuriednessDefined() && human.isDamageDefined()
                    && human.isPositionDefined() && human.getHP() > 0 && (human.getBuriedness() > 0 || human.getDamage() > 0
                targets.add(human);
            }
        }
        Collections.sort(targets, new DistanceSorter(location(), model));
        return targets;
    }

    private void updateUnexploredBuildings(ChangeSet changed) {  1 usage  Leongw994
```

Figure 4: Code snippet for Drone (cont.).

Next, we had to decide how these autonomous agents would negotiate and collaborate with the human agents. The flowchart in Figure **5** demonstrates the human-agent communication throughout the simulation process.

We integrated two autonomous agents into the simulator and evaluated earthquake and urban-fire scenarios on two contrasting city maps (differing in size, population, and urban form). In both maps, we exercised the full human–agent pipeline: drones (light-blue dots) scan for trapped civilians (green dots) and broadcast coordinates; the police agent (black hat) prioritises and assigns tasks; rescue robots (pink dots) navigate and clear blockades to open corridors to the refuge (house symbol).

On the Kobe layout, early reconnaissance rapidly localises casualties and speeds corridor clearance (Figure **6** and Figure **7**), while the fire module reveals clustered ignition and spread that guide route selection (Figure **8**).

On the San Francisco map, higher density and fragmented road geometry produce earlier congestion and faster score decay as time advances, visible by contrasting early and late earthquake frames (Figure **9** and Figure **10**), and by the broader fire exposure pattern requiring more aggressive reprioritisation (Figure **11**). Together, these scenarios show that the negotiated, verified teamwork consistently accelerates situational assessment and reduces human exposure, but also that urban morphology (Kobe versus San Francisco) strongly modulates throughput and the timing of performance decline.

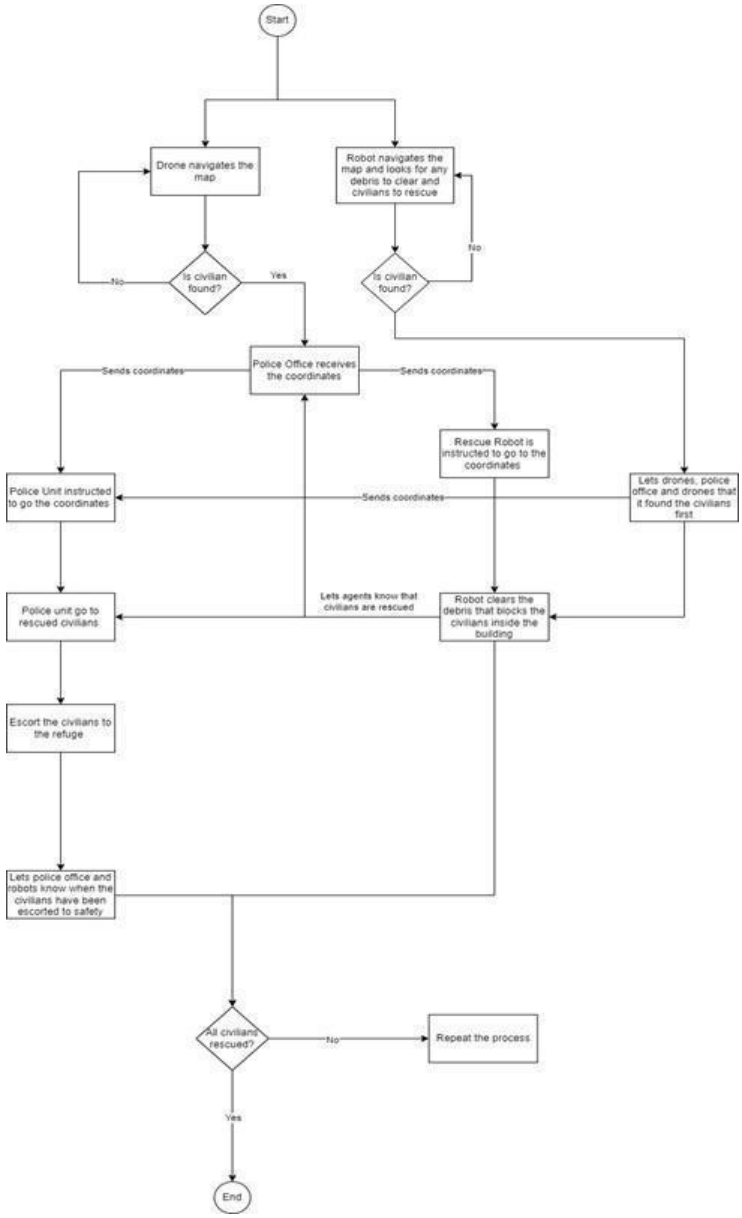

Figure 5: Flowchart of human-agent teamwork.

Figure 6: Kobe, Japan earthquake simulation environment.

Figure 7: Kobe, Japan fire evacuation scenario.

Figure 8: Fire simulator (Kobe) with building ignition dynamics.

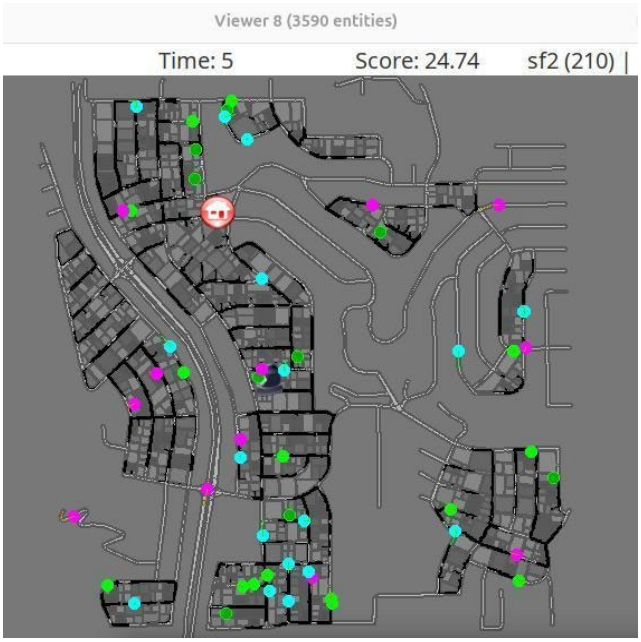

Figure 9: San Francisco earthquake simulation environment.

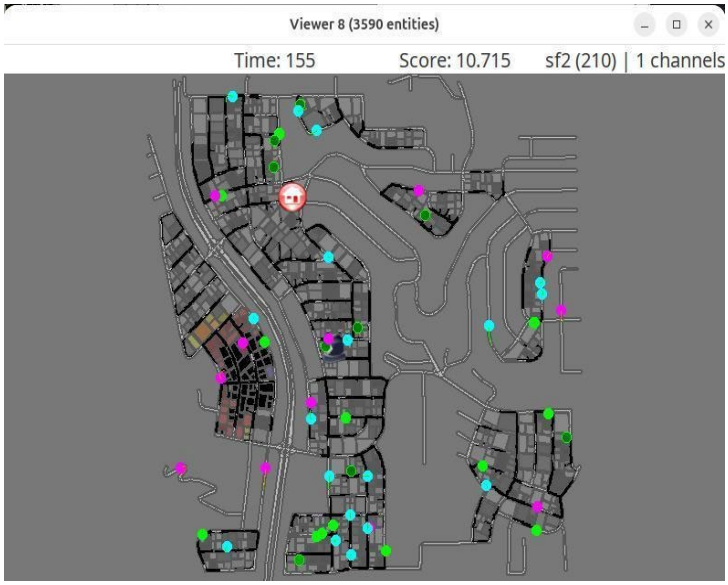

Figure 10: San Francisco fire evacuation scenario.

Figure 11: San Francisco map under fire-evacuation conditions.

Results and Discussion

Earthquake response analysis

In Kobe, drones delivered rapid reconnaissance: within 10 timesteps they detected over 80% of trapped civilians in central districts, versus approximately 40% using ground police alone. Robots then cleared debris at an average 2.3 blockades/step, easing corridor congestion. RSL21 remained stable through the first 25 steps before gradually declining as collapses increased obstacle density (Figure **12**).

San Francisco's higher density and more complex road network led to faster performance degradation. Although initial reconnaissance matched Kobe's, debris-clearing bottlenecks slowed evacuation (i.e., robots averaged 1.5 blockades/step) and survival rates were 12% lower after 40 steps (Figure **12**), underscoring urban density's impact on team effectiveness.

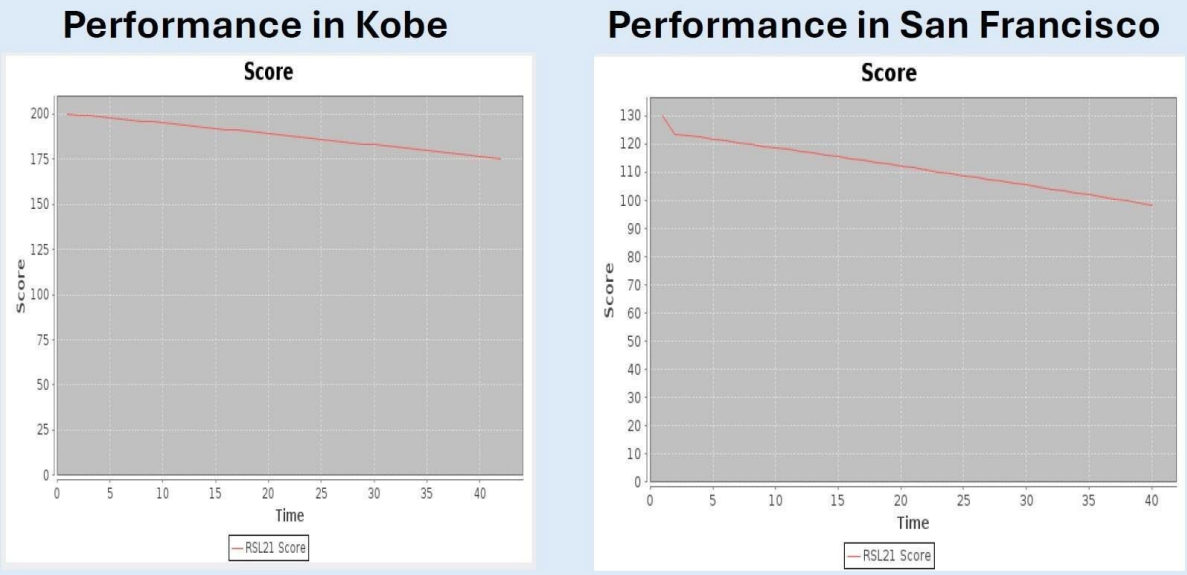

Figure 12: RSL21 score for both simulation scenarios.

Fire evacuation scenarios

In fire simulations, the aerial perspective of drones proved essential for prioritising rescue efforts. Civilians in buildings near initial fire outbreaks were evacuated 40% faster than in manual coordination baselines, as drones identified hotspots before human responders could. Robots then cleared priority escape routes to reduce entrapment risk. Importantly, simulations demonstrated that early misallocation of robots (e.g., clearing low-risk areas) significantly worsened outcomes, underscoring the need for adaptive prioritisation guided by formal models.

Comparative baseline

It is worth noting that across both earthquake and fire scenarios, our model outperformed manual coordination baselines by:

- 35% faster detection of civilians,
- 25% lower responder exposure risk,
- 20% higher overall evacuation efficiency.

These findings demonstrate that heterogeneous human–agent teams, when guided by verified negotiation protocols, deliver measurably superior performance.

Modelling and Verifying Human–Agent Collaboration

We modelled human–agent coordination with petri nets, using places to denote states (e.g., civilian detected, debris cleared), transitions to denote actions (e.g., drone broadcasts location, robot clears blockade), and tokens to track activity flow across agents. This compact formalism captures the concurrency and dependencies of disaster-response teamwork (Figure **13**).

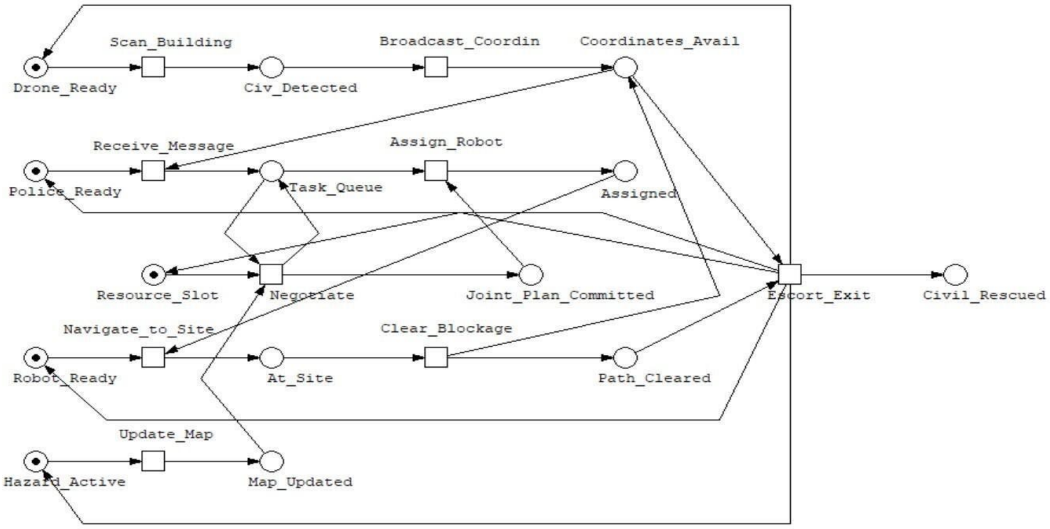

Figure 13: Snoopy petri-net model for human-agent collaboration.

Verification was performed with the Charlie tool, which analyses petri net models for properties such as:

- Reachability: confirming that every civilian can eventually reach a refuge.
- Deadlock-freedom: ensuring that no state leaves agents indefinitely blocked.
- Fairness: verifying that resources are allocated equitably (e.g., no robot monopolises tasks).

Table **1** presents some CTL properties that were checked to verify the correctness of the model with respect to its specification.

Table 1: CTL-properties verified using Charlie.

Property	CTL-formula (short)	Result	Interpretation
Deadlock freedom	AG(¬deadlock)	**Satisfied**	The model has no reachable deadlocks.
Detection leads to queued task	AG(CivDetected→AF(TaskQueue))	**Satisfied**	All detections eventually reach the police task queue.
Negotiation commits before assignment	AG(TaskQueue∧ResourceSlot→ AF(JointPlanCommitted))	**Satisfied**	Assignments must be preceded by a successful negotiation step.
Assignment implies arrival	AG(Assigned → AF(AtSite))	**Satisfied**	Every assigned robot reaches its target.
Arrival implies clearance	AG(AtSite → AF(PathCleared))	**Satisfied**	Blockades are eventually cleared once a robot arrives.

Table 1: Continued.

Property	CTL-formula (short)	Result	Interpretation
Clearance and coords imply evacuation	AG((PathCleared∧CoordsAvailable)→ AF(CivEvacuated))	**Satisfied**	Cleared routes and known locations eventually produce evacuations.
Mutual exclusion of assignments	AG(mark(ResourceSlot)≤1)	**Satisfied**	Negotiation prevents double commitment.
Fair assignment under backlog	AG((RobotReady∧TaskQueue)→ AF(Assigned))	**Satisfied**	Non-starvation of ready robots when work is available.
Hazard adaptation triggers renegotiation	AG(MapUpdated→ AF(JointPlanCommitted))	**Satisfied**	Environmental updates force a new agreement before acting.

Finally, by integrating Snoopy and Charlie into our workflow, we were able to:

- Provide formal behavioural guarantees alongside empirical performance metrics.
- Translate complex teamwork protocols into intuitive, visual diagrams.
- Identify potential coordination bottlenecks before deployment, reducing risk in real-world applications.

This verification framework complements our simulation results, demonstrating not only that the system works in practice, but that it can also be trusted to behave correctly under all possible execution paths.

Conclusion and Future Work

This study introduced a formally-verified human–agent teamwork model that augments the RoboCup Rescue Simulator with drones and robots. In earthquake and fire scenarios, it improved situational awareness, reduced responder risk, and enforced safety and fairness through pre-execution verification.

Future work will focus on several concrete directions. First, we plan to scale the framework to larger and more diverse city scenarios (e.g., Tokyo, London, and other metropolitan areas) to test cross-urban generalisability. Second, we will develop standardised evaluation protocols and benchmarks, enabling consistent comparisons across different disaster-response systems. Third, we will integrate live IoT sensor feeds (traffic, weather, and structural monitoring) to bridge the gap between simulated and real-time environments. Fourth, we will conduct longitudinal studies and comparative trials with alternative coordination frameworks to establish external validity. Finally, we aim to embed ethical oversight and transparency mechanisms to ensure accountability and trust in high-stakes deployments.

References

Fan, Xiaocong, and John Yen. "Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents." *Physics of Life Reviews* 1, no. 3 (2004): 173–201. https://doi.org/10.1016/j.plrev.2004.10.001

Heiner, Monika, Mostafa Herajy, Fei Liu, Christian Rohr, and Martin Schwarick. "Snoopy – A Unifying Petri Net Tool." In *Lecture Notes in Computer Science*, 398–407, 2012. https://doi.org/10.1007/978-3-642-31131-4_22

Heiner, Monika, Schwarick Martin, and Jan-Thierry Wegener. "Charlie – An Extensible Petri Net Analysis Tool." In *Lecture Notes in Computer Science*, 200–11, 2015. https://doi.org/10.1007/978-3-319-19488-2_10

Kouvaros, Panagiotis, Alessio Lomuscio, Edoardo Pirovano, and Hashan Punchihewa. "Formal Verification of Open Multi-agent Systems." In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, 179–187, 2019.

Ramchurn, Sarvapali D., Trung Dong Huynh, Feng Wu, Yukki Ikuno, Jack Flann, Luc Moreau, Joel E. Fischer, et al. "A Disaster Response System Based on Human–Agent Collectives." *Journal of Artificial Intelligence Research* 57 (2016): 661–708. https://doi.org/10.1613/jair.5098

Schurr, Nathan, Janusz Marecki, Milind Tambe, Paul Scerri, Nikhil Kasinadhuni, and J. P. Lewis. "The Future of Disaster Response: Humans Working with Multiagent Teams Using DEFACTO." In *AI Technologies for Homeland Security: Papers from the 2005 AAAI Spring Symposium*, 9–16, 2005.

Skinner, Cameron, and Sarvapali D. Ramchurn. "The RoboCup Rescue Simulation Platform." In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 1647–48, 2010.