An interaction-centric approach to support peer coordination in distributed emergency response management

Maurizio Marchese | Lorenzino Vaccari | Gaia Trecarichi
---|---|---
University of Trento | University of Trento | University of Trento
maurizio.marchese@unitn.it | vaccari@disi.unitn.it | gtrecari@disi.unitn.it
Nardine Osman | Fiona McNeill | Paolo Besana
University of Edinburgh | University of Edinburgh | University of Edinburgh
N.Osman@sms.ed.ac.uk | f.j.mcneill@ed.ac.uk | p.besana@sms.ed.ac.uk

ABSTRACT

Modern information systems are required to operate in distributed and dynamic environments. In such open settings, coordination technologies play a crucial role in the design of flexible software systems. Research efforts in different areas are converging to devise suitable mechanisms for process and peer coordination: in particular, current results on service-oriented computing and multi-agent systems are being integrated to support dynamic decision-making processes among autonomous components in large, open systems. This paper addresses how agent technologies can be designed, applied, and eventually integrated with standard technologies, in order to build more robust and intelligent systems. The focus of our research is on the engineering, exploitation and evaluation of an agent protocol language in realistic contexts. In particular, a specific executable protocol language is adopted to specify simulated interactions among distributed processes which will be tested in emergency response domain activities (that we will refer to hereafter as e-Response activities), chosen as an example of knowledge-intensive

1 This paper is a revised and extended version of a paper presented at the 5th International ISCRAM Conference – Washington, DC, USA, May 2008, with the title “Interaction models to support peer coordination in crisis management”

2 Corresponding Author:
Maurizio Marchese, University of Trento, DISI
via Sommarive, 14, 38100 Trento, Italy
phone: +39 0461 882083; fax: +39 0461 882093
and dynamic application domain where intelligent decision making is crucial. We present a novel approach based on shared protocols models distributed through a peer-to-peer infrastructure and we show how it can be applied in the context of crisis management to support coalition formation and process coordination in open environments. Specifically, a prototype e-Response simulation system – built on a Peer-to-Peer (P2P) infrastructure – has been developed to execute interaction models describing common coordination tasks in the emergency response domain. Preliminary evaluation of the proposed framework demonstrates its capability to support such e-Response tasks.

Keywords

1. INTRODUCTION

One of the major challenges of modern information systems is to operate within open and dynamic environments. For instance, in a e-Response scenario, a large number of actors (e.g., local governments, emergency coordination center, fire brigade, police and health agency, volunteers, citizen) are involved in the emergency activities: people in a coordination center may need to retrieve and integrate data and services from different geographical information systems in order to plan activities, make decisions and forward directives to their subordinates; geographically dispersed agents may have to collaborate and coordinate in the disaster scenes by exchanging information with each other and reporting to the people in the control room; moreover, the information distribution system has to cope with unexpected situations (e.g., actors changing roles, taking on new and interrupting old activities).

The scenario above aims to outline some of the key elements needed in e-Response situations. Emergency management activities are developed and implemented through the essential analysis of information and
the coordination of the involved peers (both institutional, like emergency personnel, army, volunteers, etc as well as ordinary people involved in the crisis). The existence of numerous and different actors, policies, processes, data standards and systems results in coordination problems with respect to data analysis, information delivery and resources management, all critical elements of emergency response management.

In our current research, we want to explore the flexibility and adaptability of an interaction-driven mechanism for knowledge sharing that relies on a distributed – Peer-to-Peer – infrastructure. At the core of the approach is a specific view of the semantics of both service and agent coordination (as proposed in[15]), where peers share explicit knowledge of the “interactions” in which they are engaged and these models of interaction are used operationally as the anchor for describing the semantics of the interaction. Instead of requiring a universal semantics across peers, the approach requires only that semantics are consistent (separately) for each instance of an interaction. This is analogous to the use in human affairs of contracts, which are devices for standardising and sharing just those aspects of semantics necessary for the integrity of specific interactions. In this paper, we apply the above approach and related prototypical infrastructure – referred to hereafter as the OpenKnowledge kernel [3,15,18] – in the context of crisis management. The nature of the e-Response domain is such that process-aware systems are beneficial to prevent chaotic and uncontrolled conditions. Nevertheless, taking into account adaptability is fundamental to handle unexpected situations (i.e. sudden road blockage, fast and unpredicted events, etc.) which are likely to occur in emergency situations. While the general vision of interaction protocols accounts for the “structured coordination” requirement of the problem, the adoption of models specifically designed to explicit interactions in a P2P fashion and passed through an underling open infrastructure accounts for the support for flexibility and dynamicity. The main contributions of the paper are:

- the usage of a protocol language that, expressing P2P style interactions and relying on a distributed infrastructure, provides a mechanism for coalition formation, peer coordination and Web service composition [3,15,16];
• the provision of a simulation environment in which to evaluate how suitable the proposed interaction
models and distributed infrastructure are in coordinating peers in real time;
• the evaluation of the proposed approach to support typical emergency response tasks, requiring both
structured and dynamic coordination.

In what follows, we first introduce the basic elements of the OpenKnowledge framework, that is, the
protocol language used to express interactions among peers and the core software component which
makes these interactions executable. We then present the emergency response case study we have been
working on, specifically, the evacuation management plan during a flooding event in the Trentino region
in Italy. Next, we describe the e-Response system which is built on top of the OpenKnowledge
infrastructure and its main components, namely, the e-Response simulator and the peer network. In the
following sections we give some detailed examples of the interaction models used in the use-case
simulation and we then briefly present a preliminary prototype of the overall system with which we assess
the simulation of initial emergency response coordination activities. In the final sections, we discuss
related and future work.

2. THE OPENKNOWLEDGE FRAMEWORK

The European project OpenKnowledge\(^3\) has provided the operational framework for running the
experiments in our work. The core concepts in OpenKnowledge are (1) the interactions between peers,
defined by interaction models written in the Lightweight Coordination Calculus (LCC) and published by
the authors on a Distributed Discovery Service (DDS) with a keyword-based description; and (2) a
distributed infrastructure that supports the publishing, discovery, execution, monitoring and management
of the various interaction models.

\(^3\) OpenKnowledge Project: [www.openk.org/](http://www.openk.org/)
In what follows we first describe the main characteristics of the LCC protocol language and we then sketch the basic ideas behind the sophisticated infrastructure (the OpenKnowledge kernel) which allows the running and management of coordinated tasks between participating peers.

2.1. Introduction to LCC

LCC is a protocol language used to describe interactions among distributed processes, e.g., agents and web services [13,14]. LCC was designed specifically for expressing P2P style interactions within multi-agent systems, i.e., without any central control; therefore, it is well suited to modeling coordination of software components running in an open environment. Its main characteristics are flexibility, modularity and the neutrality to the distributed communication infrastructure.

Interactions in LCC are expressed as the message passing behaviors associated with roles. The most basic behaviours are to send or receive messages, where sending a message may be conditional on satisfying a constraint (precondition) and receiving a message may imply constraints (postcondition) on the agent accepting it. As an example, a basic LCC interaction is shown in Fig. 1 where double and single arrows indicate respectively message passing and constraints to satisfy.

According to the interaction shown in Fig. 1, the agent A1 playing the role r1 verifies if it needs the information X (constraint need(X)); if so, A1 asks for X from the agent A2 playing the role r2 by sending the message ask(X). A2 receives the message ask(X) from A1 and then gets the information X (constraint get(X)) before sending back a reply to A1 through the message return(X). After having received the message return(X), A1 updates its knowledge (constraint update(X)).

The constraints embedded into the protocol express its semantics and could be written as first-order logic predicates (i.e., in Prolog) as well as methods in an object-oriented language (i.e., in Java). Furthermore, these constraints could hide simple functionalities (e.g., provided by web services) as well as complex
algorithms. This is the characteristic of modularity previously mentioned that allows the separation of the protocol from the agent or service engineering. While performing the protocol, peers can therefore exchange messages, satisfy constraints before (after) messages are sent (received) and jump from one role to another so that a flexible interaction mechanism is enabled whilst still following a structured policy, this being absolutely necessary for team-execution of coordinated tasks.

The LCC language has been extended in the OpenKnowledge project to include annotations: interaction models can be annotated by their authors, in particular to specify the semantic type of the variables in messages and constraints. Whilst this semantic annotation was the original motivation, in fact any element in the LCC can be annotated: for example, it is possible to mark messages and constraints for logging.

2.2. The OpenKnowledge Kernel

A peer willing to perform some task, such as reporting an event or verifying the accessibility of a road, queries the discovery service (DDS) for published interaction models with descriptions matching the task (message searchIMs in Fig. 2A). The DDS replies with a list of interaction models that the peer needs to evaluate, selecting and subscribing to the preferred one. Interactions whose constraints do not match the peer's capabilities need to be excluded, though inbuilt matching techniques mean that this match does not have to be perfect or obvious. A peer solves the constraints in the LCC using plug-in components: a component is a jar file containing a java class that exposes methods that are matched against the requested constraints. A peer can use methods from different plug-ins for satisfying the constraints of one role. The peer compares the list of interaction models received from the DSS with the available plug-in components: the matching process generates a set of adaptors, one for each interaction model. Every adaptor contains a measure of the similarity between the adapted interaction and the peer's components. The peer, using the computed similarity and possibly other measures such as the popularity of the
interaction, selects the interaction model that best fits its capabilities and its interests and subscribes to perform one of its role in the DDS (message subscribe in Fig. 2A).

When all the roles in an interaction model are subscribed by enough peers (the minimum number of peers per each role is specified in the header of the LCC models), the DDS randomly chooses a peer from the P2P network and asks if it can bootstrap and then coordinate the interaction (message startIM in Fig. 2B). If it agrees, the DDS forwards to it the interaction model with all the subscriptions. Peers advertise their coordination capabilities: peers running on small devices such as palmtops may avoid the computational workload simply by not registering as coordinators.

The roles in the interaction model received by the coordinator may be subscribed by many different peers. In many cases, fewer peers than those subscribed can play in the interaction. For example, in an emergency scenario, all the members of a fire brigade might be subscribed to a “firefighter” role but a fire chief might need to send only part of its personnel to a given location. In particular, the fire chief peer may be interested in interacting with only those peers which are close to the final destination, or may want to avoid inexperienced peers. Therefore, the coordinator asks all the peers to select the other peers they want to interact with and then creates a group of peers whose preferences are mutually compatible (messages selectPeers and selectedPeers in Fig. 2B). If the group of selected peers covers all the roles in the interaction – there might be mutual exclusions - then it can start the interaction, asking a commitment to the peers (messages askCommitment and commit in Fig. 2C).

While different implementations are possible, in the OpenKnowledge framework the coordinator runs locally the interaction, creating a local proxy for each peer. The messages are exchanged between the proxies, and the peers are contacted when constraints need to be solved (messages solveConstraint and constraintSolved in Fig. 2D). In the example interaction of Fig. 1, the peer performing role r1 will be contacted to solve constraint need().
At the end of the interaction, the coordinator sends to all the peers the result of the interaction (whether it was successful or not). Fig. 2 shows the complete lifecycle of an interaction, from the search for an interaction model matching a task description to its execution.

3. THE EMERGENCY RESPONSE CASE STUDY

The management of emergency response activities during natural disasters raises challenging issues regarding the effectiveness of task coordination and the related decision making processes, as briefly described in the introduction section. Our case study is based on the management of a particular natural disaster, namely a flooding event in the Trentino region (Italy). We describe below the specific evacuation use case considered and, in the next subsection, we give details on the simulation system developed in order to evaluate the framework previously presented in a real scenario.

3.1. Flooding evacuation use case

At 23:00 on November 4th, 1966, the river Adige, the main river of the Trentino region in Italy, broke its banks at different sites and flooded the majority of the territory of the Trentino main town, Trento. Moreover a considerable amount of oil, from housing heating systems, fuel depositories and petrol stations, mixed with the mud waters of the river. The majority of the Trento population as well as of surrounding areas were affected. Today, in 2006, the flooding of the Adige river is still the most probable emergency event in the Trentino region. In such a situation, knowledge of the state of the emergency is vital. For example: (i) knowledge of the conditions of the roads affected by the flooding event; (ii) knowledge of all public buildings (e.g. schools, offices, etc.) that are contained in the flooding area, since they are critical sites and might contain a high number of people; (iii) knowledge about the service infrastructures - such as the electricity network, the waterworks network, the pipeline network, the telecommunication. The primary goal of the municipality emergency plan is to evacuate the population (ca. 110,000 persons) effectively and rapidly from the critical area.
In our work we have focused on flood response as an interesting example of an important and potentially devastating type of emergency in which there can be enormous impact on people, property and infrastructure. Such an emergency includes a range of potential uses for geo-informatics services, links to large scale sensor grids, potential for linking to automated or semi-automated response systems, and the need for coordinating processes with many organizations and agencies. Timely decisions and properly executed processes can make an enormous impact on the final outcome.

For the analysis and simulation of the flooding crisis emergency, we have grounded our work on the actual emergency plan in the Trentino region [10,12]. We have focused on the evacuation of the people from the probable flooding areas to the refuge centers outside these areas. This specific use case assumes that the institutional emergency peers have been alerted to be prepared to face a possible flooding of the river Adige within the next six hours. In the emergency room, the emergency coordinator is present, together with the coordinators of the main different agencies involved in the evacuation plan, namely, in our case: the firefighter coordinator, the police coordinator, the medical personnel coordinator. All other institutional peers involved are located in the disaster area and they include:

- Fire fighters (~ 50): supervision of evacuation plan at meeting points and at refuge centers;
- Police officers (~ 22): control of road gates;
- Medical staff (~ 88): supervision of medical requirements;
- Bus/Ambulance Drivers (~ 44).

A number of “system” peers are also present in the scenario: they represent a possible set of digital services that support the evacuation plan, among others:

- The emergency coordination data service: it is based on the emergency database and it maintains the current status of the enacted emergency plan;
- SDI Services for map information: mainly provide gazetteer services, map services, download (feature) services as in [19];
- The weather forecast service;
- The sensor network: provides the water level of the river;
- The route service: provides routing information including updates on blocked roads.

The emergency coordinator evaluates the information collected by the peers and the system services and propagates its plans to the subordinate coordinators, who are responsible for acting on the plan by distributing plans/sub-plans to each organization or group involved in the crisis management. In our use case, the evacuation plan is enacted by the emergency coordinator, by (i) propagating the alarm to the bus drivers and assigning them to the appropriate destination; (ii) propagating the alarm to the subordinate coordinators; (iii) sending the evacuation alarm and information to the citizens; (iv) continuously monitoring the crisis information from all available sources (e.g. sensors, institutional peers, volunteers, citizens) and taking appropriate action.

### 3.2. The e-Response system

Since emergency situations are of a critical nature, it is essential that any infrastructure and related processes to assist the emergency response are fully tested and evaluated prior that situation. We have therefore developed a simulator in order to demonstrate and evaluate our approach within a simulation, so as to estimate how it could perform in a genuine emergency. The e-Response system we are developing is used in our current work:

1. to model and execute interactions between peers involved in an emergency response activity, whether individuals, institution peers, sensors, web services or others;
2. to provide feedback about the environment at appropriate moments, in a way that mirrors the real world (for example, a peer attempting to take a road will be informed that the road is blocked only when it is actually at that road, and it can then share this information with other peers through the network).
3. to visualize and analyze a simulated coordination task through a Graphical User Interface (GUI).

The developed e-Response system is composed of two major components: the e-Response simulator, the peer network (and related interaction models). Fig. 3 sketches the overall architecture of the system. The
e-Response simulator provides the disaster scene and its evolution, thus representing the “real world” within which all the actors (network peers) acts. Every peer (either simulator or network peer) has an OpenKnowledge plug-in component (the OpenKnowledge kernel) which enables it to publish (or search for) interaction specifications and be involved in a coordination task with other participants. Some of the peers in the peer network interact with both the simulator and network peers: these are the peers that ‘exist’ in the physical location and can directly influence the simulated world, such as firefighter peers; other network peers communicate among themselves and never connect to the simulator: these are peers that are not physically involved in the simulation and cannot directly affect the world, such as geographical map provider peers. In the following subsections, we give some details on the e-Response simulator and the peer network respectively, focusing on a specific sub-scenario of the overall flooding use case.

3.3. The e-Response Simulator

The simulator is composed of three peers: the controller, the disaster sub-simulator, and the visualiser (see Fig. 4). The controller is the core of the simulator: it drives the simulation cycles. The controller has one main goal: to keep track of the current state of the world. In order to achieve that, the controller needs to know what changes are happening to the world, both through the actions of peers in the world and through the changes simulated by the disaster sub-simulator, and update its state accordingly. After updating its state, it informs the peers of those changes that are relevant to them; that is, those changes that occur in their locality, which they are thus able to ‘sense’.

In what follows we particularize our description to a flood sub-simulator but the discussion applies to any kind of simulated disaster. The goal of the flood simulator is to simulate the flood. It is composed of a set of predefined equations that define how the flood evolves with time. The goal of the visualiser is to simply visualise what is happening in the flood area. This includes both the effects of the flood and the
location of peers. This information is provided to the visualiser by the controller. Simulation is then divided into cycles, which are driven by the controller. The steps of one simulation cycle are:

1. The controller shares the initial topology of the world with others simulator peers: the flood sub-simulator and the visualiser. The controller receives also information about the changes that happened to the world:
   a. It receives the flood changes from flood sub-simulator.
   b. It receives other changes from the peers in the peer network that wish to cause these changes and verifies their validity: the peers are sent failure messages if they wish perform an action that is impossible or that they are not capable of.

2. The controller updates the state of the world to reflect these changes.

3. The controller sends information about the changes that happened in the world:
   c. It sends a list of all the changes to the simulator’s visualiser;
   d. It sends changes that occurred in a peer’s vicinity to each peer in the peer network.

4. The controller updates the time step, and repeats by going back to step 1.

The flow of the interactions between the controller and the other peers is also depicted in Fig. 3. Each black arrow represents a different interaction model, which also represents the flow of information between peers. For example, the flood interaction model flood IM is used to allow the flood sub-simulator to communicate with the controller, sending it flood changes at each time step. As mentioned before, since the above architecture is modular, it is reasonably easy to add as many emergency sub-simulators (e.g., landslides, earthquake, volcanic eruption, etc.) as needed: each implemented sub-simulator would, in fact, communicate with the controller through an interaction model nearly identical to the given one and the controller would be enhanced to handle the new types of emergency data.

Note that the controller (and the simulator in general) does not interfere or help coordinate peer’s actions in the peer network. It is simply used to simulate the real world. For instance, peers cannot ask the
simulator questions about the world, such as the location of something or what the flood level is at a given point. In the real world, peers (whether humans or sensors) are able to sense certain things in their vicinity. In our simulation, this information is provided by the controller. At every time step, the controller sends sensory info for each connected peer through the connect interaction model. Also, the real world usually prevents humans (or sensors or other actors) from performing certain unrealistic actions. For example, one person may try to drive a car in a flooded road, but will fail even if he insisted. Usually, the rules of physics of the real world prevent such unrealistic actions. Again, in our simulation, the controller will verify whether certain actions are legal or not before they are performed, and if a certain action is illegal, the peer is informed of the reason of failure. This is done through the “coordinate action” interaction model (see Fig. 3). Note that any peer that either needs to perform an action (such as a policeman closing a road) or needs to receive sensory info about its vicinity (such as a sensor) needs to be connected to the simulator. We call these peers physical peers. They are represented as shaded ellipses in the peer network of Fig. 3. Of course, not all peers need to connect to the controller. Non-physical peers, such as a web service that provides information about the weather, do not need to communicate with the controller but with other peers in the peer network (such as a sensor, a human user, etc.), just as in the real world such peers would not actually be in the disaster area and could not affect it directly, but could provide services to peers that are there. Non-physical peers are represented as not shaded ellipses in the peer network of Fig. 3.

Please note that in the current system, the existence and regular functionality of communication channels among peers is assumed. More complex situations in which some communications are degraded could be in principle be included and emulated in the flood sub-simulator – for instance, following an implementation similar to the OCTOPUS Virtual Environment [1] – but are not included in the present prototype. For further information on the simulator’s interaction models we refer to the technical report [7].
3.4. The Peer Network

In Fig. 3, the peers in the peer network are depicted as ellipses. These peers constitute all the actors participating in a possible disaster scene. They are either emergency peers, that is, agents acting on behalf of human emergency personnel, or system peers, that is, digital services providing information that are essential for coordinated task execution. These peers can run LCC interactions with other network peers (gray arrows in Fig. 3) and with the simulator (black arrows in Fig. 3).

We designed and implemented a sub-scenario where a network of different participating actors, with different roles, coordinate each other and rely on existing service components. This scenario is running on the OpenKnowledge kernel infrastructure (i.e. each of the participating peers is running an OpenKnowledge kernel plug-in) and on a local P2P network. Our sub-scenario of the overall scenario flooding evacuation use case relates to the phase where an emergency coordinator (the firefighter coordinator in this case) sends a directive to its organizational peers (firefighters) that have to accomplish it. The coordination problem expressed above can be described in a verbose form as follows:

After having discovered all the available firefighters, the fire chief (FFC) selects those who are close enough to some given destinations; once the selection phase has taken place, the fire chief (FFC) assigns to each firefighter a specific destination. In order to find a route, the firefighter will rely on the route service component, whose aim is to provide a suitable path to the requester. The firefighter communicates each action to the simulator which checks the feasibility of the action (i.e., the action of moving could be impossible because a road is blocked); in case of a blocked road the firefighter asks the route service an alternative path (which doesn't contain the blocked road) and keeps trying alternatives till either there are free paths connecting him/her with the destination or this latter is unreachable. If the firefighter reaches the previously assigned destination he/she will confirm this to the FFC; otherwise he/she will inform the FFC that he/she cannot reach the destination. The FFC reassigns a destination to firefighters who couldn't reach their originally assigned destination.
Three types of network peers (fire chief, firefighter, route service) and the simulator peers are involved in the task. The above description covers two different phases: one related to the selection of suitable firefighters and the other related to the actual task to be accomplished. Such task descriptions can also be sketched in a graphical representation, such as the activity diagram shown in Fig. 4. The diagram shows the workflow related to the coordination task *Go To Destination*. An interaction model, expressed in a compact, formal language such as LCC, can be derived either from the narrative description above or from an activity diagram. In this latter case, the conversion could be done either manually or automatically.

Before going into the details of the interactions involved, it is worth mentioning here how the selection phase takes place in this scenario. The fire chief peer can, as can every peer, take advantage from the OpenKnowledge kernel functionality allowing a peer to specify which selection strategy to use in order to select the peers it want to interact with (see Fig. 2B). In the sub-scenario considered, the need to adopt a specific selection strategy pertains to the fire chief peer only, while every other peer will accept all the subscribed peers. Prior to this bootstrap phase, each peer subscribes to a specific role with a given *subscription description*. For example, the subscription descriptions for the fire chief (*FFC*), a firefighter (*Fi*), the route service (*RS*) may be *fire_chief(D1,D2,...,Dn)*, *firefighter(Pi)* and *router* respectively. Here, *Di* indicates a possible destination where to send the firefighters and *Pi* represents the current position of a firefighter *Fi*. Based on this information, the selecting peer *FFC* can filter the firefighter peers according to their actual positions. Fig. 5 sketches these phases (subscription and selection) in the case of four firefighters. Possibly, some of them will be preferred over the others. Of course, the simulator peers also have to enact this phase; these details are omitted here for clarity purposes. After the whole bootstrap phase is completed, the interaction run can commence; in our case, a possible peer configuration may look like that illustrated in Fig. 6. Here, the firefighter peer F4 was filtered out in the selection phase. Also, notice that the interaction *connect* has not been depicted in Fig. 6 for clarity purposes, but is necessary to connect each firefighter peer to the simulator.
In the next section, details on the interaction model involved in this sub-scenario are given. The focus will be on describing the interaction specifications rather than describing how the peers solve the constraints defined in them.

4. EXAMPLES OF INTERACTION MODELS

As examples of the implementation and usage of the interaction model language, Fig. 7, 8 and 9 show some excerpts of LCC code related to the firefighter coordinator, firefighter and route service roles respectively. It is worth noting here that the LCC code presented in this paper is only part of the code needed to run the interaction. The whole code is more lengthy, since it also entails the role of the simulator peer and its interactions with the network peers. However, for our purposes here – to indicate how to model the possible interactions among the network peers – we restrict our description to these excerpts.

(i) Fire chief

The fire chief, FFC, initiates the coordination task by taking on the role fire_chief. According to this role a list of available firefighters, FFL, must be retrieved in order to send an alert message to each of them. The role involves a recursion over the list FFL so that a destination meeting point (MP) can be assigned and an alert message can be sent to each firefighter First_FFL, in FFL. After having sent the messages to all the firefighters, the FFC assumes the role fire_chief2. Here the FFC receives a confirmation (or disconfirmation) of the reached (not reached) destination MP from a given firefighter Id (in Fig. 8 action_performer is a role assumed by the firefighter role) and the role recurses. If a disconfirmation is received, the FFC assigns to the firefighter Id a new destination - NewMP - which is then embedded in the alert message sent back.
(ii) Fire-fighters

Each firefighter, $FF$, receives from the fire chief, $FFC$, an alert message stating that the destination $MP$ has to be reached. Once the message is received, the firefighter, $FF$, assumes the role of $route\_finder$. If the firefighter, $FF$, is not yet at the location $Dest$ that must be reached, a vehicle will be needed together with the name of a digital service able to provide a route. A message for requesting a route between two locations ($Pos$ and $Dest$) is then sent to the route service $RS$. Once the firefighter $FF$ receives the requested path together with its sub-paths through a message sent by the route service $RS$, he/she stores the path in its local memory and assumes the role $action\_performer$ in order to communicate the move action to the simulator which in turn will check its feasibility (more details in the activity diagram in Fig. 5).

(iii) Route Service

The route service, $RS$, after having received a route request from a route finder $Id$, selects a path from node $From$ to node $To$, splits the path in sub-paths and sends them back to the route finder $Id$. The route service role then recurses to be able to accept other requests. The route service can receive two types of messages from a route finder:

- A message asking for a path from $A$ to $B$ (a vehicle is also specified);
- A message asking for a path from $A$ to $B$ (a vehicle is also specified) excluding certain sub-paths (i.e., the blocked subpaths).

5. SIMULATED EVACUATION SCENARIO

We developed an initial evacuation scenario prototype in which the coordination activities between the network peers can be executed, visualised and analysed to evaluate the effectiveness of the LCC protocol.
The schematic diagram of the selected scenario involved is depicted in Fig. 6 including the peers involved, the interaction models and the simulator peers. In our first prototype, the ongoing simulation and the resulting movements of the emergency peers are visualized through a Graphical User Interface (GUI), as shown in Fig. 10. The GUI is similar to the control panel used by the emergency coordinators, though with the important difference that this GUI provides an accurate reflection of what is really going on in the simulation, and receives its information directly from the simulator, whereas the control panel used by the emergency coordinator can only reflect information that the coordinator has gathered from the physical peers in the simulation, which may be incomplete or incorrect. Through the emergency GUI, users can visualize, among other things, the map of the emergency situation.

The map shows static geographic datasets (topographic map, probable flooding areas, escape roads, meeting points, refuge centers, sensor network) as well dynamic datasets (like the position of the firefighters involved in the simulation). Moreover, through the GUI, the emergency coordinator can perform actions (enact the emergency plan, recall digital services, change the map legend, search for other GIS datasets, send statements to the emergency peers, etc) as well as ask for information about the emergency situation (i.e. evacuated people, list of the emergency peers, blocked roads, situation of the meeting points and of the refuge centers, etc).

The main components of the GUI are:

- The main menu in the upper part of the GUI: loads initial configuration and available interaction models;
- The peer and coordination messages frame: shows the messages exchanged between the coordination peer and the emergency peers (different for each peer);
- The weather forecast frame: useful to analyze the current weather conditions;
- The news frame: gathers and shows general information about the emergency situation;
The map frame: shows, together with a real map of Trento and some static interesting points (coordination center, the meeting points, the refuge points, etc.), the movement of the peers during a simulated coordination task.

We are running a number of simulations involving the peer types and the interactions described above. In the one visualized in Fig. 10, we use 5 emergency peers (1 firefighter coordinator and 4 firefighters), 1 digital service (the route service) and the simulator peer; the coordination task was to follow a directive sent by the firefighter coordinator to its personnel. The directive consisted in moving to a given destination. We set 5 different destinations (meeting points) where to send the firefighters and initially fixed some specific roads as blocked. First, all the firefighters were at different locations. By running the simulation we obtained different outcomes for the four firefighters, due to the different conditions surrounding their destinations:

1. one firefighter was excluded from the interaction run by the fire chief selection strategy because it was not close to any destination;
2. one firefighter reached the assigned destination without any problem;
3. one firefighter reached the destination after having tried more than one alternative path;
4. the fourth firefighter couldn’t reach the assigned destination and, as a result, was redirected to a second location.

Fig. 10 shows a snapshot of the evolution of the simulated Go To Destination process. With these preliminary experiments we give an initial qualitative evaluation of the suitability of the protocol language to handle processes involving both agents and digital services. The modelling of interactions in the LCC language allows us to structure the coordination task whilst still maintaining some adaptability, this latter being the capability to cope with situations which are not known a priori (i.e., the blockage state of the road can change over time ).
A more comprehensive evaluation would entail two dimensions: (a) a qualitative validation of the simulation system and the interaction models by involving the local institutions working in crisis management. Expert people in the field can give useful feedback by commenting on how well the simulations reflect the actual plans. Moreover, the simulations can give useful hints to the experts by showing possible scenarios they didn’t foresee, since their plan is merely written and never actually tried in real situations. In this case, the simulation system could act as a training system for the experts; (b) an evaluation of how the approach scales. Such evaluation can be done by simulating a large number of peers participating in many interactions; measures on performance (e.g. response time) could be adopted. We are currently working in these directions.

6. RELATED WORK

In the present work, we show how an interaction-oriented approach, combined with an appropriate middleware infrastructure, might be adopted to handle the coordination problems arising when multiple agencies need to collaborate on emergency response management. Our approach envisages a twofold perspective. The first perspective is related to the provision of a controlled but chaotic environment in which to study the effectiveness of interaction models in coordinating agents in real time. Here the emphasis is on a multi-agent simulation as a testbed for investigating, along with the validity of interaction models, the impact of a P2P infrastructure in emergency response. The idea of applying multi-agent models for such purposes is not new, since a number of complex multi-agent systems (MAS) simulators are under development [5,8,11,17].

The second perspective, the one distinctive to our work, is concerned with the specifics of an agent protocol language (LCC in our case) and its related middleware infrastructure specifically designed for expressing interactions in a P2P fashion, which we use to provide a mechanism for knowledge coalition formation and Web service composition, following the approach in [6, 16, 21]. Here the emphasis is on
MAS techniques for interoperability and coordination tasks employed in P2P architecture. The combination of the two perspectives has the potential to handle the dynamic and distributed aspects of emergency situations: in such scenarios, a P2P architecture is always preferred over a centralised client/server one since it allows the involvement of a large number of participants interacting in a distributed and decentralised manner. We describe collaboration between these participants through the specification of a message passing behaviour for each service involved.

Related research work is either specifically devised for the emergency management area or focused more on the architectural aspect. In particular CASCOM⁴, WORKPAD⁵, EGERIS⁶, EUROPCOM⁷, POMPEI⁸, POPEYE⁹, WIN¹⁰ are few such projects. For example, in the CASCOM project (Context-Aware Business Application Service Coordination in Mobile Computing Environments), an intelligent agent-based peer-to-peer (IP2P) environment is under development [4]. Here, the service coordination mechanism relies on Semantic Web technologies, such as OWL-S and WSMO, rather than explicit lightweight protocols. Also, the WORKPAD project aims at designing and developing an innovative software infrastructure (software, models, services, etc.) for supporting collaborative work of human operators in emergency/disaster scenarios [9]. A set of front-end peer-to-peer communities providing services to human workers, mainly by adaptively enacting processes on mobile ad-hoc networks, is part of the system developed [2]. Each community is lead by a super-peer, which is the only peer managing workflow composition and coordination in an adaptive manner. In this case, a mechanism based on our approach would allow each peer to execute, and eventually modify, the workflow, thanks to the sharing of the multi-agent protocol. The work in [6] also represents an effort in this direction: in this case the approach has been applied directly to business modelling. Our method has proved to be promising also in the field of e-Science [20].

⁴ http://www.ist-cascom.org
⁵ http://www.workpad-project.eu/description.htm
⁶ http://www.egeris.org
⁷ http://www.ist-europcom.org
⁸ http://www.pompei-eu.com
⁹ http://www.ist-popeye.org
¹⁰ http://www.win-eu.org
7. CONCLUSIONS AND FUTURE WORK

The purpose of this paper is to show how an approach based on interaction model sharing through a P2P network can be applied successfully to support coordination problems and related decision-making processes in the context of emergency response management.

Currently, our e-Response system is running on the first release of the OpenKnowledge kernel [3]. The developing OpenKnowledge platform [18] provides a means for us to move to a more dynamic scenario: peers will be able to use the DDS (discovery service) to search for appropriate interaction models, look for suitable peers, and make use of some other important functionalities, such as computing trust, verifying interaction models, providing automated assistance for generation of LCC, etc.

But how does this help our (flood) emergency scenario? Let us consider the interaction model of previous section. A more dynamic version of this scenario would be to have a fire chief asking firefighters to perform certain actions. However, instead of specifying how the firefighter will get the path between two nodes, we may keep this a general constraint to be solved by the peer. At runtime, the peer may then succeed in satisfying the constraint either by consulting its own knowledge base or by using some interaction model to communicate with others in order to achieve its goal. It could then use the discovery service of the OpenKnowledge system to search for suitable interaction models and peers. One solution could be to ask a route service, another one could be to consult other peers (other emergency personnel, available sensor networks or even normal citizens equipped with mobile phones) in that vicinity: in short, we enhance the peer autonomy in the decision making process.

For emergency response scenarios, this ultra dynamic P2P approach is crucial since it implies that even if parts of the system fail for one reason or another, peers will still be able to find other methods for achieving their critical goals. In the near future, we will address a number of issues and limitations of the current system: namely, improved interaction model design, use of appropriate workflow technology and more dynamic interactions.
8. ACKNOWLEDGMENTS

This work has been supported by the OpenKnowledge project (FP6-027253).

The scenario and flooding use case is based on extensive interviews with involved local authorities and analysis of current local emergency plans. We want to thank here the availability of Mario Perghem Gelmi, Marco Bommassar, Federico Bortolotti, Mirko Gasperotti and Michele Zanolli. We would like to thank David Dupplaw for the development of the emergency GUI. We acknowledge Fausto Giunchiglia and Dave Robertson for comments and advice on the e-Response system architecture.

9. REFERENCES


Fig. 1: Basic LCC interaction

\[
\begin{align*}
a(r_1, A_1) &::= \\
&\quad \text{ask}(X) \Rightarrow a(r_2, A_2) \leftarrow \text{need}(X) \text{ then} \\
&\quad \text{update}(X) \leftarrow \text{return}(X) \leftarrow a(r_2, A_2) \\
\end{align*}
\]

\[
\begin{align*}
a(r_2, A_2) &::= \\
&\quad \text{ask}(X) \leftarrow a(r_1, A_1) \\
&\quad \text{return}(X) \Rightarrow a(r_1, A_1) \leftarrow \text{get}(X) \\
\end{align*}
\]
Fig. 2: OpenKnowledge lifecycle
Fig. 3: The e-Response system’s architecture
Fig. 4: Activity diagram of the coordination task “Go to Destination”
Fig. 5: Bootstrap phase in the e-Response task “Go To Destination”
Fig. 6: Simulated evacuation scenario (GoToDestination) components and main interactions
\[ \text{affire\_chief}(\text{FFL}, \text{FFC}) : \]
\[ \left\{ \begin{array}{l}
\text{alert}(\text{MP}) \Rightarrow \text{affirefighter}, \text{First\_FFL} \leftarrow \text{FFL} = \left[ \text{First\_FFL}, \text{FFL}\_T \right] \text{ and} \\
\quad \text{assign}(\text{MP}, \text{First\_FFL}) \\
\text{null} \leftarrow \text{FFL} = \left[ \right] \\
\text{affire\_chief}\_2, \text{FFC}) \\
\end{array} \right. \]

\[ \text{affire\_chief}\_2, \text{FFC}) : \]
\[ \left\{ \begin{array}{l}
\text{confirm}(\text{Id}, \text{MP}) \Leftarrow a(\text{action\_performer}, \text{Id}) \text{ or} \\
\text{disconfirm}(\text{Id}, \text{MP}) \Leftarrow a(\text{affirefighter}, \text{Id}) \text{ then} \\
\text{alert}(\text{NewMP}) \Rightarrow a(\text{affirefighter}, \text{Id}) \leftarrow \text{assign\_newDest}(\text{MP}, \text{NewMP}, \text{Id}) \\
\text{affire\_chief}\_2, \text{FFC}) \\
\end{array} \right. \]

\textbf{Fig. 7:} LCC fragment for the firefighter coordinator role
affirefighter,FF):.

    alert(MP) ⇐ affirefighter_coordinator( ), FFC) then

    a(route_finder(MP), FF).

a(route_finder(Dest),Id)::

    \[
    \begin{align*}
    &\left( \text{request route(Pos, Dest, Vehicle, BanSubPaths)} \Rightarrow a(\text{route_service, RS}) \text{ then} \right) \text{ or} \\
    &\quad \left( \begin{array}{l}
    \left( \begin{array}{l}
    \text{at(Pos) and not(Pos = Dest) and blocked_roads(BanSubPaths) and} \\
    \text{set_vehicle(Vehicle) and get_route_serviceID(RS)}
    \end{array}\right) \\
    \text{then}
    \end{array}\right)
    \end{align*}
    \]

    \[
    \begin{align*}
    &\left( \text{request route(Pos, Dest, Vehicle)} \Rightarrow a(\text{route_service, RS}) \text{ then} \right) \\
    &\quad \left( \begin{array}{l}
    \left( \begin{array}{l}
    \text{at(Pos) and not(Pos = Dest) and set_vehicle(Vehicle) and get_route_serviceID(RS)}
    \end{array}\right)
    \end{array}\right)
    \end{align*}
    \]

    \[
    \begin{align*}
    &\text{route(From, To, Path, SubPaths)} ⇐ a(\text{route_service, RS}) \text{ then} \\
    &\quad a(\text{action_performer(move(From, To, Path, Vehicle), FFC, Id)} ⇐ \text{decompose(Path, SubPaths, Vehicle)}
    \end{align*}
    \]

Fig. 8: LCC fragment for the firefighter role
Fig. 9: LCC fragment for the route service role
Fig. 10: The emergency simulation Graphical User Interface