UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

OPENKNOWLEDGE AT WORK: EXPLORING CENTRALIZED AND DECENTRALIZED INFORMATION GATHERING IN EMERGENCY CONTEXTS

Gaia Trecarichi, Veronica Rizzi, Lorenzino Vaccari, Maurizio Marchese and Paolo Besana

January 2009

Technical Report # DISI-09-011

Also: submitted to 6th International Conference on Information Systems for Crisis Response and Management

OpenKnowledge at work: exploring centralized and decentralized information gathering in emergency contexts

Gaia Trecarichi
University of Trento
gtrecari@disi.unitn.it

Veronica Rizzi
University of Trento
vrizzi@disi.unitn.it

Lorenzino Vaccari
University of Trento
vaccari@disi.unitn.it

Maurizio Marchese
University of Trento
maurizio.marchese@unitn.it

Paolo Besana
University of Edinburgh
p.besana@sms.ed.ac.uk

ABSTRACT

Real-world experience teaches us that to manage emergencies, efficient crisis response coordination is crucial; ICT infrastructures are effective in supporting the people involved in such contexts, by supporting effective ways of interaction. They also should provide innovative means of communication and information management. At present, centralized architectures are mostly used for this purpose; however, alternative infrastructures based on the use of distributed information sources, are currently being explored, studied and analyzed. This paper aims at investigating the capability of a novel approach (developed within the European project OpenKnowledge[1]) to support centralized as well as decentralized architectures for information gathering. For this purpose we developed an agent-based e-Response simulation environment fully integrated with the OpenKnowledge infrastructure and through which existing emergency plans are modelled and simulated. Preliminary results show the OpenKnowledge capability of supporting the two afore-mentioned architectures and, under ideal assumptions, a comparable performance in both cases.

Keywords

Interaction Modeling, p2p Networks, Process Coordination, Centralized Knowledge Sharing, p2p Knowledge Sharing, Crisis Management, Agent-based Emergencies Simulation.

INTRODUCTION

All phases of emergency response management - that in the following we will reference as emergency response (e-Response) activities – depend on data from a variety of sources. Moreover, during an emergency it is critical to have the right data, at the right time, displayed logically and contextually, to respond and take the appropriate actions. At present, most of the information management infrastructures required for dealing with emergencies are based on centralized architectures that (i) are specifically designed prior to the emergency, (ii) gather centrally the available information, (iii) distribute it upon request to the appropriate agents (e.g., emergency personnel, doctors, citizens). While centralized infrastructures provide a number of significant advantages (in terms of quality control, reliability, trustworthiness, sustainability, etc.), they also present some well-known intrinsic problems (e.g., physical and conceptual bottlenecks, communication channel overloads, single point of failure). All these issues are taken into account in the design and deployment of current mission-critical centralized systems. However, information sharing breakdowns are still possible and have occurred also in recent emergency events, such as the catastrophic passage of Hurricane Katrina in New Orleans in 2005[2]. Alternative data management (both for gathering and providing

[1] www.openk.org

[2] http://en.wikipedia.org/wiki/Effect_of_Hurricane_Katrina_on_New_Orleans

information) infrastructures are currently being explored, studied and analyzed (Lorincz, Malan, Fulford-Jones, Nawoj, Clavel, Shnayder, Mainland, Welsh, Moulton, 2004; Mecella, Catarci, Angelaccio, Buttazzi, Krek, Dustdar, Vetere, 2006; D'Aprano, de Leoni, Mecella, 2007) in order to support data sharing also in the absence of a centralized infrastructure.

In this study, we explore the flexibility and adaptability of the framework developed within the FP7 European project OpenKnowledge[1]. This framework provides a distributed infrastructure, that enable peers to find and coordinate with each other by publishing, discovering and executing *interaction models*, i.e. multi party conversational protocols; the key novelty of the approach is that no a-priori agreement or knowledge of the conversation partners is needed to have meaningful interactions. In this paper, the proposed OpenKnowledge (OK) infrastructure is used to explore its capability to support both centralized and decentralized architectures for information gathering in open environments.

For this purpose, we built a simulation–based test-bed fully integrated with the OK platform. The final goal of such virtual environment is to evaluate this framework in the e-Response domain. In particular, we implemented an e-Response simulation environment through which existing emergency plans based on real-data are modelled and simulated. Moreover, a suite of experiments has been designed and run to evaluate the performance of the OK e-Response system under specific assumptions. Preliminary results show the system's capability of supporting the two afore-mentioned architectures and a comparable performance in both cases. To summarize, the main contributions of the present paper are:

- The full use and testing of the current release of the OpenKnowledge infrastructure in a realistic and demanding use case;
- The provision of an agent-based simulation environment in which to evaluate interaction models, coordination tasks and diverse emergency information-gathering models;
- A preliminary analysis and comparison between the effectiveness of the OpenKnowledge infrastructure in centralized (hierarchical) and decentralized (p2p) information gathering.

The idea to explore and test the effectiveness of different data management architectures in "real-world" e-Response setting is not new. It is recognized (Bellamine-Ben Saoud, Dugdale, Pavard and Ben Ahmed,2004) that realistic computer simulations can be a valuable tool to investigate innovative solutions, such as new collaborative information systems, new cooperation configurations and communication devices. In fact, several multi agent-based simulation applications have been developed in diverse domains (Murakami, Minami, Kawasoe, Ishida, 2002; Bellamine-Ben Saoud et al, 2004; Bellamine-Ben Saoud, Ben Mena, Dugdale, Pavard and Ben Ahmed, 2006; Kanno, Morimoto, Furuta , 2006; Massaguer, Balasubramanian, Mehrotra, Venkatasubramanian , 2006).

Related research works are either specifically devised for the emergency management area or focused more on the architectural aspect. In particular CASCOM[3] , WORKPAD[4], EGERIS[5], EUROPCOM[6] , POMPEI[7], POPEYE[8] , WIN[9] are among 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 was developed (Helin, Klusch, Lopes, Fernandez, Schumacher, Schuldt, Bergenti, Kinnunen, 2005). Also, in the FireGrid project (Han, Potter, Beckett, Pringle, Sung-Han, Upadhyay, Wickler, Berry, Welch, Usmani, Torero, Tate,2009), a software architecture to help fire-fighters in e-Response events has been built. They realized an integrated system where real-time sensor data are processed using sophisticated models, running on HPC resources accessed via a Grid interface, and finally presented to humans using a command-and-control multi-agent system. In this case, a mechanism based on the OpenKnowledge approach would allow each agent to execute, and eventually modify, the workflow, thanks to the sharing of the multi-agent protocol.

In what follows, we first focus on a brief description of the OpenKnowledge framework. We then present the e-Response case study that we are using to ground our approach on realistic data and emergency plans. Next, we

[3] http://www.ist-cascom.org
[4] http://www.workpad-project.eu/description.htm
[5] http://www.egeris.org
[6] http://www.ist-europcom.org
[7] http://www.pompei-eu.com
[8] http://www.ist-popeye.org
[9] http://www.win-eu.org

describe the e-Response simulation environment architecture and, later on, we present the experimental test-bed designed for the evaluation; preliminary results of centralized vs. decentralized information management architectures are also discussed. In the final section, we draw our conclusion and future work.

THE OPENKNOWLEDGE FRAMEWORK

The OpenKnowledge framework has been developed within the European project OpenKnowledge and provides the underlying peer-to-peer infrastructure needed to run our experiments. The core concepts in OpenKnowledge are (Robertson, Giunchiglia, van Harmelen, Marchese, Sabou, Schorlemmer, Shadbolt, Siebes, Sierra, Walton, Dasmahapatra, Dupplaw, Lewis, Yatskevich, Kotoulas, Perreau de Pinninck. and Loizou, 2008): (1) the interactions between agents, defined by *interaction models* published by the authors on a peer-to-peer infrastructure with a keyword-based description; and (2) a distributed infrastructure, denoted as OpenKnowledge Kernel, that supports the publishing, discovery, execution, monitoring and management of the various interaction models.

The OpenKnowledge kernel (Siebes R., Dupplaw D., Kotoulas S., Perreau de Pinninck A., van Harmelen F., Robertson D., 2007) provides the layer that assorted services and applications can use to interact using a choreography-based architecture able to deal both with the semantic heterogeneity of the actors and with their discovery. The framework allows a direct translation of a choreography-oriented design, such as UML activity diagrams, into an executable application. The core concept is the *interaction model*, a multi party conversational protocol performed by different applications and service providers. These actors are the participants (called also *peers*) of the interactions, and they play roles in them. In an interaction all the roles have equal weight; the behavior of all the participants and, in particular, their exchange of messages are specified

Interaction models are written in Lightweight Coordination Calculus (LCC) (Robertson, 2004-1, Robertson, 2004-2) and published by the authors on a *distributed discovery service* (DDS) with a keyword-based description (Koutoulas, Siebes, 2007). LCC is an executable choreography language based on process calculus. An interaction model in LCC is a set of clauses, each of which defines how a role in the interaction must be performed. Roles are described by their type and by an identifier for the individual peer undertaking that role. Participants in an interaction take their *entry-role* and follow the unfolding of the clause specified using a combination of the sequence operator *('then'*) or choice operator (*'or'*) to connect messages and changes of role.
Messages are either outgoing to ('⇒') or incoming from ('⇐') another peer in a given role. A peer can take, during an interaction, more roles and can recursively take the same role (for example when processing a list). Message input/output or change of role is controlled by constraints defined using the normal logical operators for conjunction and disjunction. In its definition, LCC makes no commitment to the method used to solve constraints, so different participants might operate different constraint solvers (including human intervention). Figure 4 (in the next section) shows the LCC clauses for some interactions relevant to our experiments.

The peers that want to perform some task, such as verifying the state of flooding in some area, or providing the water-level information service, search for published interaction models for the task by sending a keyword-based query to the DDS. The DDS collects the published interaction models matching the description (the keywords are extended adding synonyms to improve recall) and sends back the list.

Interaction models and peers are designed by possibly different entities, and therefore the constraints and the peers' knowledge bases are unlikely to be perfectly corresponding. The heterogeneity problem is dealt splitting the task in three phases and limiting its scope: (1) the DDS selects the interactions by matching their descriptions using a simple query expansion mechanism; (2) the peers compare the constraints in the received interaction models with their own capabilities, and (3) finally the peers need to map the terms appearing in constraints and introduced by other peers (Besana, Robertson, 2007). The scope of the matching problem is limited to the specific interaction model in the second phase, and to the specific interaction run in the third phase.

The peer capabilities are provided by plug-in components, called OpenKnowledge Components (OKC). An OKC exposes normally a set of Java methods that are compared to the constraints in the interaction models. The comparison is performed between the signatures of the constraints and of the methods, transforming them into trees and verifying their similarity (Giunchiglia, McNeill, Yatskevich, Pane, J., Besana, Shvaiko , 2008). The signatures can be annotated with the semantics of each parameter, which, in turn, can be structured terms. The comparison process creates automatically adaptors that connect the constraints to the methods. An adaptor has a confidence level that reflects the similarity between each constraint and the best matching method: the average of all the confidences

of constraints gives a measure of how well the peer can execute an interaction, and it is used to select the most fitting one.

Once the peer has selected an interaction, it advertises its intention of interpreting one of its roles to the DDS by subscribing to it. Figure 1 shows the status of a network when roles in at least one interaction are all subscribed by at least one peer: e.g., IM_1 has all the roles subscribed, while role 3, r_3, is subscribed by two peers (P_3 and P_4). The peers have installed locally their OKCs: some shared OKCs can be found online (e.g., OKC_1, OKC_2, and OKC_3), and are available to all, while some OKCs might be private to a peer (OKC_4 for example is installed only on P_4).

THE E-RESPONSE CASE STUDY

We applied the OpenKnowledge framework in a case study involving emergency response coordination activities. In particular, our case study regards the evolution of a flooding event in Trento (Italy). The work moved its steps from a preliminary analysis on this kind of disaster.

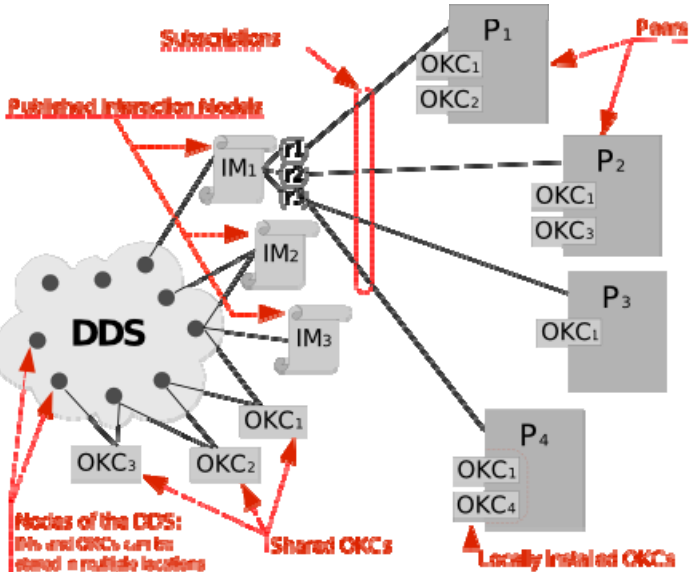


Figure 1: OpenKnowledge Architecture

The available analysis resulted from documents related to the current flood emergency plan in the Trentino region (Municipality of Trento, 2002, Autonomous Province of Trento, 2005) and from interviews with experts. We identified *emergency agents* (e.g., firemen, police, buses), the main organizations involved (e.g., Emergency Coordination Center, Fire Agency, Civil Protection Unit), a hierarchy between the actors (e.g., emergency chief, subordinates), *service agents* (e.g., water level sensors, route services, weather forecast services, GIS services) and a number of possible *scenarios*, that is, possible interactions among the agents. Our analysis resulted in the modeling of the *pre-alarm* and the *evacuation* phases of the emergency plan.

These phases unfold as follows: an Emergency Monitoring System (EMS) continuously gathers data from water level sensors placed in strategic points (break points) along the river. It also checks weather information in order to enrich the data needed to predict the evolution of a potential flooding. When a critical situation is registered, the automatic system notifies the emergency chief so to make her/him able to take a decision on whether to enact the evacuation plan or not. The evacuation plan consists of agents (e.g., emergency subordinates such fire-fighters) moving to specific locations assigned by the chief. In order to move, they need to perform some activities: choosing a path to follow by asking a route service; checking if the path is practicable by interacting with the Civil Protection or with available reporters distributed in the area; proceeding along the path. The Civil Protection is able to serve requests on the blockage state of a given path, since it continuously gathers information from reporters (e.g., sensors) scattered around the emergency area and reporting the water level registered at their locations.

Figure 2 gives a schematic view of the two phases involved in our case study. It shows the involved actors (denoted by round circles), their interactions and the kind of information exchanged. The smooth rectangle denotes the

simulator, that is, the virtual environment where all the peers act; obviously, it doesn't correspond to any entity in the reality, therefore, we don't describe it in this context. However, the simulator is essential for the simulation-based test-bed and will be illustrated in detail in the next section.
The figure also shows two different evacuation sub-scenarios: in both of them, an agent need to get information on route's practicability but while in one case an emergency subordinate *ES* gets route information by asking the Civil Protection *CP* (area above the red line), in the other one the agent interacts directly with reporters *r1/r2/r5* physically present at the locations of interest (area below the red line). These two ways of gathering information are referred to as *centralized* and *decentralized strategies*.

On the basis of the described case study, we built a test-bed which simulates the evacuation phase. More details on the agents and their interactions involved in the evacuation phase are given in the next section.

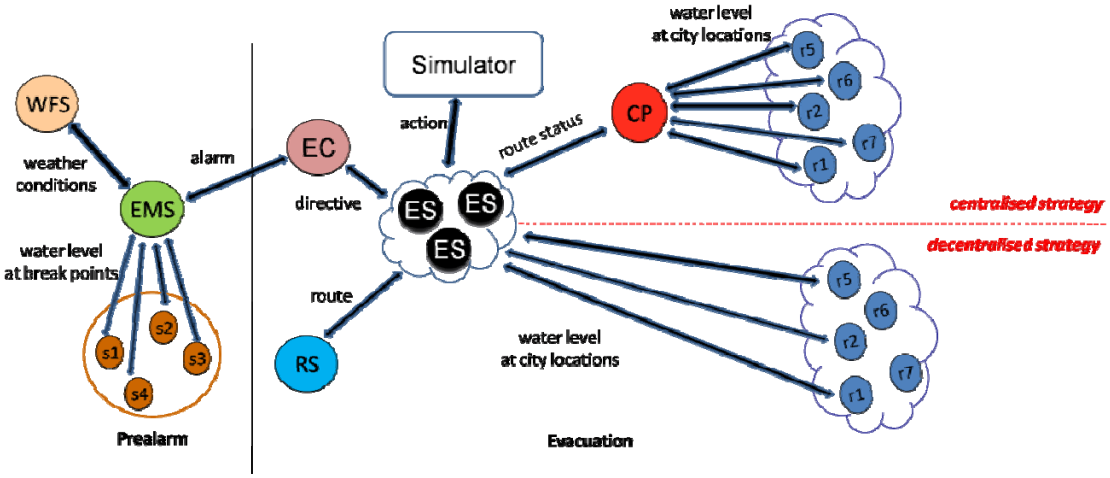


Figure 2: e-Response case study

THE E-RESPONSE SYSTEM ARCHITECTURE

Due to the critical nature of emergency situations, the infrastructures supporting crisis response operations require a full testing and a proper evaluation. With this in mind, we built an e-Response simulation environment which makes use of the OpenKnowledge kernel. The present simulation environment is based on the system presented in (Marchese, Vaccari, Trecarichi, Osman, McNeill, 2008) and extends it both in a complete integration with the OpenKnowledge kernel and in the inclusion of a realistic flood-simulator. Through simulations, it is possible to estimate how the platform could perform in realistic emergency scenarios. In particular, the developed e-Response simulation system is used to: (1) model the behaviour of each peer involved in an e-response activity, (2) execute predefined interaction models within a P2P infrastructure and (3) visualize and analyze a simulated coordination task through a Graphical User Interface (GUI). The e-Response system is composed of two main components: the peer network and the e-Response simulator. Figure 3 sketches its overall architecture. All peers are equipped with their specific OpenKnowledge plug-in component(s); each black arrow represents a different interaction model, which also represents the flow of information between peers; the greys arrows indicate interactions among network peers only. In the next two subsections, we illustrate the peer network and the e-Response simulator respectively.

The peer network

The peer network represents the group of agents involved in a simulated coordination task. An agent in the peer network can interact with other agents, perform some actions (e.g., moving along a road) and gather information. (e.g., sense the water level in its vicinity).

In order to perform an action or receive sensory information near its location, a peer must connect to the simulator by enacting the "connect" interaction model. Once added to the simulation, the connected peer periodically receives sensory information from the simulator via the "sensory-info" interaction model; finally, to perform an action, a connected peer enacts the "coordinate-action" interaction model which models the action coordination with the simulator. The connected network peers are called *physical peers* (shaded ellipses in Figure 3). Of course, not all peers must connect to the simulator.

Non-physical peers, such as a route service that provides existing routes, do not need to communicate with the controller but only with other peers in the peer network. 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 Figure. 3.

In what follows, we describe the agents and their interactions, which take place during the evacuation phase of Figure 2. Such agents are:

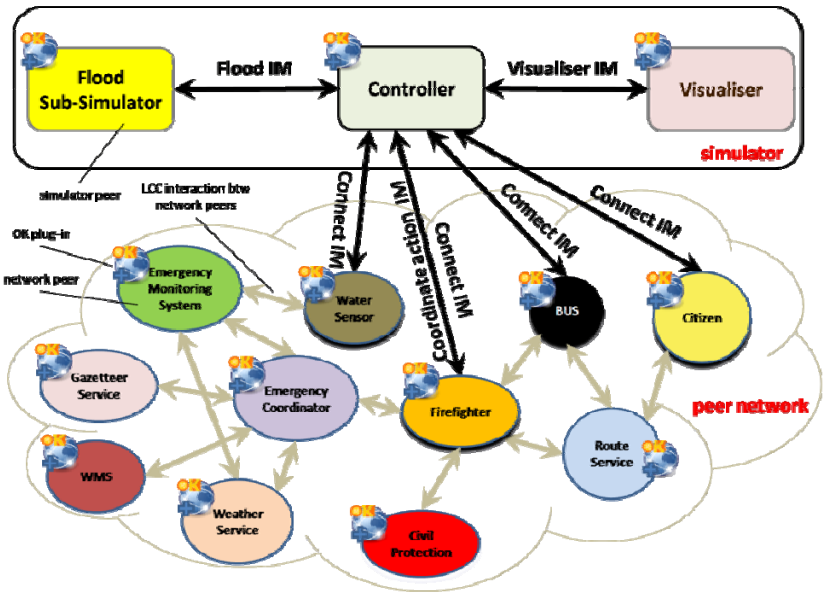


Figure 3: The e-Response system's architecture

- *Emergency Chief (EC)*: a top-level authority responsible for the coordination of all the emergency activities, from the propagation of the alarm to resource allocation. In the simulation, it just sends the directive of moving to a given location to its subordinates;
- *Emergency Subordinate (ES)*: an agent (e.g., a fire-fighter, a policeman, a doctor) which needs to move to a specific location;
- *Route Service (RS)*: a service providing routes which connect two given locations and do not pass by a set of "undesired" locations;
- *Civil Protection (CP)*: in our simulation, this agency informs people on the blockage status of a given route;
- *Reporter (r)*: an agent (e.g., citizen, sensor device) providing information on the water level registered at its location. In our simulation, reporters are assumed to be sensor devices located at specific locations.

The interactions among the above agents - modeled as LCC specifications - can be recap as follows:

1. *Start evacuation*: the *EC* alerts its *ES* to go to a specific location. The team-member prepares to satisfy the directive;
2. *Find a route*: a subordinate *ES* asks a route service *RS* for an existing route connecting its location to the destination;
3. *Check the route practicability*: a subordinate *ES* requests information to the *CP* (centralized strategy) or asks reporters dislocated in the interested area (decentralized strategy);
4. *Gather real-time data from reporters*: an agent (e.g., *CP*, *ES*) asks information about the water level to a group of reporters;

The "*Start evacuation*" interaction model is the main one in the selected use case. It simulates the evacuation phase and can be used in all those situations where an emergency chief sends the directive of reaching specific locations to its subordinates. In short, an emergency subordinate *ES* receives an alert message from the chief and resolves some constraints in order to set the goal to be achieved (reach the goal destination *G*) and get the current position. The

activities of ES thus evolve through three key roles: the *goal achiever* role which abstractly models the activity of searching for a path and moving towards the goal; the *free_path_finder* role which defines the operations needed to find a free path; the *goal mover* role which models the actions needed to move towards the goal destination. Figure 4 shows an LCC code snippet for two of the key *ES* roles. The constraints specified in bold indicate that they are solved by enacting separate interaction models. This is a key functionality of the OK platform, since it allows to write simple, modular and reusable LCC specifications. The lack of space prevents us to describe all the developed LCC interactions. For more details, we direct the interested reader to the technical report (Trecarichi, Rizzi, Vaccari, Marchese, 2008).

```
a(emergency_subordinate,FF)::

 alert(G) <= a(emergency_chief,FFC) then
   null <- set_goal(G) and get_current_position(CurrPos) then
     a(goal_achiever(CurrPos,G),FF)


a(goal_achiever(From,To),GA)::

 (
   //moving peer already at destination
   null <- equal(To,From) and setGoalAchieved(To)

   or

   ( //try to find a free path
     a(free_path_finder(From,To,FreePath), GA) then

       //no free paths between From and To
       null <- FreePath=[] and setGoalUnreachable(To)

       or

       //move towards the goal destination
       a(goal_mover(From,To,FreePath),GA)
   )
 )
```

```
a(free_path_finder(From,To,FreePath), FRF) ::

 null <- find_path(From,To,Path) then
 (
   //no paths are found
   null <- Path=[] and makeEmptyList(FreePath)

   or

   (
    //check if the path is free
    null <- request_path_state(Path,PathState) and
            path_free(PathState) then
       null <- assign(Path,FreePath)
   )

   or

   //search for an alternative path which is free
   a(free_path_finder(From,To,FreePath), FRF)
 )
```

Figure 4: LCC fragment for the emergency subordinate role

The e-Response Simulator

The simulator is designed to represent the environment where all the involved agents act. It is composed of three modules which are themselves peers: the *controller*, the *flood sub-simulator*, and the *visualiser* (see Figure 3). The controller regulates the simulation cycles and the management of the simulated agent activities; the flood sub-simulator reproduces the actual evolution of the 1966 flood in Trento; the visualiser stores simulation information used by the GUI to view a simulation run in a step-by-step way. The simulator does not interfere or help coordinate peer's actions in the peer network. It is used to simulate the real world.

The Controller

The controller is the core of the simulator: it drives the simulation cycles and keeps track of the current state of the world. In order to achieve that, it needs to know what changes are happening to the world and updates its state accordingly. After updating its state, it also informs the relevant peers of these changes. The simulation thus evolves through cycles (or time-steps). A simulation cycle foresees two main operations:

- *Gathering changes*: the controller receives information about the changes that happened to the world: (a) it receives the disaster (e.g., flood) changes from the disaster sub-simulator via the "flood" interaction model and (b) it serves requests of performing (move) actions with the "coordinate-action" interaction model (see Figure 3). In this latter interaction, the controller verifies 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;
- *Informing peers*: the controller sends information about the changes that happened in the world: (a) it sends, at each time-step, local changes to each connected peer via the "sensory-info" interaction model and (b) it sends to the visualiser information on - (i) the locations of all connected peers; (ii) the status of the reporter peers (e.g., available, responding to requests) and (iii) the water level registered; here, the "visualiser" interaction model is used.

Before a simulation cycle commences, some preliminary activities are performed such as: establishing key parameters (e.g., maximum number of simulation cycles, timeouts, water level thresholds), connecting with the flood sub-simulator, sharing with it the initial topology of the world, and adding connecting peers. Once a

simulation cycle terminates, the controller updates the time-step and starts the next cycle. Notice that, due to the modularity of the above architecture, it is reasonably easy to add as many disaster sub-simulators (e.g., landslides, earthquake, volcanic eruption, etc.) as needed;

The Flood-SubSimulator

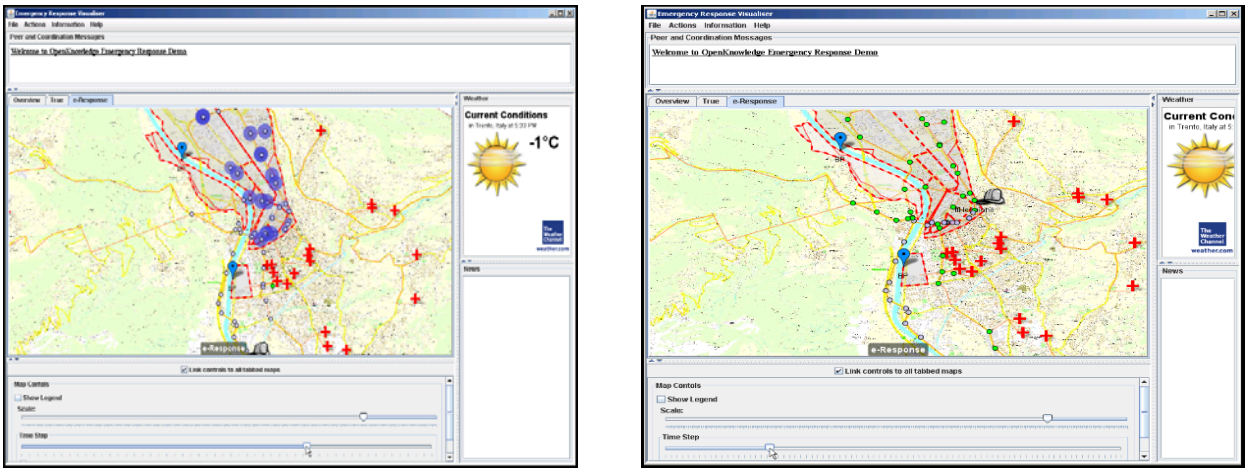
The flood sub-simulator goal is to simulate a flood in town of Trento (Italy). The equation defined in its core OKC is based on flooding levels and flooding timings deduced from a work presented in (Alkema, Cavallin, De Amicis and Zanchi, 2001). This study is based on the digital terrain model of the river Adige valley, on historical hydrological data of the flood experienced in Trento in 1966, on the localization of ruptures of the river Adige's dikes and on floodplain topography changes from year 1966 to year 2000.

To the purpose of our test-bed the territory is divided into flooded areas: each area is characterised by the maximum water height reached during the inundation and the time when this level was touched. We digitalised these flooded areas from (Alkema et al., 2001) analysis assuming that the time required to reach the maximum flood level was always one hour. We then stored this digitalised data in different tables in a geographical database. Each table has a field representing x,y coordinates of digitalized points. At simulation time, only the data of the interested area are joined in a single table using an Open Geospatial Consortium standard spatial SQL query.

The flood sub-simulator has been developed in Java and it is fully integrated into the OpenKnowledge kernel. The main component is an OK peer that subscribes to two interaction models. The first interaction model is enacted at the beginning of the simulation. It shares the topology (e.g., point of interests and roads) of the world between the controller and the flood sub-simulator peers and stores, in the controller local knowledge, the connection state of the sub-simulator. The second interaction model is used by the controller at every time step, in order to get from the flood sub-simulator flood level changes registered at nodes in the topology.

The Visualiser

This component enables the GUI used to visualise the simulation. In particular, the GUI shows the information provided by the controller through the “visualiser” interaction model. At every time-step, the visualiser receives the changes and updates its history according to the new information. The update results in a change on the GUI. Figure 6 shows the appearance of the GUI in two simulated scenarios.



a) Simulation of centralized information gathering

b) Simulation on decentralized information gathering

Figure 6: Emergency GUI

A *green dot* represents a reporter agent available for giving information on the water level registered; a *grey dot* represents a reporter agent giving this information; the water level at a location is depicted as a blue circle, which size depends on how high the water level is; the *hat* represents the emergency subordinate.

For more detailed information on the simulator's implementation please refer to the technical report (Trecarichi et al., 2008).

THE EXPERIMENTAL TEST-BED

We built a test-bed which is based on simulations and integrated with the OpenKnowledge platform. With such test-bed, we want to evaluate the OK framework in the e-Response domain. We designed a series of experiments with a three-fold aim:

- Put the OpenKnowledge framework in action in a realistic (and demanding) scenario;
- Demonstrate that the technology provided by OpenKnowledge supports different models of information gathering (centralized vs. distributed scenario);
- Investigate when and in which cases the OK paradigm can improve performance in emergency tasks.

In respect to the last point, the hypotheses to be tested are that: (1) under ideal conditions, the OK system exploited in its decentralized nature is comparable in performance to traditional centralized systems and (2) it improves on conventional centralized systems when specific fault conditions arise. In this paper, we only test the first hypothesis, leaving the exploration of the second one to future work.

In the context of our experiments, what we measure as performance is: (i) the *percentage of times* an emergency subordinate arrives at destination; (ii) the *number of time-steps* needed to achieve this goal. These indicators are used to compute the results of the experiments and to make a comparison among them. A preliminary analysis on the variables which may affect the experimental results was carried out. Among the variables considered, those used in the experiments reported here, are: (A) the number of emergency subordinates; (B) the destination assigned, (therefore the routes to be followed); (C) the flooding law; (D) the locations where reporters are present; (E) the number of reporters at each location;

The experiment design

Two types of experiments were designed to simulate the two evacuation scenarios differing on the information gathering strategy. The scenarios evolve under ideal conditions, i.e., the absence of faults (like failures in communication and inaccurate signalling) is assumed.

In order to interpret appropriately the obtained results, it is important to recall the assumptions made:

I. The Civil Protection (CP) peer has infinite resources: it can serve any number of simultaneous requests and its communication channel never breaks. Bottleneck problems due to overwhelming requests thus never occur;
II. An emergency subordinate asking information to a group of reporters will receive all the answers within a time-step. This is due to how the time-step interval is set: the value is such that the time elapsing between one time-step and the next one is sufficiently high to guarantee the replies from all the reporters.

Under the above assumptions, we simulate a scenario where advantages and disadvantages of both centralized and decentralized architectures are kind of balanced. Table 1 summarizes the experiment configuration: each experiment is run 10 times; at each run, the only variables that change are the destination assigned and the locations where reporters are present. Such locations are determined according to the set of routes associated with the destination assigned. The flooding law, the number of emergency subordinates and reporters remain unchanged during all runs.

			Variable Settings				
Exp N°	Information Gathering	Runs	A	B	C	D	E
1	centralized	10	1	1 destination x run	fixed	60 x run	1
2	decentralized	10	1	1 destination x run	fixed	60 x run	1

Table 1: Experiments configuration

Experimental results

In order to run an experiment, a number of processes equal to the number of total agents considered need to be launched. For this purpose, we developed a Java program that reads the selected configuration variables from a database and then launches the processes with different combination of parameters. This mechanism exploits the distributed nature of the OK platform. For example, while the DDS is run in one server, the processes associated

with the reporter peers are launched in a different machine. Each experiment consists in the simulation of the evacuation scenario described in the previous section. Independently on the kind of strategy adopted, the final goal of an emergency subordinate is to safely reach the assigned destination. There are three situations which may happen: (1) the agent reaches the destination by following the first route found; (2) the agent finds blocked routes but finally reaches the destination after a number of alternative paths and (3) the agent doesn't reach the destination at all. We refer to each situation as the *outcome* of the experiment.

We run each experiment 10 times. The simulations were visualized on the GUI in order to analyze the movements of the emergency peers and verify the correct mechanism in the coordination among the agents.

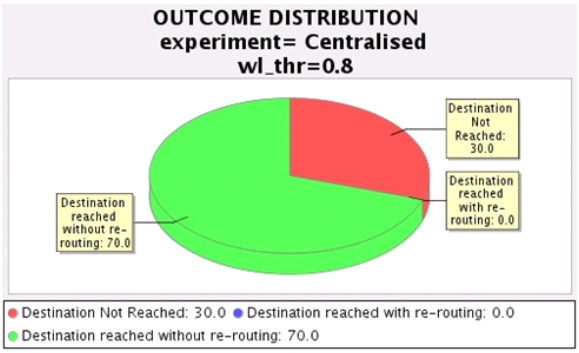


Figure 7: Outcome Distribution (centralized scenario)

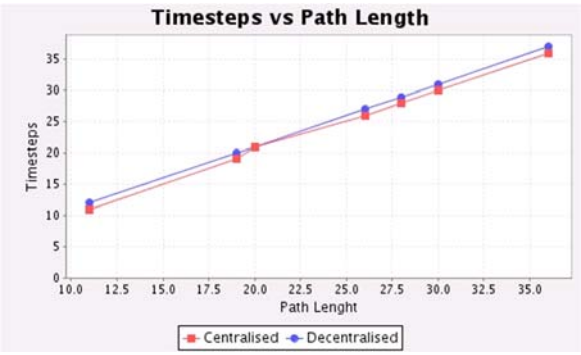


Figure 8: Time-steps vs. Path Length

Figures 5-a) and 5-b) show a simulation run for the centralized and the decentralized scenario respectively. Figure 5-a) shows the agent out from the flooded area; here, all the dots are grey, meaning that all reporters are being queried by the Civil Protection; some of them register high levels of water. Figure 5-b) shows the agent moving along a route which can be deduced by the grey dots ahead the agent; these dots represent in fact those reporters located along the route followed and therefore queried by the moving agent; all the other reporters remain available. Here, the OK paradigm is exploited in its decentralized nature, since the information gathering is based on the use of distributed information reporter agents and not on a unique provider, as in the first case.

Figure 7 shows the outcome distribution obtained by running 10 times the first experiment. As can be seen, 70 percent of the times, the experiment has outcome (1) (the peer reaches the destination without problems) while 30 percent of the time, the outcome is (3). The outcome (2) is never obtained. Although we setup the routes in order to cover different kind of areas (either safe or flood-prone areas), the case where an agent finds free routes after a re-routing never happens. This could be explained by considering how the design of the flooding law and its "speed" affects the evolution of the scenario. The outcome distribution related to the second experiment, which simulates the decentralized scenario, is identical to the one found for the first experiment and hence is not reported here. This result can be explained with the assumptions previously made: asking information on the route's practicability to either the Civil Protection or reporters scattered around the city doesn't make the difference.

Figure 8 shows the time taken (measured as the number of simulation time-steps) by an agent to reach the goal location according to the shortest distance (in terms of intermediate locations) between the initial position and the final destination. The trend is shown for both experiments. It can be observed that, in both cases, the time needed to achieve the goal is nearly equal to the shortest distance. This can be explained by how the simulation is designed – an agent moves from a location to the next one exactly in a time-step – and by the missing outcome (2). Finally, Figure 8 reveals very similar trends for both centralized and decentralized scenarios. Again, this is mainly due to the assumptions made and the variable settings.

In view of the results described above, we can conclude that our first expectation is met: the use of the OK framework supports both architectures (centralized and decentralized) and provides comparable performances under the selected - ideal - assumptions.

CONCLUSIONS AND FUTURE WORK

The aim of this paper is to show how the OK framework is capable to support the coordination of emergency activities and how, in absence of fault conditions, the OK p2p framework is comparable in performance to

traditional centralized gathering approaches. An agent-based e-Response simulation system fully integrated with the OpenKnowledge infrastructure has been developed. The system currently runs on Java and exploits the distributed nature of the OK platform. It is used to model specific emergency scenarios and agents in terms of both LCC specifications and OKC's components. A suite of experiments has been designed and run to evaluate the performance of the OK e-Response system in different scenarios and under specific assumptions. The preliminary results thus obtained show how the OK infrastructure is equally effective in both centralized (hierarchical) and decentralized (p2p) information gathering.

We are currently working on further experiments. In particular, we want to repeat the experiments here reported by increasing the number of runs and by tuning parameters like the "speed" of the flood and the routes to follow, in a way that diversified outcomes can be obtained. In this way, we could reconfirm our hypothesis in a more robust setting. Also, we want to run experiments where specific fault conditions are injected. These experiments foresee two types of fault conditions: failures in the communication channels and inaccurate signaling.

The following variables will be considered in addition to the ones already described:

- *Distribution of trustworthy (reporter) peers*: the number of reporter peers having a trustworthy behaviour, i.e., peers which always report accurate water level values. By setting this variable, the fault due to inaccurate signaling, its location and its severity can be simulated.
- *Degradation of the CPU communication channel*: measured as the likelihood of a fault in the communication channel of the Civil Protection agent. For example, a degradation of the 80% means to have this agent serving incoming requests only 20% of the times.
- *Distribution of degraded reporters' communication channels*: defines, for each reporter communication channel, the probability of its disruption.

What we want to investigate by running these new experiments, is if - and eventually under which conditions - a complete p2p architecture improves the overall performance and robustness over traditional centralized architectures. Finally, from the point of view of the simulated scenarios and the agents involved, it would be interesting to consider the reporter agents as mobile emergency agents rather than fixed sensors. In this way, we could explore how the OK platform supports the coordination of team-members in an emergency site.

ACKNOWLEDGMENTS

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

We are thankful to Dave Robertson for his advices on the formulation of the hypothesis to test. Also, we are grateful to David Dupplaw for the development of the emergency GUI, to Juan Pane who developed the database we further extended and to Fiona McNeill who developed the initial prolog controller that we adapted to the OK platform.

REFERENCES

1. Alkema, D., Cavallin, A., De Amicis, M. and Zanchi, A. (2001) Evaluation of the flooding effects: the case of Trento (Italy). Valutazione degli effetti di un alluvione: il caso di Trento, *Studi Trentini di Scienze Naturali – Acta Geologica*, 78, 55-61.
2. Autonomous Province of Trento (2005) Provincial and municipal emergency protection guidelines. Linee guida per le attività di previsione, prevenzione, protezione e la pianificazione di protezione civile provinciale e comunale, *Technical report, Autonomous Province of Trento*.
3. Bellamine-Ben Saoud, N., Dugdale, J., Pavard, B. and Ben Ahmed, M. (2004) Towards planning for Emergency Activities in Large-Scale Accidents: An Interactive and Generic Agent-Based simulator, *Proceedings of the first International workshop on Information Systems for Crisis Response and Management*, Brussels, Belgium.
4. Bellamine-Ben Saoud, N., Ben Mena, T., Dugdale, J., Pavard, B. and Ben Ahmed, M. (2006) Assessing large scale emergency rescue plans: an agent based approach. *Special Issue on Emergency Management Systems. International Journal of Intelligent Control and Systems*, 11, 4, 260-271.
5. Besana, P. and Robertson, D. (2007) How Service Choreography Statistics Reduce the Ontology Mapping Problem, *Proceedings of the 6th International Semantic Web Conference, Busan, Korea.*

6. D'Aprano, F., de Leoni, M. and Mecella, M., (2007) Emulating Mobile Ad-hoc Networks of Hand-held Devices. The OCTOPUS Virtual Environment, *Proceedings of the first ACM Workshop on System Evaluation for Mobile Platform: Metrics, Methods, Tools and Platforms (MobiEval) at Mobisys*. Puerto Rico.
7. Giunchiglia, F., McNeill, F., Yatskevich, M., Pane, J., Besana, P. and Shvaiko, P. (2008) Approximate structure-preserving semantic matching, *Proceedings of the 7th Conference on Ontologies, DataBases, and Applications of Semantics*. Monterrey, Mexico.
8. Han, L., Potter, S., Beckett, G., Pringle, G., Sung-Han, K., Upadhyay, R., Wickler, G., Berry, D., Welch, S., Usmani, A., Torero, J. and Tate, A. (2009) FireGrid: An e-Infrastructure for Next-Generation Emergency Response Support, Submitted *to Royal Society Phil. Soc. A.*, (http://www.aiai.ed.ac.uk/project/ix/documents/2009/2009-philtransa-han-firegrid.pdf).
9. Helin, H., Klusch, M., Lopes, A., Fernandez, A., Schumacher, M., Schuldt, H., Bergenti, F. and Kinnunen, A. (2005) Context-aware Business Application Service Co-ordination in Mobile Computing Environments. *Proceedings of the fourth conference of Autonomous Agents and Multi Agent systems - Workshop on Ambient Intelligence - Agents for Ubiquitous Computing*, Utrecht, The Nederlands.
10. Kanno, T., Morimoto, Y. and Furuta, K. (2006) A distributed multi-agent simulation system for the assessment of disaster management systems, *International Journal of Risk Assessment and Management*, 6, 4/5/6, 528 – 544.
11. Kotoulas, S. and Siebes, R. (2007) OpenKnowledge deliverable 2.2: Adaptive routing in structured peer-to-peer overlays, *Technical report, OpenKnowledge project*.
12. Lorincz, K., Malan, D.J., Fulford-Jones, T.R.F., Nawoj, A. Clavel, A. Shnayder, V., Mainland, G. Welsh, M. Moulton, (2004) Sensor networks for emergency response: challenges and opportunities, *Pervasive Computing*, IEEE, Volume: 3, Issue: 4, Pages:16- 23, ISSN: 1536-1268.
13. Marchese, M., Vaccari, L., Trecarichi, G., Osman, N. and McNeill, F. (2008) Interaction models to support peer coordination in crisis management, 5th International Conference on Information Systems for Crisis Response and Management, Washington, DC, USA http://www.iscram.org/dmdocuments/ISCRAM2008/papers/ISCRAM2008_Marchese_etal_b.pdf
14. Massaguer, D., Balasubramanian, V., Mehrotra, S. and Venkatasubramanian, N. (2006) Multi-Agent Simulation of Disaster Response, *Proceedings of the First International Workshop on Agent Technology for Disaster Management*, Hakodate, Japan.
15. Mecella, M., Catarci, T., Angelaccio, M., Buttazzi, B., Krek, A., Dustdar, S., and Vetere, G. (2006) Workpad: an adaptive peer-to-peer software infrastructure for supporting collaborative work of human operators in emergency/disaster scenarios, *Proceedings of the 2006 International Symposium on Collaborative Technologies and Systems*, Las Vegas, Nevada, USA.
16. Municipality of Trento (2002) Civilian emergency plan for the municipality of Trento. Piano di Protezione Civile Comunale contro il Rischio Idrogeologico di Inondazione del Fiume Adige, *Technical report, Municipality of Trento.*
17. Murakami, Y., Minami, K., Kawasoe, T. and Ishida, T. (2002) Multi-agent simulation for crisis management, *Proceedings of the IEEE International Workshop on Knowledge Media Networking*, Kyoto, Japan.
18. Robertson, D., Giunchiglia, F., van Harmelen, F., Marchese, M., Sabou, M., Schorlemmer, M., Shadbolt, N., Siebes, R., Sierra, C., Walton, C., Dasmahapatra, S., Dupplaw, D., Lewis, P., Yatskevich, M., Kotoulas, S., Perreau de Pinninck, A. and Loizou, A. (2008) Open Knowledge: coordinating knowledge sharing through Peer-to-Peer Interaction, *Languages, Methodologies and Development Tools for Multi-Agent Systems*, 5118, 1-18.
19. Robertson, D. (2004-1) A lightweight coordination calculus for agent systems. *Declarative Agent Languages and Technologies*, 183-197.
20. Robertson, D. (2004-2) A lightweight method for coordination of agent oriented web services. *Proceedings of the AAAI Spring Symposium on Semantic Web Services*, California, USA.
21. Siebes, R., Dupplaw, D., Kotoulas, S., Perreau de Pinninck, A., van Harmelen, F., and Robertson, D. (2007) The OpenKnowledge System: An Interaction-Centered Approach to Knowledge Sharing, *Proceedings of the 15th International Conference on Cooperative information systems*, Vilamoura, Algarve, Portugal.

22. Trecarichi, G., Rizzi, V., Vaccari, L., Pane, J. and Marchese, M. (2008) Openknowledge deliverable 6.8: Summative report on use of OK approach in eResponse: integration and evaluation results. *Technical report, Openknowledge project.*