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
marchese@disi.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 both centralized and 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 and 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. 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).

In this study, we explore the flexibility and adaptability of the framework developed within the OpenKnowledge project[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. In this

[1] www.openk.org

paper, the proposed OpenKnowledge 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 OpenKnowledge 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 OpenKnowledge 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.

The idea to explore and test the effectiveness of different data management architectures in "real-world" e-Response settings 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. Several multi agent-based simulation applications have been developed in diverse domains (Bellamine-Ben Saoud et al, 2004; Bellamine-Ben Saoud, Ben Mena, Dugdale, Pavard and Ben Ahmed, 2006; Kanno, Morimoto and Furuta, 2006; Massaguer, Balasubramanian, Mehrotra and Venkatasubramanian, 2006; Murakami, Minami, Kawasoe and Ishida, 2002).

Other research works are focused more on the architectural aspects of the ICT infrastructures: CASCOM[2], WORKPAD[3], and EGERIS[4], are among such projects. For example, in the CASCOM project, an intelligent agent-based peer-to-peer (Ip2p) environment was developed (Helin, Klusch, Lopes, Fernandez, Schumacher, Schuldt, Bergenti and Kinnunen, 2005). Also, in the FireGrid project (Han, Potter, Beckett, Pringle, Sung-Han, Upadhyay, Wickler, Berry, Welch, Usmani, Torero and Tate, 2009), a software architecture to help fire-fighters in e-Response events was built. Here, real-time sensor data are processed using sophisticated models and finally presented to humans using a command-and-control multi-agent system.

In what follows, we first focus on a brief description of the OpenKnowledge framework. We then present the e-Response case study used to ground our approach on realistic data and emergency plans. Next, we describe the e-Response simulation environment architecture and, afterwards, 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 conclusions and future work.

THE OPENKNOWLEDGE FRAMEWORK

The OpenKnowledge (OK) framework provides the underlying peer-to-peer infrastructure needed to run our experiments. The core concepts in OK are: (1) the interactions between agents, defined by *interaction models*; and (2) a distributed infrastructure, denoted as OK *kernel*, that supports the publishing, discovery, execution, monitoring and management of the various interaction models.

Interaction models are written in Lightweight Coordination Calculus (LCC) (Robertson, 2004-1), an executable choreography language based on process calculus. An LCC interaction model is a set of clauses, each of which defines the expected behavior of a role. Participants in an interaction take their *entry-role* and follow the unfolding of the clause specified using a combination of sequence *('then'*) and choice (*'or'*) operators to connect messages and changes of role. A peer can take, during an interaction, more roles and can recursively take the same role (for example when processing a list). Messages are either outgoing to ('⇒') or incoming from ('⇐') another peer in a given role and their exchange is subject to constraint satisfaction. 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 OK kernel (Siebes, Dupplaw, Kotoulas, Perreau de Pinninck, van Harmelen and Robertson, 2007) provides the layer that assorted services can use to interact using a choreography-based architecture able to deal with both the semantic heterogeneity of the actors and their discovery.

[2] http://www.ist-cascom.org

[3] http://www.workpad-project.eu/description.htm

[4] http://www.egeris.org

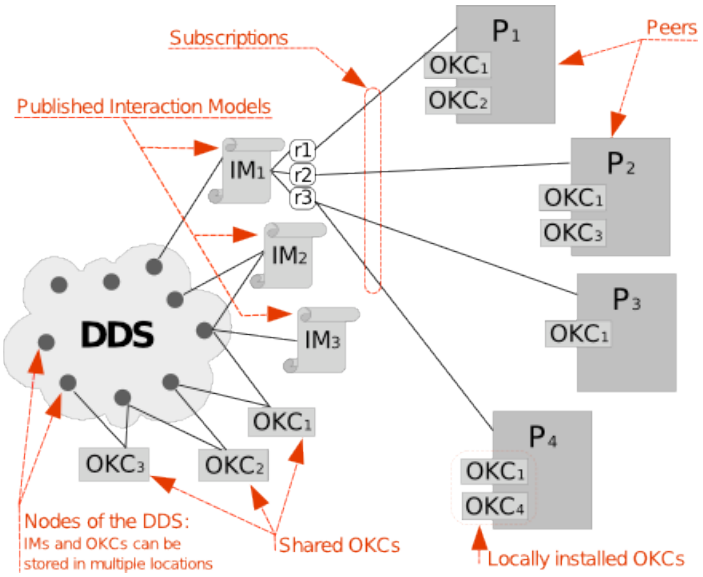


Figure 1: OpenKnowledge Architecture

Interaction models are published by the authors on a *distributed discovery service* (DDS) with a keyword-based description. A peer willing to perform a task (e.g., verifying the flood state in some area), searches for published interaction models by sending a keyword-based query to the DDS which in turn collects them by matching the query and sends back the list to the peer. It then selects an interaction and subscribes to one of its roles on the DDS. When, in a given interaction, all the roles are subscribed by at least one peer, a compatible team is formed and the interaction is executed.

Since interaction models and peers are designed by possibly different entities, the constraints and the peers' capabilities may not correspond perfectly. The way this heterogeneity problem is tackled by the peers is by comparing the constraints in the received interaction models with their own capabilities. Such capabilities are provided by plug-in components, called OpenKnowledge Components (OKC) and consist of a set of Java methods. The comparison process results in a confidence level parameter that reflects how well the peer can execute an interaction and is used to select the most fitting one (Giunchiglia, McNeill, Yatskevich, Pane, Besana and Shvaiko, 2008).

Figure 1 shows a snapshot of a network, when the roles in interaction IM1 are all subscribed by at least one peer. The peers have installed their OKCs locally: some of them can be found online (e.g., OKC_1, OKC_2 and OKC_3), others might be private to a peer (e.g., OKC_4).

THE E-RESPONSE CASE STUDY

We applied the OK framework in a case study involving emergency response coordination activities. Our case study regards the evolution of a flooding event in Trento (Italy) and is based on the current flood emergency plan and interviews with experts. We identified *emergency agents* (e.g., fire-fighters, policemen, 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) 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.

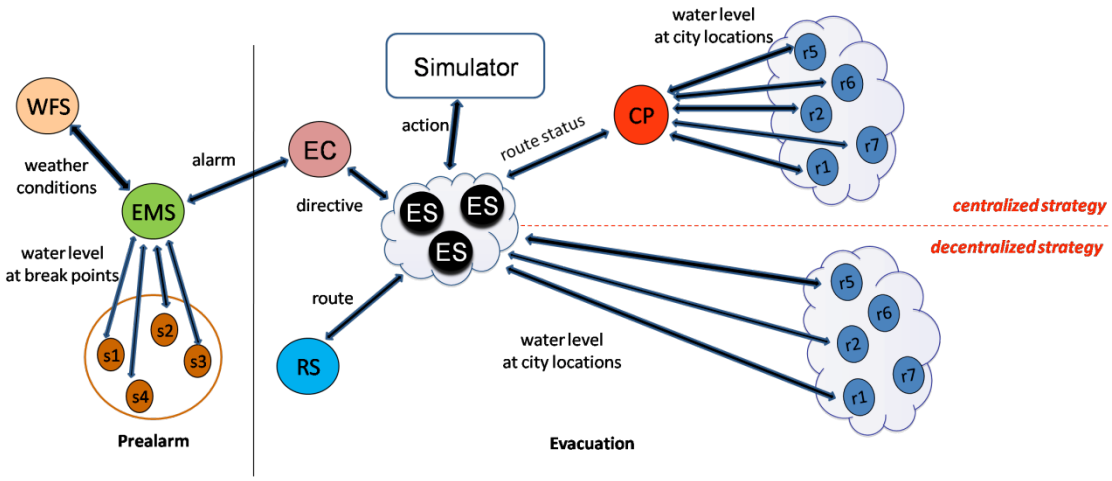


Figure 2: e-Response case study

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.

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.

As Figure 2 shows, the following agents are considered for the evacuation phase:

- *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) which needs to move to a specific location;
- *Route Service (RS)*: a service that provides routes connecting two given locations and excluding a set of "undesired" locations;
- *Civil Protection (CP)*: in our simulation, this agency is able to serve requests on the blockage state of a given route, since it continuously gathers information from reporters scattered around the emergency area and reporting the water level registered at their locations.
- *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 figure also depicts two different evacuation sub-scenarios: in both of them, an agent needs to get information on route's practicability but while in one case an emergency subordinate *ES* gets route information by asking the *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 interactions between the described agents are given in the next section.

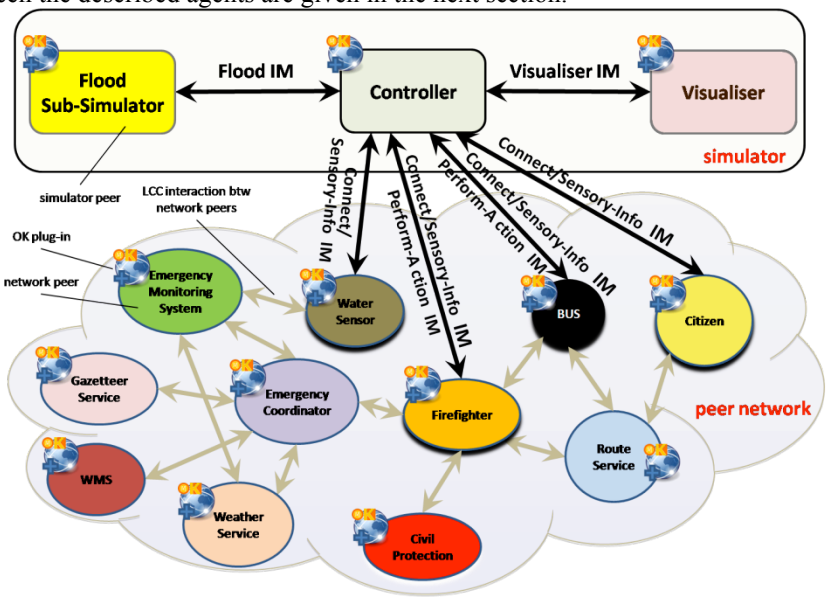


Figure 3: The e-Response simulation environment architecture

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 OK kernel. The present simulation environment is based on the system presented in (Marchese, Vaccari, Trecarichi, Osman and McNeill, 2008) and extends it both in a complete integration with the OK 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 has two main components: the peer network and the e-Response simulator.

Figure 3 sketches its overall architecture. All peers are equipped with their specific OK 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 "perform-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.

The interactions among the agents in the peer network, 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 and 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,Fron) 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(Fron,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 "perform-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 so to share the initial topology (e.g., point of interests and roads) 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 Trento town. The flooding law defined in its core OKC is based on flooding levels and timings described in (Alkema, Cavallin, De Amicis and Zanchi, 2001). To the purpose of our test-bed, the territory is divided into flooded areas, each stored in a geographical database and characterised by the maximum water height `MaxWL` reached during the inundation and the time `MaxT` when this level is touched.

The flood sub-simulator is developed in Java and is fully integrated into the OK kernel. Its associated OK peer subscribes to two interaction models: with the first one, the flood sub-simulator gets the initial topology, performs a spatial query in order to acquire, for each node, the parameters (`MaxWl` and `MaxT`) of the flooded area of pertinence and, finally, acknowledge the controller; with the second one, it sends the flood level changes in the nodes of the topology to the controller. Such changes are computed by accessing the information previously acquired and applying the flooding law, which foresees the absence of water until one hour before the time `MaxT`.

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 accordingly. The update results in a change on the GUI. Figure 5 shows the appearance of the GUI in two simulated scenarios.

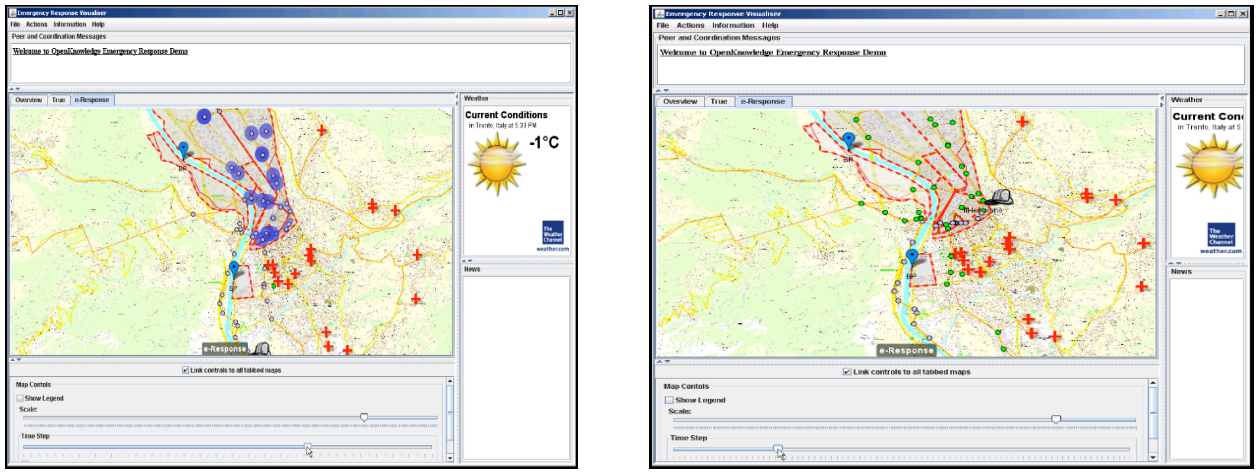
A *green dot* represents a reporter agent available for giving information on the water level registered; a *grey dot* represents a reporter agent giving this information; the water level at a location is depicted as a blue circle, which size depends on how high the water level is; the *hat* represents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



a) Simulation of centralized information gathering

b) Simulation on decentralized information gathering

Figure 5: Emergency GUI

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 (e.g., failures in communication, inaccurate signalling) is assumed. In order to interpret appropriately the obtained results, it is important to recall the assumptions made:

I. The 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^o	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.

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 CP; 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 which are located along the route taken by the moving agent and, therefore, queried by it; 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 6 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 CP or reporters scattered around the city does not make the difference.

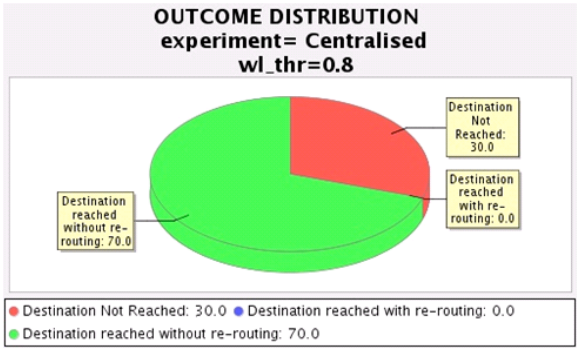


Figure 6: Outcome Distribution (centralized scenario)

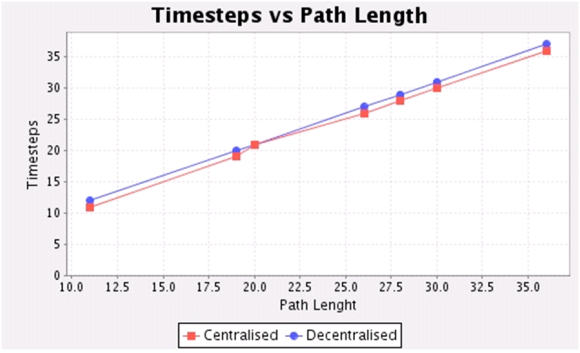


Figure 7: Time-steps vs. Path Length

Figure 7 shows the time (measured as the number of simulation time-steps) taken 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 7 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 OK 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. In particular, two types of fault conditions (i.e., disruption of communication channels and inaccurate signaling) will be simulated by introducing the following variables:

- *Distribution of trustworthy (reporter) peers*: the number of reporter peers having a trustworthy behaviour, i.e., always reporting accurate water level values;
- *Degradation of the CP communication channel*: the likelihood of a fault in the communication channel of the CP agent;
- *Degradation of reporter communication channels*: for all reporter channels, the probability of their disruption.

By running these new experiments, we want to investigate 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. 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.
3. 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.
4. 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.
5. 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.,*
6. 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.
7. 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.
8. 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
9. 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.
10. 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.
11. Robertson, D. (2004-1) A lightweight coordination calculus for agent systems. *Declarative Agent Languages and Technologies*, 183-197.
12. 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.
13. Trecarichi, G., Rizzi, V., Vaccari, L. 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.*