Crisis Management

Pervasive Software Environments for Supporting Disaster Responses

In complex emergency scenarios, teams from various emergency-response organizations must collaborate. These teams include both first responders, such as police and fire departments, and those operators who coordinate the effort from operational centers. The Workpad architecture consists of a front- and a back-end layer. The front-end layer is composed of several front-end teams of first responders, and the back-end layer is an integrated peer-to-peer network that lets front-end teams collaborate through information exchange and coordination. Team members at the front end carry PDAs, with team leaders' PDAs equipped with gateway communication technologies that let them communicate with the back-end centers.

Tiziana Catarci, Massimiliano de Leoni, Andrea Marrella, and Massimo Mecella
Sapienza–Università di Roma

Berardino Salvatore and Guido Vetere
IBM Italia

Schahram Dustdar, Lukasz Juszczyk, Atif Manzoor, and Hong-Linh Truong
Vienna University of Technology

Emergency or crisis management includes the coordinated activities involved in preparing, supporting, and rebuilding society when natural or human-made disasters occur. Such activities consist of five phases: planning, mitigation, preparedness, response, and recovery. The European Workpad project (www.workpad-project.eu) aims to provide a software and communication infrastructure to support operators facing an emergency. The project focuses on the response and short-term recovery phases — the most critical phases in disaster management — although many of our results are applicable to other phases. Response-phase activities include assisting victims, stabilizing the situation, speeding recovery actions, and reducing the probability of secondary damage. Short-term recovery activities include returning vital life-support systems to a minimum operating standard.

To devise successful information, communication, and media technology (ICMT) architectures for emergency management, Workpad employs user-centered techniques from human–computer interaction paradigms.[1] User-centered design relies on continuous interaction with end users to understand how organizations are arranged during disasters, what information is critical, and how teams exchange this information among themselves and with their operational centers. Our application of user-centered techniques, in collaboration with the Civil Protec-

 1089-7801/08/$25.00

tion of Calabria, Italy, involved interviewing officers and generic actors from the organizations most critical to emergency management in that region. We also studied the emergency-management structures of different European countries and found that most have emergency-management structures similar to Italy's (see the "Emergency Management in Italy" sidebar on p. 33).

This article focuses on the Workpad architecture's most innovative components — that is, its front- and back-end structures. The "Communication in Emergencies" sidebar on p. 35 describes Workpad's use of specific communication technologies.

The Workpad Architecture

Based on our understanding of how Civil Protection works in Italy and other countries during an emergency, as well as other researchers' results and the collected user requirements,[2] we identified two user typologies: back-end and front-end users. Front-end users are the operators acting directly in the field during disasters (ranging from fire fighters to voluntary associations). Back-end users are the operators who manage the situation from control rooms, providing instructions and information to front-end operators.

Figure 1 shows the Workpad architecture. Several teams comprise the system's front end. Team members belong to the same organization (for example, police or fire departments) and carry mobile devices (such as PDAs and smart phones). They establish a mobile ad hoc network (manet) for coordination and intrateam communication. In a manet, nodes can communicate with each other without an underlying infrastructure. Manet nodes communicate with their neighbors (that is, nodes in radio range) directly via wireless links. Nonneighbor nodes can communicate using intermediate nodes as relays that forward packets toward destinations. All nodes maintain routing tables so they can identify usable paths for forwarding data packets. The lack of a fixed infrastructure makes this kind of network suitable in emergency-management scenarios, in which users must quickly deploy a network but aren't guaranteed access points. In addition, because the available bandwidth (roughly 11 Mbps) is sufficient, manets can guarantee a good QoS level.

The Workpad back end is a peer-to-peer (P2P) overlay network that includes the operating organization's back-office systems (such as services and databases). By entering the Workpad network, back-end peers can easily integrate their data, content, and knowledge. Front-end operators access the back-end network through their back-office systems. There, they can get or set information that's relevant to their situation or planned action. Because of the integration layer, such information isn't necessarily contained in single systems, but is potentially spread over the network and is delivered, collected, and reconciled on demand.

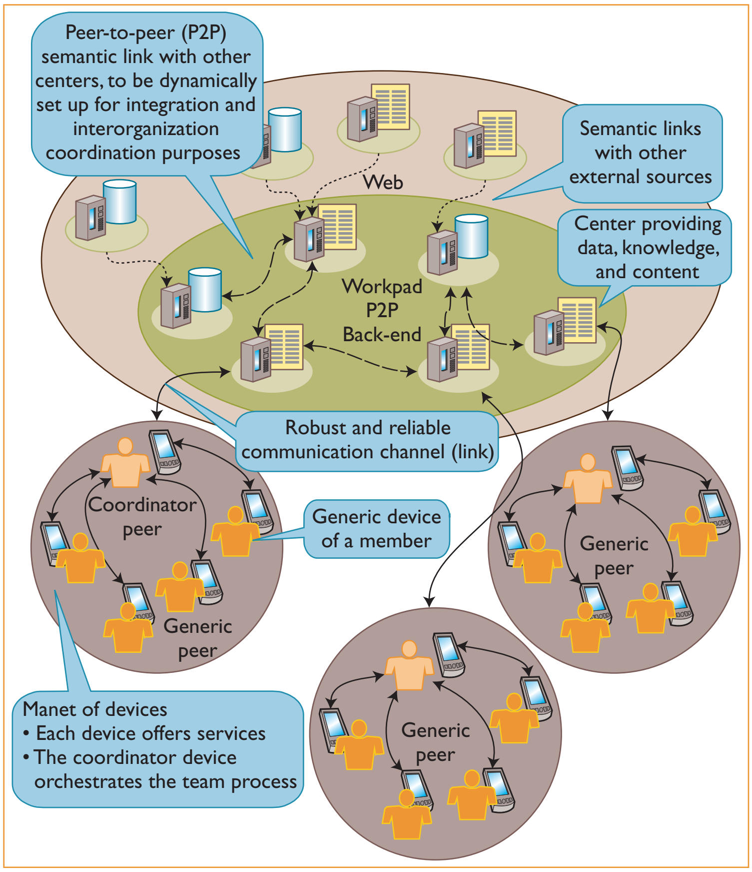

Figure 1. Workpad architecture. Workpad consists of front-end emergency-management teams and back-end control rooms. Back-end centers typically communicate directly with the team leader over available technologies, whereas team members communicate through a mobile ad hoc network (manet).

Not all devices can be equipped with costly and reliable technologies for communicating between the front-end teams and the back end (see the "Communications in Emergencies" sidebar). Instead, in Workpad, all devices can communicate with each other through the manet, while a few of them (the team leader's device and a few backups) act as gateways to the back

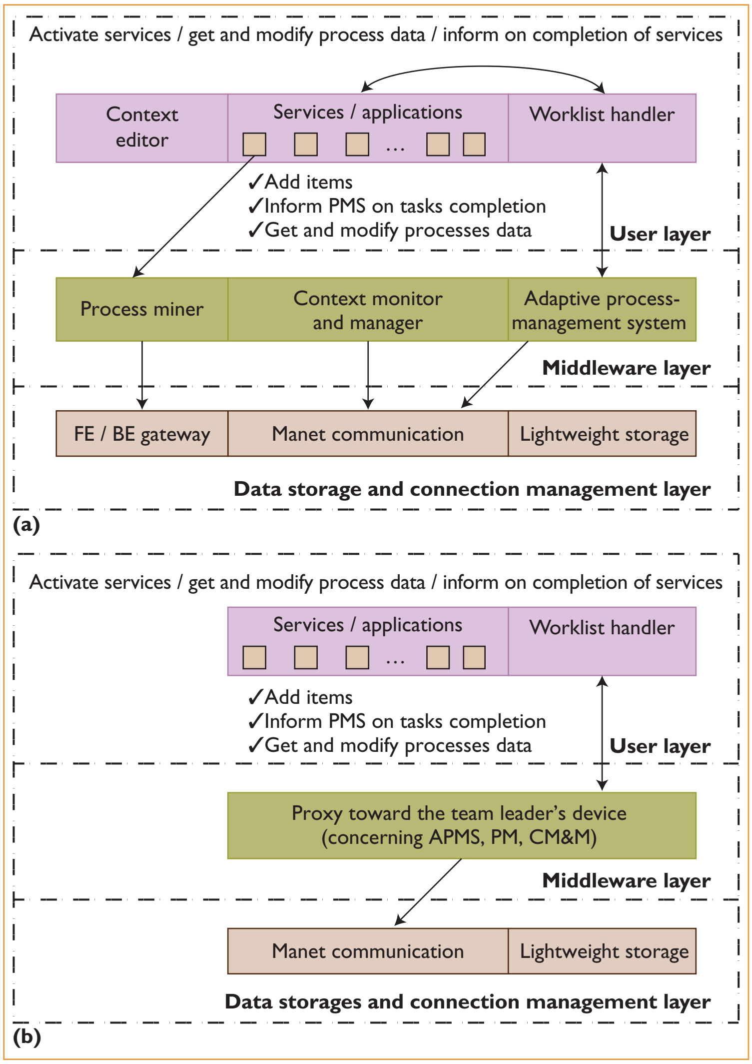

Figure 2. Overview of components of the Workpad front end. (a) The team leader's device, and (b) a team member's device. All devices contain modules for manet communication and lightweight storage. Team leaders' devices also include a gateway device that lets them communicate with back-end operators.

end. We can then equip the gateway devices with various technologies. (Workpad is currently experimenting with satellite, the Universal Mobile Telecommunications System [UMTS], and Terrestrial Trunked Radio [Tetra].) Devices can switch among these technologies depending on their availability.

Workpad Front End

Figure 2 shows the conceptual architecture at the Workpad front end. The front end consists of a data-storage and connection-management layer, a middleware layer, and the user layer. Each layer has several components. We haven't deployed all the components shown in the architecture in every front-end device. Instead, we customize their deployment, depending on devices' capabilities and the role of the team member controlling the device.

The data-storage and connection-management layer includes two modules:

- the *manet communication module*, which implements manet multihop communication; and
- the *lightweight storage module* for data and knowledge storing (either local or distributed).

Current operating systems don't allow communication among nonneighboring wireless peers, so we need a specific software module that implements one (or more) manet algorithms.[3] A few devices (including the team leader's) also include a front-end/back-end gateway to handle connections with the back end.

The *adaptive process-management system* (APMS) is the core element of the front-end middleware. It adaptively controls emergency-management processes based on contextual information retrieved by the context monitor and manager. (We describe this component in greater detail later.) This contextual information is associated with devices, networks, team members, activities, and so on. A process miner detects workflow patterns, individual member and team social behavior, and possible correlations.[4] The middleware layer at generic nodes is simpler, consisting only of specific modules whose purpose is to interact with the team leader's counterparts.

When the APMS assigns a task to an actor, it inserts the task into that actor's worklist handler (one for each device). Users learn their assigned tasks by querying this list. When they're ready to perform a given task, they pick the corresponding item in the worklist together with data needed for its execution. The handler knows which skill (service) is required for its execution, and, according to the required service, the handler runs the corresponding application to provide the service. When the application is closed, modified data returns to the handler with a notification of task completion. The handler forwards the data and termination to the APMS. The context editor component lets users enter additional contextual information that the front-end middleware couldn't capture.

Specific services offered by the devices are deployed based on the corresponding team members' capabilities and skills.

Example Application

When an emergency occurs, the back end sends emergency information to the team leader. It loads this information, which includes a description of the disaster, goals, and geographic information, onto the team leader's PDA (Figure 3a). When the team leader presses the "see geographic information" button, a map of the stricken place opens, showing the PDA owner's current position. To determine which members are best qualified to face the emergency, the team leader can perform a search that returns a list of actors who are ready to intervene at that moment, along with their personal capabilities (Figure 3b). Alternatively, the system can propose a team configuration based on available services deployed on the reachable team members and automatically discovered.[5] Finally, the leader defines the process schema by customizing predefined generic process templates provided by the GUI (not described in this article due to space limitations),[6] and moves the process schema to the APMS as input together with the initial context.

From our collection of user requirements, we learned that, for the most part, emergency-management operators define the processes to be enacted starting from predefined templates, and later instantiate the processes to the specific context. That is, there exist specific well-known practices that are generically applicable.

The APMS automatically assigns the tasks whose conditions are fulfilled to the actors able to execute them (because each task requires a well-defined set of capabilities), and each client's worklist handler receives notification of the assigned tasks (Figure 3c). Next, the actor picks a task from his or her worklist, and the worklist handler starts the applications needed to perform the task. (For example, in Figure 3d, the task consists of taking a photo.) After executing a task, an actor's system automatically alerts the team leader's PDA, which has a complete view of the situation (Figure 3e). At each moment, the APMS can analyze whether new tasks are assignable and, if so, assign these tasks to the device whose capabilities best match the situation.

If an actor becomes disconnected due to movement, another PDA can act as a bridge. The system automatically assigns a PDA to follow the disconnected node to keep alive a multihop path among devices, and a popup notifies the corresponding operator.

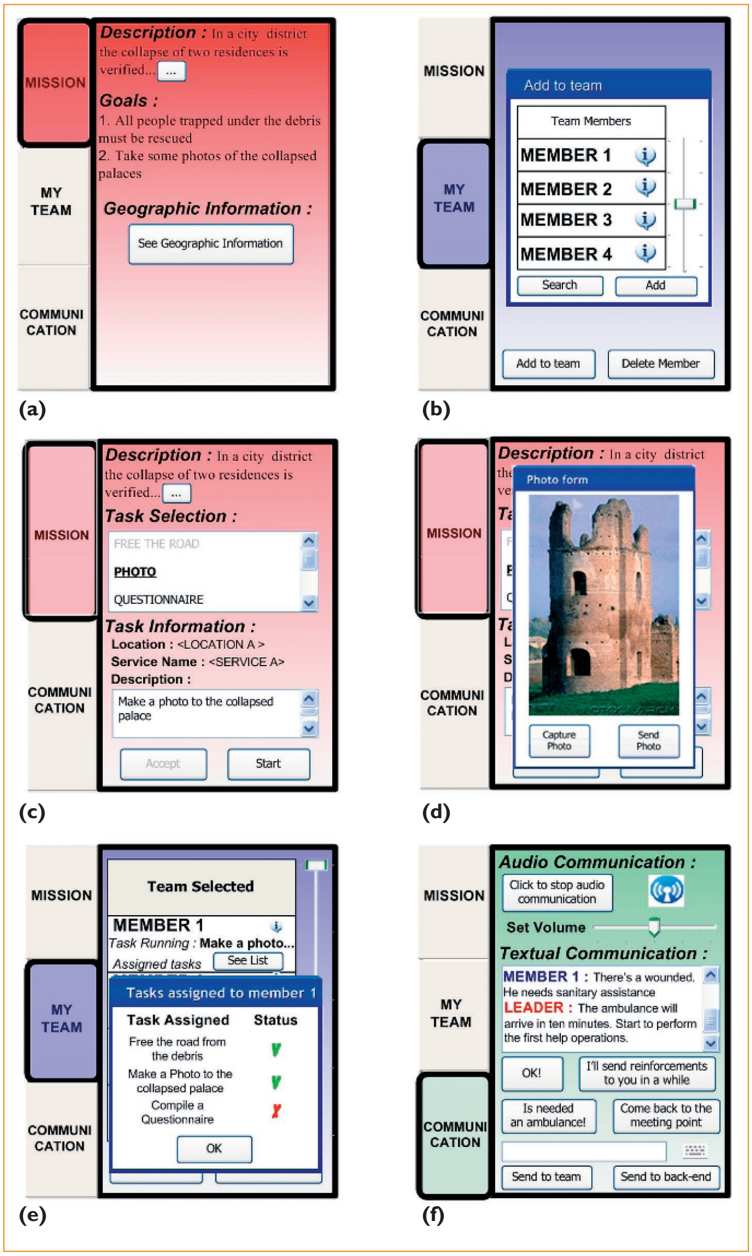

Figure 3. Workpad's front-end GUI. The GUI provides several types of information, depending on the device owner. (a) The team leader's device, showing details of the event; (b) a list of team members and their capabilities on the team leader's device; (c) a description of a task assigned to a specific team member; (d) an example task in which the team member is to take a photograph of an affected building; (e) the team leader's view of a task's status; and (f) the communication mechanism.

Finally, each PDA is equipped with a com-

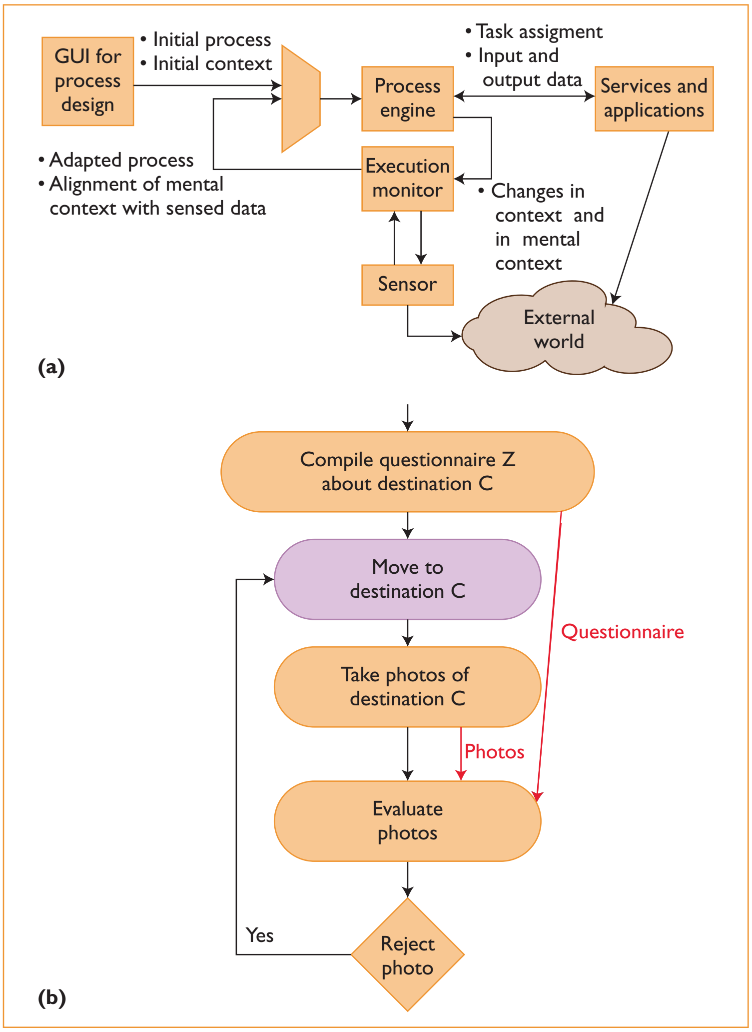

Figure 4. Workpad's adaptive process-management system. (a) The APMS architecture. Sensors process raw data from applications, video and audio streams, and other sources to find relevant events. (b) An example template for a building-assessment process.

munication mechanism (Figure 3f). The system exploits the manet technology to let team members interact through audio communication. In this way, the PDA also acts as a transceiver. If audio communication doesn't work (that is, the network is overloaded), the system provides the user with an alternative and lighter way to communicate — that is, exchange messages in a text form. As Figure 3f shows, users can add text to the form using a set of predefined messages or a virtual keyboard, as in a chat.

Adaptive Process-Management System

Many businesses and government agencies use process-management systems (PMSs).[7] Such scenarios present mainly static characteristics (that is, deviations are the exception). PMSs can also be useful in mobile and highly dynamic situations, for coordinating operators, devices, robots, and sensors in emergency situations.[8]

In the Workpad project, teams of emergency operators use PMSs to coordinate their activities. For example, to perform their activities, the devices in the manet must be continuously connected. But this isn't guaranteed — the environment is highly dynamic because nodes (that is, devices and the related operators) move in the affected area to perform assigned tasks. Movements can cause disconnections and, hence, node unavailability. Therefore, the PMS should adapt the process. Adaptiveness might consist of simply assigning the task in progress to another device, but our observations of actual user requirements show that most teams are formed by a few nodes (less than 10), and therefore such reassignment often isn't feasible. Conversely, we can envision other kinds of adaptiveness, such as recovering the disconnecting node through specific tasks — for example, when X is disconnecting, the PMS could assign the "follow X" task to another node to guarantee the connection. So, in such scenarios, the process is designed (and deployed on the PMS) as if no problems will occur during runtime, and it must be continuously adapted based on rules that aren't foreseeable at design time.

Process schemas describe the different aspects of a PMS — that is, tasks or activities and control and data flow. Every task has an associated set of conditions that must be true to perform the task. Conditions are defined on control and data flow (a previous task must be finished, a variable must be assigned a specific range of values, and so on). We can consider such conditions internal — that is, the PMS handles them internally, so they're easily controllable. External conditions, on the other hand, depend on the environment in which process instances occur. These conditions are more difficult to control, so require continuous monitoring to detect discrepancies.

Indeed, we can distinguish between two types of reality:

- *external* reality is the conditions' actual value, and
- *internal* reality is the model of reality that the PMS uses in making deliberations.[9]

The PMS builds the internal reality by assuming that tasks and actions fill expectations (that is, they correctly modify conditions) and that no exogenous events that can modify conditions will break out. Still, one or more events could occur that cause the two types of reality to deviate. In this case, the PMS could ignore the deviations. However, this is generally infeasible because the new situation might be such that the PMS can no longer perform the process instance. To protect against such a situation, then, the process schema can include actions to cope with such failures. This is feasible in static contexts (and indeed is the most common technique in many other contexts) but difficult in emergency-management scenarios because defining all possible discrepancies is difficult. Alternatively, a general recovery method that can handle any kind of exogenous events could be devised. This is the approach taken with the Workpad APMS (see Figure 4a).

After each action, the PMS must align the internal representation with the external one, which could differ due to unforeseen events. Before the PMS starts to execute a process, it takes the initial context from the real environment as the initial situation, together with the process to be performed. For each execution step, the PMS, which has complete knowledge of the internal reality, assigns a task to a service. The only assignable tasks are those whose preconditions are fulfilled. A service can collect from the PMS the data required to execute the task. When a service finishes executing the task, it alerts the PMS. The monitor can interrupt the PMS's execution when it senses a misalignment between the internal and external realities. When this happens, the monitor adapts the process to deal with the discrepancy. The APMS realizes the concepts we've described here using novel techniques based on situation calculus and ConGolog.[10]

In this article, a sensor is any software or hardware component that gets contextual information from the external reality. The external world provides sensors with information through raw data sources, called *e-chronicles*. E-chronicles typically represent logs from applications, video or audio streams (for example, from cameras in the environment), and many other sources. Workpad sensors serve as modules in charge of processing these huge raw data streams to discover relevant events. Relevancy, of course, depends on the model associated with the specific data sources. The sensors, while keeping all records, also signal relevant events and aggregate data at the desired level of granularity (similarly to the SATWare framework).[11]

For example, assume we have a predefined template for a building assessment, as Figure 4b depicts. Our process is roughly parallel to three instances of the template on buildings A, B, and C. To show the difference between the external and internal reality, consider the "move to destination C" step. In the internal reality, at the end of the task's execution, the actor performing the task is close to destination C. But in the external reality, the task's execution could lead the actor away from that position. The system retrieves the actor's GPS information as well as the GPS positions sent by generic peers. If this information shows that the actor isn't close to the destination, the system could assume that the node can't go closer, so it assigns the task to another node. The external reality alerts the system that the task finished prematurely when the position was Y because the node was going to disconnect from all other nodes. Then, the APMS creates a "go to Y" task and assigns it to another node to let the former node continue to move toward building C while staying connected. Decisions as to how to realign realities aren't made manually by operators, but automatically by the APMS.

Context Manager and Process Miner

Two types of information substantially impact a team's ability to work collaboratively in responding to a disaster: context information and service–human interaction patterns. Context information is common in context-aware systems, but few of these systems are dedicated to disaster scenarios. Understanding the context of disaster scenarios is critical to the success of disaster responses.

We've studied all context information relevant to the entities inherent in disaster scenarios. Such entities are either affected by the disaster, participate in the disaster, or are used in the response. Our approach in studying context information in disaster scenarios is to determine entities according to the five classes that Renato Bulcão Neto and Maria da Graça Campos Pimentel proposed[12] — that is, we answer the following questions:

- Who's involved in disaster scenarios?

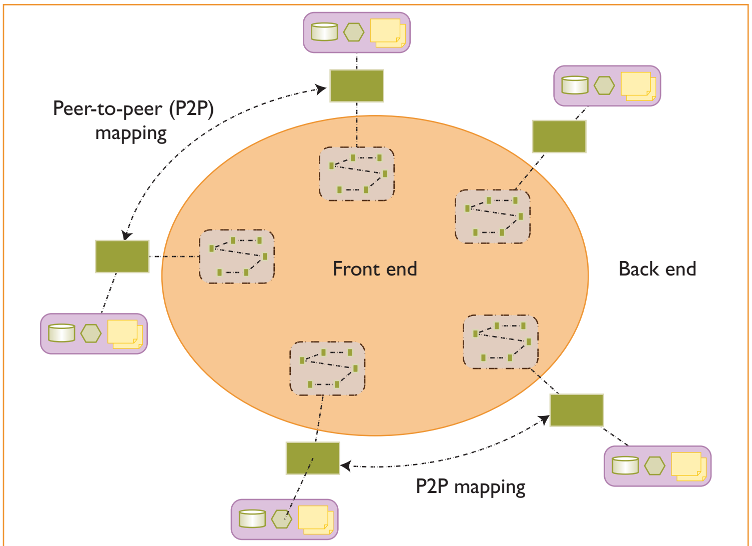

Figure 5. Back-end peers form an overlay network. Administrators use Workpad management tools to define organizational ontologies and data sources and specify conceptual mappings between ontologies.

- Where are the entities in disaster scenarios?
- When does an activity's status change?
- What activities were done, are being done, and will be done?
- What is the entities' status and how capable are their profiles?

Answering these questions will help us design a model describing context information in disasters. We implemented such a model in the context manager using a suitable ontology that consists of concepts and relationships describing context information based on the five questions outlined above. Moreover, the ontology supports hierarchical views of context information, as Christoph Dorn and Schahram Dustdar propose.[13]

Processes defined during disaster responses are based on various kinds of human interactions in disaster scenarios. It's important to study and optimize these interactions because humans, not technologies, play the key role in disaster response. Although context information is important for defining processes conducted during the disaster response, humans establish these processes based on information available at a disaster site. Given the time-constrained aspect of disaster response, errors can occur in such process establishments. In addition, actors might want to know whether the tasks planned are optimal or not, or if frequent patterns (such as work delay and failure) occurred and were associated with existing members, and subsequently affected the entire response process's performance. The operator could answer such questions at runtime or through *process mining*.[14]

We distinguish between two types of process mining. Online mining aims to discover abnormal behaviors as soon as possible during the disaster response so the system can reallocate tasks, resulting in a better plan. In online mining, the process miner supports the APMS by detecting failures and various types of violations based on predefined constraints. In offline mining, the operator conducts advanced analyses that need more time and computational power. In this mode, we focus on the analysis and comparison of the overall processes' performance and the optimization of these processes based on detected performance problems and failures. This way, we can learn many lessons from previous response processes and use them to define increasingly precise process templates.

Workpad Back End

Workpad front-end networks are connected to specific back-end systems, which include a Web services platform to allow data exchange and integration. This platform is designed as a P2P network, in which each system (peer) can act as data provider, consumer, and integrator.

By plugging into Workpad's back-end network, a back-office system qualifies as a Workpad back-end peer. This peer exports its ontology (that is, a schema reflecting its conceptual model); allows a rapid integration of various data sources, both internal and external (including other peers), through mappings from available sources to the ontology; and can answer conjunctive queries expressed in the alphabet of ontology terms. Front-end services or other back-end systems can issue these queries. A peer can also receive notifications of relevant updates in other systems in the network.

By mapping their ontologies onto one another, peer systems achieve semantic integration without resorting to global ontologies. Obviously, peers can also conform to shared conceptualizations. Moreover, the more peers agree on shared conceptualizations, the fewer mappings between ontologies are needed. The ultimate goal, however, is to relieve peers from having to adopt large and complex shared ontologies,

Emergency Management in Italy

In Italy, *Law 225* (24 February 1992) regulates disaster response. This law establishes Civil Protection as a national service coordinated by the premier. Civil Protection consists of central and local authorities, and includes public corps and voluntary institutions in the national territory.

At the regional level, the Civil Protection has access to an operational control hall — Sala Operativa Unificata Regionale (Regional Unified Room for Operations). SOUR's activities focus on controlling possible alerts (for example, from a hydrometeorological system); receiving, checking, and managing events; and issuing alert messages. SOUR is constantly in contact with the national Civil Protection Department and, when a calamitous event occurs, with the affected areas' prefectures, providing logistic and informative support.

When a disaster breaks out, the province in which the disaster occurs coordinates the required activities (in Italy, each region consists of several provinces). These tasks include activating the Centro Coordinamento Soccorsi (Center for Coordination of Aids), which represents the Civil Protection's strategic-operational top line at this level (one CCS exists in each province). The prefect coordinates the CCS, which consists of officers from several organizations (police, fire brigade, and so on) involved in emergency management. Primary CCS tasks are inspection, collection, and elaboration of data and information concerning the evolving situation, and coordination of all activities performed by Centri Operativi Misti (Mixed Operational Centers).

A COM is an operative decentralized structure that depends on the CCS. Provinces can have several COMs arranged in a capillary way — for example, the province of Reggio Calabria has 19 COMs. A COM is closer to the disaster and thus immediately acknowledges local demands and organizes the required work. COMs should return results to the CCS because the CCS needs a complete and updated scenario and can coordinate several COMs' work. COM organizations communicate with their respective control rooms through radio frequencies or, if unavailable, by telephone and fax. If an organization doesn't have a control room, COMs communicate directly with front-end teams by telephone or fax (the same is true for all other communications, such as between SOUR and the CCSs, and between CCSs and COMs).

thus facilitating their rapid integration. The integration and involvement of a particular peer is dynamically and adaptively decided on the basis of the specific process to be put into effect. This process, in turn, depends on the given emergency situation. The integration logic will therefore be distributed throughout the entire system, so peers will need no specific integration systems beforehand.

Back-End Peers

Workpad peers are systems that perform ontology-based knowledge retrieval over a set of local sources and a specific overlay network. Local sources can be relational database management systems, Web services, XML documents, and so on. Workpad also gives knowledge administrators a set of tools for managing and controlling peers. These tools provide utilities that let administrators define organizational ontologies and data sources, specify conceptual mappings between ontologies, and so on (see Figure 5).

We organize peer functionalities in three layers: modeling, query and subscription, and information and reasoning (see Figure 6).

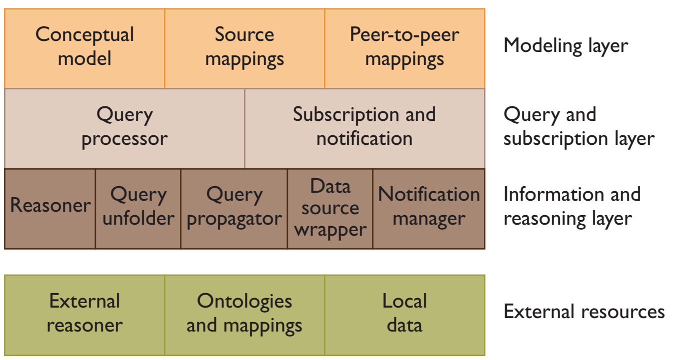

Figure 6. Conceptual architecture of a back-end peer. Peer functionalities can be categorized as modeling, query and subscription, and information and reasoning. The final layer consists of external resources that are relevant to the peer.

Modeling layer. The modeling layer comprises the functionalities that let knowledge engineers define peer ontologies, map data sources (local data) onto these ontologies, and specify conceptual mappings between locally defined ontologies and those defined by other peers. We built all of these functionalities over a rich client platform based on the Eclipse technology.

Query and subscription layer. This layer manages conjunctive queries (that is, a restricted form of first-order queries) and subscription requests. It also supports notifications of data and knowledge changes. Clients (front-end services or external systems) will be able to subscribe to any conjunctive query so they can be notified when a query's extension changes (that is, when new records are added, deleted, or changed), provided that the client supports callback functionalities compliant with the WS-Notification specification. The subscription and notification component will handle these functionalities.

Information and reasoning layer. At the third layer, clients' concrete queries are propagated to other peers or executed toward local data. The *query propagator* and the *data source wrapper*, respectively, perform these two functions. The *notification manager* receives a message each time the local data are modified. This message triggers the upper layer's subscription and notification component, which consequently notifies all subscribed end points for queries that are impacted by such changes. Any application that changes the local data must promptly send a message to the notification manager. The *query unfolder* reformulates the input query in terms of concrete queries toward local data sources and conjunctive queries toward related peers. The *reasoner* provides the necessary reasoning capabilities for reformulating queries.

External resources layer. The final layer contains all the external resources that are relevant to the back-end peer. These resources consist of the external reasoners, the materialization of the ontologies in suitable databases, and the mappings and local sources (or local data).

P2P Information Integration

Workpad's back end is a flexible data-integration infrastructure that supports P2P networking, in addition to classic mediator-based architectures. In general, data integration involves combining data from different sources and providing the user with a unified and consistent view of that data. Most current data-integration platforms are based on *centralized semantic mediation* — that is, a single system provides a mediated schema (a global view) over distributed data sources and performs sound and complete query answering over them. Semantic mediators take full responsibility for interpreting data sources because they're generally defined within the same epistemic boundaries. So, the system acts as if it's a single database in which query answering is given standard first-order semantics. On the other hand, P2P data integration combines autonomous systems and so must answer queries using possible-world semantics. From a single peer's viewpoint, the key issue is how to use information gathered by other peers.

One possibility is to adopt a global semantics-based approach. In this case, Workpad peers could draw mappings between their concepts and others'. A mapping M between two peers A and B is constituted by a set of assertions. For example, $q_A \rightarrow q_B$, where q_A and q_B are two conjunctive queries of the same arity, over the ontology of A and the ontology of B, respectively. Researchers have extensively studied and experimented with this type of pure P2P system. Hyper, for instance, takes each peer as an independent knowledge base and lets a single peer import knowledge from another peer under specific epistemic conditions valid in the entire P2P system's context.[15]

In general, a query to a P2P networking system must account for the epistemic difference between what the queried system knows (that is, actual data) and what it knows the others know (that is, information collected on the network). In other words, it must distinguish between the knowledge of facts and the knowledge of what is referred. We propose a *doxastic approach* such that peers don't transfer their knowledge of facts to others. Instead, for a specific peer A, the knowledge of another peer B becomes knowledge of that peer's opinion. (We're currently developing a complete theory of doxastic P2P integration.)

Intuitively, whereas an epistemic approach aims to model knowledge of facts with respect to many possible states of affairs, a doxastic approach separates facts from opinions, giving the latter a specific ontological status. Epistemic systems such as Hyper require peers to deal with inconsistencies. For example, contrasting knowledge (such as $Damaged(e)$ at peer A, and $\neg Damaged(e)$ at peer B), if not repaired, would lead to global uncertainty about the item referred to. For this reason, a doxastic approach seems to be appropriate in situations in which inconsistencies might easily arise and can't be easily repaired. Actually, our approach directly leads to the consequence that knowledge of facts by peer A can't be inconsistent with knowledge of the same facts by peer B because the knowledge of these two peers is formally separated.

Starting from a doxastic approach, Workpad uses modal queries with a special syntax to request beliefs of the queried peer. The peer

Communication in Emergencies

In emergencies, different communication requirements arise at the front and back ends. At the front end, in addition to voice-based communications (transceivers and mobile phones), emergency-management teams need data-based communication systems that can be easily deployed even if preexisting infrastructures are damaged or nonexistent. Therefore, as B.S. Manoj and Alexandra Baker discuss,[1] solutions based on mesh and ad hoc networks are viable ones. In these systems, the front-end network is easily deployed and provides a data connection to operators with discrete quality of service. Of course, if preexisting infrastructures (such as WiFi access points deployed on buildings or street lamps) in the area have survived, responders can exploit them as well, possibly using the system to adaptively switch to the channel with the highest QoS (see, for example, the PICO system[2]).

If front-end nodes need access to the back end and no direct communication infrastructures (such as WiFi Internet access points) exist, a few devices will act as gateways. Such gateways are equipped with technologies to communicate with the back end through the best performing available channel.

Several technologies for front- to back-end links exist. First, satellite communication systems are becoming more readily available. However, satellites are still a costly solution, and can't be deployed on all front-end devices. Satellite solutions guarantee high bandwidth and 100 percent coverage across a wide area. Therefore, they're used only if other connections are unavailable in the gateway nodes.

Another technology, the Universal Mobile Telecommunication System (UMTS), represents an evolution in terms of capacity, data speed, and service capabilities from second-generation mobile networks. More than 60 third-generation UMTS networks are now operating commercially in 25 European countries. However, UMTS isn't always available in emergency situations because UMTS infrastructures can be easily damaged along with civil buildings.

Finally, Terrestrial Trunked Radio is a European Telecommunications Standards Institute (www.etsi.org) standard for building specific mobile networks. Tetra's low-frequency lets it achieve a high level of geographic coverage. Indeed, Tetra is specifically intended for police and fire brigades, the army, and so on, because Tetra relays are generally arranged so if some go down, others can cover most of the area (unlike technologies such as UMTS). Tetra's main disadvantage is its low data-transfer rate, which makes it difficult to transfer and exchange a great deal of data. But this isn't an issue, because front-end teams can use wireless links (such as manets) internally and revert to Tetra only when they need to access the back end and if other technologies (such as UMTS) are unavailable.

References

1. B.S. Manoj and A.H. Baker, "Communication Challenges in Emergency Response," *Comm. ACM*, vol. 50, no. 3, 2007, pp. 51–53.
2. M. Kumar et al., "PICO — A Middleware Framework for Pervasive Computing," *IEEE Pervasive Computing*, vol. 2, no. 3, 2003, pp. 72–79.

computes the result set of an epistemic query, appropriately querying directly connected peers, and returns the set to the client. This result set is an aggregation of facts (from its own data sources) and beliefs (from mapping other peers' result sets). Each belief will include information about which peers consider it a fact, thus tracking data provenance. The query issuer will then have the task of managing and composing potential conflicts between different opinions across the network.

Concept Languages and Reasoning

Workpad supports scenarios in which peers provide their own conceptualizations, based on its data sources. Workpad back-end systems will manifest their schemas by using an expressive concept language, possibly by mapping their existing schemas (for example, relational). For data-intensive applications such as Workpad applications, the trade-off between these languages' expressive power and the computational complexity of sound and complete reasoning is crucial. Specifically, the most relevant reasoning task for a peer is answering to conjunctive queries over the set of instances maintained in its secondary storage.

Current research often represents ontologies using formal languages from the description logics family. Rich description logics based on current Semantic Web standards (such as OWL) typically suffer from worst-case exponential time reasoning when applied to query-answering tasks. In general, borrowing from database theory, we can evaluate the query-answering problem's complexity with respect to the query's size (query complexity), the data's size (data complexity), or both (combined complexity). When a system must deal with a large amount of data, you should keep data complexity as low as possible. For this reason, ontologies used in Workpad will use a particular description logics subfamily of languages, called DL-Lite.[9]

DL-Lite is rich enough to capture most basic ontology languages, such as UML class diagrams, while keeping the reasoning complexity low. DL-Lite standard reasoning tasks are polynomial in the size of ontology terms (TBox), and query answering is polynomial in the size of individual assertions (ABox). Noticeably, query answering is polynomial (logspace) in data complexity — that is, it has the same complexity of traditional database management systems. Moreover, DL-Lite lets a system separate terminological from individual reasoning, so existing database-management systems can perform processes requiring data access using suitable query-rewriting processes.

Workpad's applicability goes beyond crisis management. Other scenarios (video surveillance, cooperation among workers at a building site, and so on) can also exploit the availability of process-management systems on mobile devices, supported by context awareness and process mining, and of semantically integrated data. In the future, we envision applying and extending Workpad to such new scenarios.

Acknowledgments

The European Commission supported this work through the FP6-2005-IST-5-034749 project, Workpad. We thank Calabria's Civil Protection for its full support in collecting user requirements, the other technical partners for many interesting discussions, and the anonymous reviewers for their useful comments.

References

1. A. Dix et al., *Human Computer Interaction*, 3rd ed., Prentice Hall, 2003.
2. M. de Leoni et al., "Emergency Management: From User Requirements to a Flexible P2P Architecture," *Proc. 4th Int'l Conf. Information Systems for Crisis Response and Management* (ISCRAM), 2007, pp. 271–279.
3. I.D. Chakeres and E.M. Belding-Royer, "AODV Routing Protocol Implementation Design," *Proc. Int'l Workshop Wireless Ad Hoc Networking* (WWAN), IEEE CS Press, 2004, pp. 698–703.
4. S. Dustdar and R. Gombotz, "Discovering Web Service Workflows Using Web Services Interaction Mining," *Int'l J. Business Process Integration and Management*, vol. 1, no. 4, 2006, pp. 256–266.
5. C.A. Flores-Cortés, G.S. Blair, and P. Grace, "An Adaptive Middleware to Overcome Service Discovery Heterogeneity in Mobile Ad Hoc Environments," *IEEE Distributed Systems Online*, vol. 8, no. 7, 2007, art. no. 0707-o7001; http://dsonline.computer.org/portal/pages/dsonline/2007/07/o7001.html
6. G. De Giacomo et al., "Automatic Workflows Composition of Mobile Services," *Proc. 2007 IEEE Int'l Conf. Web Services* (ICWS), IEEE CS Press, 2007, pp. 823–830.
7. W.M.P. van der Aalst and K. van Hee, *Workflow Management. Models, Methods, and Systems*, MIT Press, 2004.
8. F. De Rosa, A. Malizia, and M. Mecella, "Disconnection Prediction in Mobile Ad Hoc Networks for Supporting Cooperative Work," *IEEE Pervasive Computing*, vol. 4, no. 3, 2005, pp. 62–70.
9. D. Calvanese et al., "DL-Lite: Tractable Description Logics for Ontologies," *Proc. 20th Nat'l Conf. Artificial Intelligence* (AAAI), AAAI Press, 2005, pp. 602–607.
10. M. de Leoni, M. Mecella, and G. De Giacomo, "Highly Dynamic Adaptation in Process-management Systems through Execution Monitoring," *Proc. 5th Int'l Conf. Business Process-management* (BPM), LNCS 4714, Springer-Verlag, 2007, pp. 182–197.
11. B. Hore et al., "SATware: Middleware for Sentient Spaces," *Multimodal Surveillance: Sensors, Algorithms, and Systems*, ITR-Rescue, 2007.
12. R.F.B. Neto and M.G.C. Pimentel, "Toward a Domain-Independent Semantic Model for Context-Aware Computing," *Proc. 3rd Latin American Web Congress* (LA-Web), IEEE CS Press, 2005, pp. 61–70.
13. C. Dorn and S. Dustdar, "Sharing Hierarchical Context for Mobile Web Services," *Distributed and Parallel Databases*, vol. 21, no. 1, 2007, pp. 85-111.
14. W.M.P. van der Aalst and A.J.M.M. Weijters, "Process Mining," *Process-Aware Information Systems: Bridging People and Software through Process Technology*, M. Dumas et al., eds., John Wiley & Sons, 2005, pp. 235-256.
15. D. Calvanese et al., "Hyper: A Framework for Peer-to-Peer Data Integration on Grids," *Proc. Int'l Conf. Semantics of a Networked World*, LNCS 3226, Springer-Verlag, 2004, pp. 144–157.

Tiziana Catarci is a full professor in the Department of Systems and Computer Science and Engineering at Sapienza–Universitá di Roma. Her research interests include user interfaces for databases, 2D and 3D data visualization, adaptive interfaces, controlled studies and usability testing, query language expressive power, heterogeneous and cooperative information systems, data integration and Web access. Contact her at catarci@dis.uniroma1.it.

Massimiliano de Leoni is a PhD student in the Department of Systems and Computer Science and Engineering at Sapienza–Universitá di Roma. His research interests include business process-management, mobile and adaptive information systems, and wireless networking in service-oriented architectures. de Leoni has an MsC in computer science and engineering from Sapienza–Universitá di Roma. Contact him at deleoni@dis.uniroma1.it.

Andrea Marrella is an MsC student in computer science and engineering at Sapienza–Universitá di Roma. His research interests include service-oriented computing, business-process management, mobile systems, human-computer interaction, and user-centered design methods. Marrella has a bachelor's degree in computer science and engineering from Sapienza–Universitá di Roma. Contact him at marrella@dis.uniroma1.it.

Massimo Mecella is an assistant professor in the Department of Systems and Computer Science and Engineering at Sapienza–Universitá di Roma. His research interests include service-oriented computing, business process-management, cooperative systems, mobile and adaptive information systems, and middleware technologies. Mecella has a PhD in computer science and engineering from Sapienza–Universitá di Roma. He is a member of the IEEE and the IEEE Computer Society. Contact him at mecella@dis.uniroma1.it.

Berardino Salvatore is an advisory software engineer in IBM's Rome Software Laboratory. His activities and interests include semantic technologies, service-oriented architectures, and business architectures. Salvatore has an MsC in the physics of elementary particles from the Università di Pisa, Italy. He is a member of the IEEE. Contact him at salvatore@it.ibm.com.

Guido Vetere is manager and research coordinator at the IBM Center for Advanced Studies, Rome, Italy. His research interests include language technologies, knowledge management, information integration, semantic technologies, and ontologies. Vetere has a PhD in the philosophy of language from Sapienza–Universitá di Roma. Contact him at gvetere@it.ibm.com.

Schahram Dustdar is a full professor of computer science, director of the Vienna Internet Technologies Advanced Research Lab, head of the Distributed Systems Group of the Information Systems Institute at the Vienna University of Technology, and honorary professor of information systems in the Department of Computer Science at the University of Groningen, the Netherlands. His research interests include service-oriented architectures and computing, mobile and ubiquitous computing, complex and adaptive systems, and context-aware computing. Dustdar has a PhD in business informatics from the University of Linz, Austria. He is a member of the IEEE Computer Society and the ACM. Contact him at dustdar@infosys.tuwien.ac.at.

Lukasz Juszczyk is a PhD student in the Distributed System Group at the Information Systems Institute, Vienna University of Technology. His research interests include service-oriented architectures, Web services, and adaptive workflow management. Juszczyk has an MsC in computer science from the Vienna University of Technology. Contact him at ljuszczyk@infosys.tuwien.ac.at.

Atif Manzoor is a PhD student in the Distributed System Group at the Information Systems Institute, Vienna University of Technology. His research interests include context-aware systems and sensor data fusion techniques. Contact him at atif.manzoor@infosys.tuwien.ac.at.

Hong-Linh Truong is a postdoctoral research scientist in the Distributed Systems Group at the Institute of Information Systems, Vienna University of Technology. His research interests include performance monitoring and analysis techniques and tools, peer-to-peer and grid computing, Internet technologies, middleware, collaborative computing, workflow systems, and autonomic computing. Truong is a member of the ACM, the IEEE, and the IEEE Computer Society. Contact him at truong@infosys.tuwien.ac.at.