A Service-oriented Framework Supporting Ubiquitous Disaster Response

Michele Amoretti, Maria Chiara Laghi, and Gianni Conte

Information Engineering Department, University of Parma, 43100 Parma, Italy,
michele.amoretti@unipr.it,laghi@ce.unipr.it,gianni.conte@unipr.it,
WWW home page: http://dsg.ce.unipr.it

Abstract. The synergy of ubiquitous computing and service-oriented technologies may lead to efficient, pervasive and dependable solutions in the challenging context of emergency management. Recently, novel paradigms have been proposed, most of them envisioning arbitrary pairs of peer application entities communicating and providing services directly with each other and to users. In order to enforce these paradigms even to systems which include devices with limited processing and storage resources, lightweight middleware components are required. We illustrate how this is provided by JXTA-SOAP, a portable software component supporting peer-to-peer sharing of Web Services, and we show how it can be used to implement disaster response software applications.

Key words: mobility, disaster response, middleware, services, peer-to-peer

1 Introduction

Emergency management (or disaster management) is the discipline of dealing with and avoiding risks [13]. It involves preparing for disaster before it happens, disaster response (*e.g.* emergency evacuation, quarantine, mass decontamination, etc.), as well as supporting and rebuilding society after natural or human-made disasters have occurred. The disaster management cycle involves four key phases:

1. Mitigation: includes any activities that prevent a disaster, reduce the chance of a disaster happening, or reduce the damaging effects of unavoidable disasters.
2. Preparedness: includes plans or preparations made to save lives or property, and to help the response and rescue service operations.
3. Response: includes actions taken to save lives and prevent property damage, and to preserve the environment during emergencies or disasters. The response phase is the implementation of action plans.
4. Recovery: includes actions that assist a community to return to a sense of normalcy after a disaster.

These four phases usually overlap. Information and Communication Technology (ICT) is being used in all the phases, but the usage is more apparent in some

phases than in the others. For example, ICT support is very important during the disaster response (DR) phase of an emergency, which may commence with search and rescue, but in all cases the focus will quickly turn to fulfilling the basic humanitarian needs of the affected population. This assistance may be provided by national or international agencies and organisations. Effective coordination of disaster assistance is often crucial, particularly when many organisations respond and local emergency management agency capacity has been exceeded by the demand or diminished by the disaster itself. Tracing missing people, coordinating donor groups, recording the locations of temporary camps and shelters are examples of problems in the immediate post-disaster period that can be effectively addressed by using ICT.

In this paper we focus on disaster response exploitation based on the concept of *ubiquitous computing*, whose main objective is to provide globally available services and resources in a network by giving users the ability to access them anytime and anywhere. In particular, we consider novel paradigms and propose advanced technical solutions for DR-supporting distributed systems. Recently, Gaber [8] has proposed two alternatives to the traditional client/server paradigm (CSP) to design and implement ubiquitous and pervasive applications: the Adaptive Services/Client Paradigm (SCP) and the Spontaneous Service Emergence Paradigm (SEP). In other words, the peer-to-peer paradigm is completed respectively by the self-organization and the self-adaptation principles. In SCP a decentralized and self-organizing middleware that implements an intelligent network should be able to provide services to users according to their availability and the network status. In SEP, spontaneous services can be created on the fly and be provided by mobile devices that interact through ad hoc connections without any prior planning.

In order to enforce these paradigms to systems which include devices with limited processing and storage resources, lightweight middleware components are strongly required. In [5], Bodhuin *et al.* compare some traditional solutions for net-centric computing middleware, such as Jini, OSGi and CORBA, listing their pros and cons. Not surprisingly, the survey does not include Sun MicroSystem's JXTA [28], probably due to the fact that in year 2005 an implementation for mobile devices was not completed. JXTA is mainly the specification of a set of open protocols for building overlay networks, independent from platforms and languages. Currently there are three official implementation of JXTA protocols: J2SE-based, J2ME-based and C/C++/C♯-based. In particular, an almost complete version of the JXTA Java Micro Edition (JXTA-J2ME, a.k.a. JXME) has been recently released. It provides a JXTA compatible platform on resource constrained devices using the Connected Limited Device Configuration (CLDC) with Mobile Information Device Profile 2.0 (MIDP), or Connected Device Configuration (CDC). Supported devices range from smartphones to PDAs.

How does JXTA cope with the service concepts characterizing the previously summarized paradigms for ubiquitous computing? The Web Service community considers services as *the only mean* for accessing resources (this concept has been explicitly formalized in the WSRF specification [34]), yet centralized registries,

themselves exposed as services (like UDDI), are still deemed the primary tool to support the publication and the discovery phases. Unfortunately, a peer-to-peer network of Web Service providers with a publication/discovery infrastructure implemented as a set of interacting Web Services would be absolutely unefficient due to the heavyness of the SOAP messaging protocol. On the other side, in JXTA each peer's service is just an example of resource which can be exploited by the user which owns the peer, or shared in the network, *i.e.* advertised by the user and exploited by other users. Resource descriptions have the shape of XML documents, namely advertisements. A JXTA advertisement can be filled with any document, *e.g.* a WSDL interface if the shared resource is a Web Service. In summary, JXTA provides a lot of flexibility by separating basic infrastructural services, mandatory for all peers, from specialized services, with different levels of description and efficiency.

Within the context of JXTA and Web Service integration, we are responsible for the development and maintenance of the JXTA-SOAP component [16], enabling Web Service deployment in JXTA peers, as well as distributed WSDL publication and discovery, and SOAP message transport over JXTA pipes (*i.e.* virtual communication channels which may connect peers that do not have a direct physical link, resulting in a logical connection bound to peer endpoints corresponding to available peer network interfaces with an example being a TCP port and associated IP address). JXTA-SOAP is currently implemented in two versions: J2SE-based (fully featured, extending JXTA-J2SE) and J2ME-based (partially featured, extending JXME).

The remainder of the paper is organized as follows. Section 2 illustrates the technological franework we propose to support disaster response activities. Section 3 describes related work on disaster response ICT systems, middleware for peer-to-peer service-oriented ubiquitous computing, and Web Services on resource-constrained devices. An overview of the internal design of the JXTA-SOAP component, and many details about its J2SE and J2ME implementations, are given in section 4. Section 5 describes a disaster response application we developed on top of JXTA-SOAP. Finally, section 6 provides a conclusive discussion and describes future work.

2 Proposed technological framework

Disasters can happen anywhere at any time. Some disasters can be prevented, while some others cannot. Preparedness however greatly increases our chances to reduce their impact. Developing effective early warning and alert systems often can save thousands of human lives. From the 2004 tsunami in the Indian Ocean to the forest fires that ravaged southern Europe in the summer of 2007, recent natural and man-made disasters (including also conflict-related complex emergencies) have highlighted the need for a more effective response.

In the area of civil protection the European Commission has recently proposed to improve the EU's capacity through a number of important measures [6].

Among others, building up the Monitoring and Information Centre (MIC), playing the role of operational centre for European civil protection intervention. This requires a qualitative shift from information sharing/reacting to emergencies towards proactive anticipation/real time monitoring of emergencies and operational engagement/coordination. This includes early warning systems, performing needs assessments, identifying matching resources, and providing technical advice on response resources to the Member States; developing scenarios, standard operating procedures and lessons learned assessments; implementing the Commission competencies to pool available transport and provide co-financing for transport; increasing training and exercise activities for Member States and other experts; and helping the Member States to set up common resources. This implies also the use of monitoring capabilities such as those developed under the Global Monitoring for Environment and Security (GMES) initiative [11] or enabling tools like GALILEO (the European satellite navigation system) [9].

Our framework focuses on the problem of *identifying matching resources* in response to disasters. The most important are human resources, *i.e.* Civil Protection volunteers, Red Cross doctors and medical attendants, firemen, policemen, army officers, etc. In a typical scenario, it is necessary to coordinate the action of rescuers that are already in the disaster location, and those that are on vehicles and may be requested to reach the disaster place. The purpose of our work is also to support the work of the back-end operators, improving the ICT infrastructure that must allow not only communications among actors, but also automated gathering, elaboration and delivery of the huge amount of data collected by each actor.

For example, in case of flooding, first volunteers arriving at the disaster location may notice that some roads are interrupted. If they are equipped with a mobile device including a camera, they may (1) send short alert messages, including their coordinates obtained by means of GPS/GIS, and (2) take and send photos to provide a more detailed description of the environment. The back-end system collects and filter these data, and sends useful advices (such as the best route to be followed) to rescue vehicles which are directed to the disaster place (fig. 1).

The infrastructure of the service-oriented applications we envision is a peer-to-peer overlay network, which is placed at level 5 in the TCP/IP stack and is almost independent from the possible connectivity solutions, that we summarize in the following.

For long distance communications, in Europe the most used infrastructure is General Packet Radio Service (GPRS), which is a packet-oriented Mobile Data Service available to users of Global System for Mobile Communications (GSM) and IS-136 mobile phones (the so-called second generation - 2G). It provides data rates from 56 up to 114 kbit/s. A more powerful technology which is assuming higher importance is the Universal Mobile Telecommunications System (UMTS), one of the third-generation (3G) cell phone technologies, which is also being developed into a 4G technology. Both 2G and 3G technologies require the presence of base stations on the territory. In case of heavy disasters such as hur-

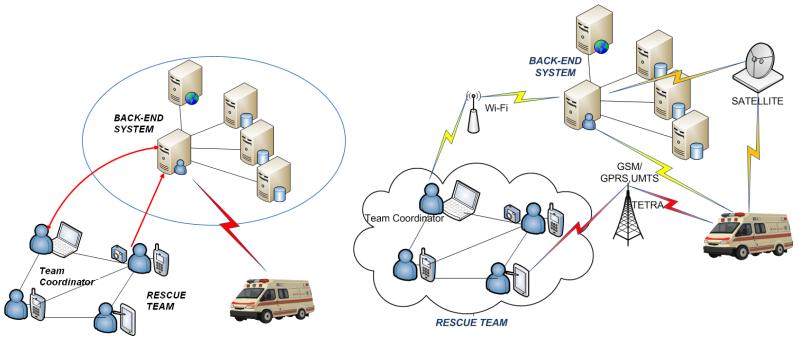

Fig. 1. The back-end system, the rescue operators and vehicles are connected in a peer-to-peer overlay network, offering services to each others (left image). Connectivity is guaranteed by different technological solutions (right image).

ricanes, base stations may be damaged, for which satellite and/or TETRA-based communications are the other options. TETRA is a telecommunications standard for Private Mobile Radio (PMR) systems developed by ETSI as an answer, at European level, to the evolving needs of PMR Operators, which have to cope with traffic congestion and a growing demand for speech and data services.

For local communications among actors equipped with mobile devices, infrastructured commmunications are usually based on WiFi. If some devices are out of the range of the WiFi access point, they can try to set up a mobile ad-hoc network (MANET), which is a self-configuring network of mobile routers (and associated hosts) connected by wireless links, the union of which form an arbitrary topology. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet.

3 Related work

This section illustrates the state of the art of software architectures supporting rescue operators during emergencies, technologies for ubiquitous peer-to-peer service sharing, and middleware solutions for deploying and consuming Web Services on resource-constrained devices.

3.1 State-of-art DR projects

In emergency scenarios, several teams belonging to different organizations need to collaborate in disaster management activities. The WORKPAD project [35], funded by the EU Commission, focuses on response and short-term recovery phases in which Public Safety Systems (PSS) use computer programs to give instructions to the rescue teams. Each team member is equipped with hand-held

devices (PDAs) and communication technologies, and should carry on specific tasks. The developed framework has two different levels: an integrated back-end community, mainly constituted by traditional computers, that interact in a peer-to-peer fashion, providing advanced services requiring high computational power, knowledge and content integration, and a set of front end peer-to-peer communities that provide services, mainly by adaptively enacting processes on mobile ad-hoc networks, to human workers.

A service-oriented architecture may be developed to achieve integration and data exchange among the different organizations systems in a disaster scenario, as proposed in AID-N project [37]. Individual tools interact with the shared data models through a set of publicly available and descriptive Web Services. The AID-N architecture addresses critical interoperability challenges through the design of data models to support a diverse variety of data from disparate systems, the design of data exchange standards to access the data model, and the design of web services that support the information needs for each system. This architecture enables real-time data communication between three deployed systems: a pre-hospital patient care reporting software system, a syndromic surveillance system and a hazardous material reference software system.

In [38] an Artificial Emergency-Logistics-Planning System (AELPS) is envisioned, which is based on artificial-society theory and uses agent modeling to describe the behaviour of basic elements in an emergency-logistics system. AELPS can form the basis of a complex computational platform that generates logistics activities during disaster relief and gives intuitive results that can be used in emergency-logistics planning. The emergency-logistics planning for natural disaster must simultaneously consider different types of request and manage the formation of coalitions of different working units in order to employ different subsystems to manage all the tasks in an emergency situation.

An agent-based simulation approach is described in [39, 40] for the evaluation of scenarios concerning different possible rescue processes. A six step methodology is proposed for developing a computer based simulator, that needs to know all the cognitive activities of emergency personnel. Rescue operators have predefined roles but may also organize themselves dinamically in groups and teams. As in real life, there are predefined rules and procedures, but rescue personnel often react to their environment in an unpredictable way. By modeling rescue personnel as agents, these characteristics made an agent-based apppproach a suitable tecnique to use. The multi-agent model also includes the notion of centralized rescue strategy used in real life; the simulator is used to test the effects of a distributed strategy, that is where independent sub-teams cooperate and share resources.

3.2 Ubiquitous peer-to-peer sharing of services

OSGi [21] is a Java-based technology which provides a service-oriented plug-in-based platform for application development. The core component of the OSGi Specifications is the OSGi Framework, which provides a standardized environment to applications (called bundles). On top of the Framework, services are

specified by a Java interface. Bundles can implement this interface and register the service with the Service Registry. Clients of the service can find it in the registry, or react to it when it appears or disappears. Advanced networking features, such as *e.g.* peer-to-peer connectivity, are not provided by OSGi and must be implemented on top of it.

To the best of our knowledge JXTA-SOAP is the sole open source project for P2P sharing of Web services being actively maintained and updated. WSPeer [14] is a J2SE toolkit for deploying and invoking Web Services in peer-to-peer Grid environments, which wraps Globus Toolkit core libraries to support the WS Resource Framework (WSRF) [34]. More interesting for ubiquitous computing environments is the Mobile Web Services Mediation Framework (MWSMF) [26, 27], an adaptation of Apache ServiceMix, which is an open source ESB (Enterprise Service Bus). It provides an hybrid solution, since it must be configured as JXTA-J2SE peer and established as an intermediary between Web Service clients and mobile hosts, the latter being configured as JXME peers. Web Service clients can invoke the services deployed on mobile hosts via the MWSMF, which compresses SOAP messages (to BinXML format) and sends them through JXTA pipes. The MWSMF also manages message persistence, guaranteed delivery, failure handling and transaction support. Unfortunately, the source code is not publicly availble and few details are given about the realization of lightweight Web Service providers running on mobile hosts.

3.3 Web Services on resource-constrained devices

Besides hardware constraints, mobile devices introduce many other specific challenges which make difficult the deployment of Web Services on top of them [4]. Unlike dedicated servers, mobile devices will typically have intermittent connectivity to the network. As a result, the services offered on a mobile device may not be accessible all the time. An application that uses or composes such Web Services needs to operate in an opportunistic manner, leveraging such services when they become available. On the server side, Web Services on mobile devices should also attempt to keep messages as short as possible. Another issue is addressing: when a mobile device moves between different locations, it may move from one administrative domain to another, causing a change in the IP address and even the Internet domain name. However, with the P2P in place, the need for the Public IP can be eliminated and the mobiles can be addressed with unique peer ID. Each device in the P2P network is associated with the same peer ID, even though the peers can communicate with each other using the best of the many network interfaces supported by the devices like Ethernet, WiFi, etc. [26].

Since the WS message protocol, namely SOAP, introduces some significant overhead, few toolkits support the deployment of Web Services on limited devices, such as PDAs, smart phones, etc. One is gSoap [30], which provides a WS engine with run-time call de-serialization. Unfortunately, gSoap is written in C/C++, thus requiring a priori stub/skeleton generation by means of a specific compiler, which also means lack of portability.

Looking at the Java Micro Edition (J2ME) platform, most libraries are only for client side functionality. The Java Wireless Toolkit (WTK) provides J2ME Web Services API (WSA) [32], based on JSR 172 [17], which specifies runtime ServiceProvider interface to allow the generation of portable stubs from WSDL files. The specification contains some notable limitations, most of them due to the requirement for WS-I Basic Profile compliance. Conforming to the profile ensures interoperability, but also prevents using alternative methods. Another widely used solution is the kSoap2 [19] open source component, which is a parser for SOAP messages (with RPC/literal or document/literal style encoding), not supporting the generation of client side stubs. kSoap2 is compliant with devices lacking JSR 172 support, and allows to access non WS-I conformant services.

To the best of our knowledge, the unique solution enabling J2ME applications (CLDC, CDC) as service endpoints is the Micro Application Server (mAS) [20]. It can be considered a lightweight version of Axis, by which it is inspired. For this reason we have chosen it to implement the J2ME version of JXTA-SOAP.

4 The JXTA-SOAP component

JXTA is a Sun MicroSystems' open framework which defines peer-to-peer protocols that should allow a vast class of networked devices (smartphones, PDAs, PCs and servers) to communicate and collaborate seamlessly in a highly decentralized fashion. The JXTA framework defines a naming scheme, advertisements, peergroups, pipes, and a number of core policies, while the JXTA middleware implements the specifications in Java and C++.

The JXTA-SOAP component is an official extension for the Java version of JXTA middleware, enabling Web Service sharing in peer-to-peer networks. JXTA-SOAP has been designed having in mind ubiquitous computing needs, to reduce the complexity otherwise required to build and deploy peer-to-peer service-oriented applications. In a previous work we described the internal architecture of JXTA-SOAP, with the purpose of conceptualizing its main features at a high abstraction level [2]. In particular we focused on service deployment, publication, lookup and invocation, considering also security aspects.

The internal architecture of JXTA-SOAP based peers is illustrated in figure 2. We implemented two (interoperable) versions of JXTA-SOAP: J2SE-based, extending JXTA-J2SE, and J2ME-based, extending JXTA-J2ME. In the following we describe their features and the different technological solutions they rely on.

4.1 JXTA-SOAP for Java Standard Edition (J2SE)

The J2SE version of the JXTA-SOAP component supports service deployment, discovery, and invocation, with optional use of standard mechanisms to secure communications among peers. The core of the component is the Apache Axis engine (v1.4), which is a producer/consumer of SOAP messages. Usually Axis is deployed in a Web application server, such as Apache Tomcat, together with

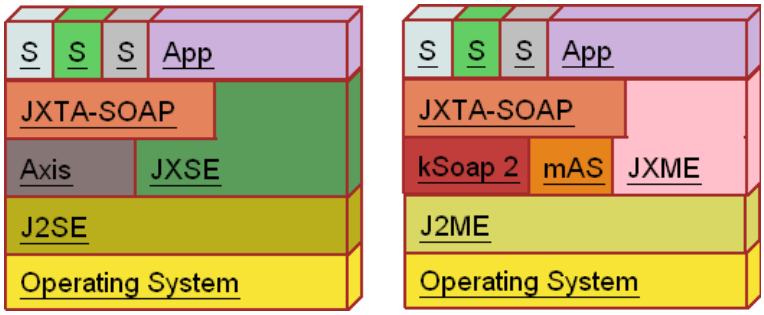

Fig. 2. Architectural layers of service-oriented peers based on JXTA-SOAP. The J2SE version (on the left) is based on Axis and JXSE, while the J2ME version (on the right) is based on kSoap2 and mAS.

the implementions of the Web Services to be deployed, while client applications use the Axis Java API to create request instances. The Axis engine provides the processing logic, either client or server. When running, it invokes a series of Handlers according to a specific order which is determined by two factors - deployment configuration, and whether the engine is a client or a server. The object which is passed to each Handler invocation is a MessageContext, *i.e.* is a structure which contains several important parts: 1) a "request" message, 2) a "response" message, and 3) a bag of properties.

At runtime, for a service provider its service objects are deployed in the Axis engine, which implements the JAX-RPC API, one of the standard ways to program Java services, also supporting the lifecycle of service endpoint instances. After being loaded and instantiated, the JAX-RPC runtime system is required to initialize the service instance before any requests can be serviced. A context parameter is pssed to the initialization function, enabling the service instance to access the context provided by the underlying JXTA-SOAP based runtime system. The context parameter is typecasted to an appropriate Java type. For services deployed in a JXTA-SOAP based runtime system, the Java type of the context parameter is defined by the developer that is using the JXTA-SOAP API, and passed to the service object. The latter instantiates the Service Descriptor, creates and publishes the public pipe and the service advertisement, and notifies itself to the Axis engine. Services can be deployed anytime, without the need to restart the peer.

Once a service instance has been initialized, the Axis engine may dispatch multiple remote invocations to it. After that, when the Axis engine determines that the service instance needs to be removed from service of handling remote invocations, it destroys it. In the implementation of the destruction functionality, the service object releases its resources.

For remote service invocation, a consumer peer needs to intantiate a Call object. JXTA-SOAP's Call class extends Axis' default one, overloading the use of service URLs with the use of the Service Descriptor and the public pipe advertise-

ment of the service. To create Call instances, the peer uses the implementation of the Call Factory class provided by Axis.

We previously described the tasks which are performed when a Web Service is deployed by a peer, and we mentioned that some parameters are put in the Service Descriptor for further use by the Axis engine. In particular, one of these parameters is the Web Service Deployment Descriptor (WSDD). When the WSDD is sent to the Axis engine running in the peer, an *Invoker* [31] is informed that it supports the new Web Service. Thus, when an invocation reaches the peer, the Invoker looks up the class which implements the service, and lets the instance handle the request message. In details, the Invoker reads incoming messages and demarshals the parameters inserted by the consumer peer's Requestor (absolute reference of the service, operation name, arguments, return value) and dispatches the message to the targeted service.

4.2 JXTA-SOAP for Java Micro Edition (J2ME)

The J2ME version of JXTA-SOAP supports Connected Device Configuration (CDC) and Personal Profile. We implemented the API which enables the development of peers that are able to deploy, provide, discover and consume Web Services in a JXTA-SOAP network. Since Axis is not available for the CDC platform, we adopted kSoap2 [19] as SOAP parser (for consumer functionalities) and, for service provision, we integrated the mAS [20] lightweight engine.

Service invocation is allowed by a kSoap2 based implementation of the Call Factory class. The latter instantiates a kSoap2's Soap Object, and sets all the properties for message exchanging through JXTA pipes. Soap Object is a highly generic class which allows to build SOAP calls, by setting up a SOAP envelope. We have maintained the same structure of J2SE-based version for Call Factory, to allow portability of service consumer applications from desktop PCs or laptops to PDAs. Internally, the Call Factory class creates a Soap Object passing references to the Service Descriptor, the public pipe advertisement of the service and the peergroup as parameters for the creation of the Call object.

The Call Factory class also allows to create an instance of kSoap Pipe Transport, the class we implemented to manage the transmission of SOAP messages using service pipes. The kSoap2 API provides a Transport class that encapsulates the serialization and deserialization of SOAP messages, but does not manage communication with the service; the HTTP Transport subclass, both in CDC and CLDC version, allows service invocation over HTTP, setting up the required properties, but it uses URLs as absolute references of remote services, and it is not suitable for usage in JXTA-SOAP, where services (as every resource) are identified by JXTA-IDs and must be invoked through JXTA pipes. Thus, we extended the Transport class with the implementation of a call functionality that configures a JXTA pipe and creates the messages to be sent over it.

After instantiating the transport using the Call Factory class, the consumer peer creates the request object, indicating the name of the remote method to invoke and setting the input parameters as additional properties. This object is assigned to a Soap Serialization Envelope, as the outbound message for the soap

call; Soap Serialization Envelope is a kSoap2 class that extends the basic Soap Envelope, providing support for the SOAP Serialization format specification and simple object serialization. The same class provides a getResponse method that extracts the parsed response from the wrapper object and returns it.

Referring to service provision, we integrated the Server class of the Micro Application Server (mAS) into the basic service class of the JXTA-SOAP API.

mAS implements the Chain of Responsibility pattern [12], the same used in Axis. It avoids coupling the sender of a request to its receiver by giving more than one object a chance to handle the request; receiving objects are chained and and the request passed along the chain until an object handles it. Moreover, mAS allows service invocation by users and service deployment by administrator; it also supplies browser management of requests, distinguishing if the HTTP message contains a Web page request or a SOAP envelope.

5 Example DR Application

Using JXTA-SOAP mobile, we developed a GUI-based application that allows to join a JXTA-based P2P network to share services for supporting disaster response activities. The application has several overlapping panels (or tabs), each one being related to a specific function. In the following we describe them, with the help of the screenshots that have been grouped in figure 3.

The Local panel (top left screenshot in figure 3) shows locally deployed services. A table lists all services and a Share Service button allows to publish their advertisements. The back-end system can invoke these services without interrupting the activity of the rescue operator. For example, there could be a service which provides the location of the rescue operator. Another (less obvious) service provides the photos of the disaster location that have been taken from the operator, without interrupting his activity. Other services could require the rescue operator to provide information to the service requestor (the back-end system, but also other operators).

The Remote panel shows discovered remote services. It is possible to search for services in the P2P network (offered by other rescue operators), and to select one of them from the resulting list, in order to see all the operations it offers, which are shown in the Operation tab. The user puts a description of the desired service in the search field, and all the matching services are listed in the table. Some services from the back-end are assumed to be always available, such as getPhotoFromSatellite.

The Operation management panel shows all the functionalities provided by the selected service; the operator can choose a particular operation and fill the input parameters table in the invocation panel.

The Invocation panel is where the user introduces the required parameters for service invocation. If the service returns a result, the user can select where to save it, for example in a file stored locally. Finally, the Task tab is the one in which the rescue operator can see its task list, provided by the back-end depending on the equipment of the operator.

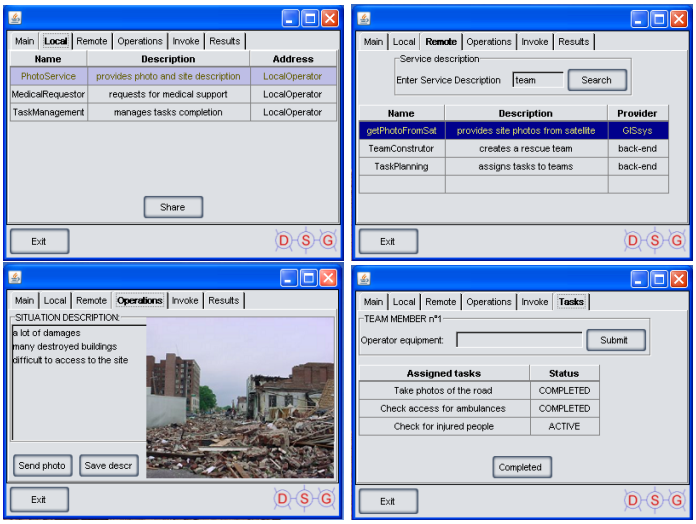

Fig. 3. Local services panel (top left). Remote service selection panel (top right). Operation management panel (bottom left). A photo of the disaster location is taken, and a short description written, both ready to be sent to the beck-end upon request, or proactively by the rescue operator. Task panel (bottom right), where the rescue operator can see its tasks (decided by the back-end) and flag them as executed, when they are.

6 Conclusions and future work

In this paper we proposed JXTA-SOAP as a powerful solution for building service-oriented, peer-to-peer ubiquitous applications to support disaster response activities. The JXTA-SOAP component enables Web Service deployment, distributed publication and discovery, and SOAP message transport over the JXTA peer-to-peer overlay network. All kinds of devices, also constrained ones, can be used, because JXTA-SOAP comes in two interoperable versions: J2SE-based and J2ME-based.

Future work on JXTA-SOAP will mainly focus on implementing the Web Service Resource Framework [34], in order to provide peers the ability to access and manipulate state, *i.e.* data values that persist across, and evolve as a result of, Web Service interactions. This is particularly important for services like Disaster Location Monitoring, which requires to collect contextual data but also historical information from services dispersed over the network.

References

1. MIT Computer Science and Artificial Intelligence Laboratory, *AIRE Group*, http://aire.csail.mit.edu/index.shtml

2. M. Amoretti, M. Bisi, F. Zanichelli, G. Conte, *Enabling Peer-to-Peer Web Service Architectures with JXTA-SOAP*, IADIS International Conference e-Society 2008, Algarve, Portugal, April 2008.
3. E. Avatangelou, R. F. Dommarco, M. Klein, S. Muller, C. F. Nielsen, M. P. S. Soriano, A. Schmidt, M.-R. Tazari, R. Vichert, *Conjoint PERSONA-SOPRANO Workshop*, Proc. of the first European Conference on Ambient Intelligence (AmI-07), Darmstadt, Germany, November 2007.
4. S. Berger, S. McFaddin, C. Narayaswami, M. Raghunath, *Web Services on Mobile Devices - Implementation and Experience*, Proc. of the Fifth IEEE Workshop on Mobile Computing Systems & Applications, Monterey, CA, USA, October 2003.
5. T. Bodhuin, G. Canfora, R. Preziosi and M. Tortorella, *Open Challenges in Ubiquitous and Net-Centric Computing Middleware*, 13th IEEE International Workshop on Software Technology and Engineering Practice, September 2005.
6. Commission of the European Communities. Communication on Reinforcing the Union's Disaster Response Capacity. Brussels, March 2008.
7. S. Edwards, *User Driven and Seamless Mobility Services for Disabled and Older People: the ASK-IT Project*, Proc. of the 5th Annual Moving On Conference, Glasgow, March 2006.
8. J. Gaber, *Spontaneous Emergence Model for Pervasive Environments*, IEEE Globecom Workshop 07, Washington DC, November 2007.
9. Gallup Organization. General public survey on the European Galileo Programme. June 2007.
10. L. Z. Granville and A. Panisson, *GigaMAN P2P project*, http://gigamanp2p.inf.ufrgs.br
11. Commission of the European Communities. Communication on Global Monitoring for Environment and Security (GMES): Establishing a GMES capacity by 2008. Brussels, 2004.
12. Gamma E, Helm R, Johnson R, Vlissides J. *Design Patterns.* Addison-Wesley, 1995.
13. G. D. Haddow, J. A. Bullock. *Introduction to Emergency Management.* Amsterdam: Butterworth-Heinemann, 2004.
14. A. Harrison, I. Taylor, WSPeer - An Interface to Web Service Hosting and Invocation, *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS05)*, Denver, Colorado, USA, May 2005.
15. IST Advisory Group, *Scenarios for Ambient Intelligence in 2010*, European Commission, 2001.
16. Distributed Systems Group and Sun MicroSystems, *JXTA-SOAP project*, https://soap.dev.java.net
17. Sun MicroSystems, *JSR 172: J2ME Web Services Specification*, http://jcp.org/en/jsr/detail?id=172
18. Krishna A., Schmidt D. C., Stal M., Context Object: A Design Pattern for Efficient Middleware Request Processing. *Proc. of the 12th Pattern Language of Programming Conference*, Allerton Park, Illinois, September 2005.
19. S. Haustein, J. Seigel, *kSoap2 project*, http://ksoap2.sourceforge.net
20. P. Plebani, *mAS project*, https://sourceforge.net/projects/masproject
21. OSGi Alliance, *OSGi: the Dynamic Module System for Java*, http://www.osgi.org
22. S. Peters, H. Shrobe, *Using Semantic Networks for Knowledge Representation in an Intelligent Environment*, Proc. of 1st Annual IEEE International Conference on Pervasive Computing and Communications (PerCom '03). Ft. Worth, TX, USA, March 2003.

23. I. Pyarali, M. Spivak, R. Cytron, D. C. Schmidt, *Evaluating and Optimizing Thread Pool Strategies for Real-Time CORBA*, Proc. of the ACM SIGPLAN Workshop on Optimization of Middleware and Distributed Systems (OM 2001), Snowbird, Utah, USA, June 2001.
24. C. Ramos, J. C. Augusto, D. Shapiro, *Ambient Intelligence - the Next Step for Artificial Intelligence*, IEEE Intelligent Systems, Vol. 23, No. 2, March/April 2008.
25. P. Costa, G. Coulson, C. Mascolo, L. Motolla, G. P. Picco, S. Zachariadis *A Reconfigurable Component-Based Middleware for Networked Embedded Systems*, International Journal of Wireless Information Networks, Springer, 2006.
26. S. N. Srirama, M. Jarke and W. Prinz, *A Mediation Framework for Mobile Web Service Provisioning*, Proc. of the 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06), Hong Kong, China, October 2006.
27. S. N. Srirama, M. Jarke and W. Prinz, *MWSMF: a Mediation Framework Realizing Scalable Mobile Web Service*, Proc. of Mobilware'08, Innsbruck, Austria, February 2008.
28. B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.-C. Hugly, E. Poyoul, B. Yeager, Project JXTA 2.0 Super-Peer Virtual Network, *Technical Report*, Sun Microsystems, 2003.
29. M. Vallee, F. Ramparany, L. Vercouter, *A multi-agent system for dynamic service composition in ambient intelligence environments*, Proc. of the Third International Conference on Pervasive Computing (Pervasive 2005), Munich, Germany, May 8-11, 2005.
30. R. A. van Engelen and K. Gallivan, *The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks*, Proc. of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), pp.128-135, Berlin, Germany, May 2002.
31. M. Volter, M. Kircher, U. Zdun, *Remoting Patterns*, Wiley, 2005.
32. Sun MicroSystems, *J2ME Web Services APIs (WSA)*, http://java.sun.com/products/wsa/
33. Banaei-Kashani F, Chen C, Shahabi C. WSPDS Web Services Peer-to-peer Discovery Service. *The 2004 International Symposium on Web Services and Applications*, Las Vegas, Nevada, USA, June 2004.
34. OASIS, *Web Services Resource Framework (WSRF) v1.2*, April 2006.
35. Catarci, T., de Leoni M, marrella A, Mecella M., Salvatore B., Vetere G., Dustdar S, Juszczyk L., Manzoor A., Truong H.L. *Pervasive Software Environments for Supporting Disaster Response*, IEEE Internet Computing 2008.
36. Mecella M., Catarci, T., Angelaccio M, Buttarazzi B., Krek A., Dustdar S, Vetere G. *WORKPAD:an Adaptative Peer-to-Peer Software Infrastructure for Supporting Collaborative work of Human Operators in Emergency/Disaster Scenarios*, 2006.
37. Hauenstein L., Gao T., Sze T. W., Crawford D., Alm A., and White D. *A Cross-Functional Service-Oriented Architecture to Support Real-Time Information Exchange in Emergency Medical Response*, 2006.
38. Li L., Tang S. *An artificial Emergency-Logistics-Planning System for Severe Disasters*, IEEE Intelligent Systems, July/August 2008
39. Dugdale J., Bellamine-Ben Saoud N. , pavard B., and Pallamin N. *Simulation and Emergency Management*, 2008.
40. Bellamine-Ben Saoud N., Ben Mena T., Dugdale J., Pavard B., and Ben Ahmed M., *Assessing large scaleemergency rescue plans: an agent-based approach*, Special Issue on Emergency Management Systems. International Journal of Intelligent Control and Systems. Vol. 11, No. 4, Dec. 2006