A Service-based Infrastructure for Advanced Logistics

Luciano Baresi, Daniele M. Braga, Marco Comuzzi,
Filippo Pacifici, and Pierluigi Plebani
Dipartimento Elettronica ed Informazione – Politecnico di Milano
piazza Leonardo da Vinci, 32 - 20133 Milano, Italy
baresi|braga|comuzzi|pacifici|plebani@elet.polimi.it

ABSTRACT

The solutions for dangerous goods transportation usually need to deal with several constraints and requirements. The technological level in logistics is low and fleet management systems typically employ proprietary solutions. Web services can help tackle these problems since they help widen the market, reduce the gap between advanced solutions and small to medium enterprises, and foster the integration with other existing systems. The paper presents our experience on implementing a service-based framework for advanced logistics. We propose a layered organization to abstract physical devices into Web services, which are then used to ease the communication between containers and interested parties. The main ideas behind the framework are discussed by considering two key processes for the management of dangerous goods: the processes of tracking and tracing goods and of alerting public authorities in case of danger situations.

1. INTRODUCTION

The solutions for dangerous goods transportation usually need to deal with several constraints and requirements. As for normative standards, it is worth noting that there is a European Agreement concerning the International Carriage of Dangerous Goods by Road (commonly referred to as ADR, as the acronym is derived from the French translation) which rules the transportation of dangerous materials on international itineraries. The agreement itself is brief and simple, and in its most important article states that, with the exception of certain exceptionally dangerous materials, hazardous materials may in general be transported internationally in wheeled vehicles, provided that two sets of conditions be met, addressing (a) the merchandise involved, and notably their packaging and labels, and (b) the construction, equipment, and use of the vehicles.

Currently, there are some other problems that arise in case of emergency. The identification of the company, the actual content of the transportation unit, and the best and most effective intervention strategy often takes a long time; an integrated and fast accessible information infrastructure may considerably reduce the intervention delay. Public Security authorities may intervene redundantly w.r.t. private specialized intervention teams, and they can even interfere with and obstruct one another; a better coordination is another expected benefit. It would be desirable to provide public authorities, like civil protection and fire brigades, with centralized monitoring systems for anticipating and preventing risky situations and timely detecting anomalies in the physical parameters of transported goods (temperature and pressure of flammable fluids or anomalous speeds). Also, it is in general quite difficult to plan and organize national and international transportation policies capable of minimizing the risk of accidents.

At this stage, transportation companies have experience in some solutions for managing their fleets which requires a direct involvement of drivers: e.g, using mobile phones or dedicated devices installed on the truck tractors. Therefore, these solution can be exploited only in case of supervised transportation units. In addition, such systems typically employ proprietary solutions. Web services can help tackle these problems. The use of open and standard technologies is a way to widen the market and reduce the gap between advanced solutions and small to medium enterprises. Web services also support open solutions that can easily be extended and integrated with other existing systems provided by new stakeholders.

In this paper, the paper presents our experience on implementing a service-based framework for advanced logistics. The infrastructure exploits Web services to (i) ease the communication with the container to obtain information of its status; and (ii) support the interoperability among the different organizations and public authorities to exploit the information from the container.

Such an infrastructure must deal with wide and heterogeneous areas where the trucks/containers should be tracked. Moreover, the presence of several geographic obstacles such as mountains, tunnels, and areas where the coverage of telecommunication networks is absent or weak, along with the potentially long periods in which the transportation units have no power supply but still need to send messages, require that multiple sensing and communication devices be adopted to provide redundant and robust solutions. For this purpose, we assume that each container has a set of sensors able to monitor the physical parameters of interest: for example, its position, temperature, and pressure. Different devices can be activated and deactivated according to the information the container has to communicate and the network infrastructure that can be exploited. For exam-

ple, parking areas may provide short-range communication infrastructures, and thus enable some devices, while when the container is traveling the only solution could be a GPS antenna to track the position and a GPRS transmitter to communicate with the interested parties.
Given the networked infrastructure, transportation companies, as well as public safety authorities, can obtain the status of the different containers by simply interacting with the corresponding Web services. Receiving fresh information about the position and status of the goods allows the travel planning unit to dynamically adapt the routes in case of anomalies, and to store the historical data for off-line analysis and policy optimizations. All the complexity about how the status is monitored and the network used to transmit the information is totally hidden by the Web services.
In particular, we have developed a layered reference architecture to integrate different kinds of active devices into existing information systems. Such an architecture hides differences of devices in terms of their technology through an abstraction. This way, a container is seen as a Web service able to provide information about the status of the goods it contains by means of different devices, which are enabled and disabled according to the needs and the different contexts.
The service-based framework is presented in two versions. In the first case, we assume that Web services run on traditional computers, while in the second case, scalability and flexibility issues made us flip our mind and think of Web services that run on small devices and thus "follow" the physical object they are associated with.

The rest of paper is structured as follows. Section 2 introduces the abstraction model that maps physical devices onto Web services and Section 3 exploits such abstractions and introduces the service-based framework. Section 4 describes a snapshot on the dynamic behavior of the system, and in particular, it discusses the processes of tracking and tracing goods and of alerting public authorities in case of danger situations. Section 5 surveys some related approaches and Section 6 concludes the paper.

2. DEVICE ABSTRACTIONS

The first step towards abstracting physical devices as Web services is the definition of the logical architecture of the system [1] to provide a conceptual view of the pervasive system and of its logic, independently of its distribution. A special-purpose metamodel defines the main concepts which are then used to define both a *structural* description of the system and a *behavioral* description of its components.

2.1 Structural description

Modeled systems are structured in three layers (Figure 1). The software components that populate each layer communicate with the upper layer by sending *events* and with the lower layer by invoking synchronous *commands*. Communication (both in terms of events and commands) can only occur between components defined in adjacent layers and that are explicitly connected. Highest layer elements can also communicate among them.

The *physical access layer* is in charge of abstracting the access points to the underlying device technology. Depending on selected technology, this layer may represent a middleware that accesses peripheral systems (such as Ultra Wide-Band tags or GPRS devices) or represent the operating system of a hardware device (like TinyOS for sensor networks [2]). Components in this layer are called *Operation Managers* and are intended to be off-the-shelf software elements.
On top of the physical access layer, we define the *logical layer* in which *Logical Objects* perform abstractions of single physical devices or aggregate physical homogeneous devices. These abstractions virtualize the actual devices, manage the communication with the *Operation Managers*, and supplement the physical object with functionality that is difficult/impossible to execute directly on the physical device (e.g., data filtering). The developer is in charge of defining the components in this layer and they are intended to be application-independent and technology-agnostic to be reusable by different business processes.
The highest abstraction layer is the *application Layer*. It provides abstractions (*Application Objects*) of concepts coming from the application domain and makes use of abstractions from the logical layer to provide/obtain data from the physical world. This way, an Application Object is able to adapt its behavior w.r.t. the devices actually used, and hides these differences to the application. A particular type of Application Object is the *Context* that, besides acting as a conventional Application Object, can also contain other objects.
An Application Object can also change its behavior by assuming different *Roles*. Every Role defines a new interface (along with a special-purpose behavior): an Application Object can explicitly change the role it plays, but we also usually assume that when it moves to a new Context, it also assumes a new Role.

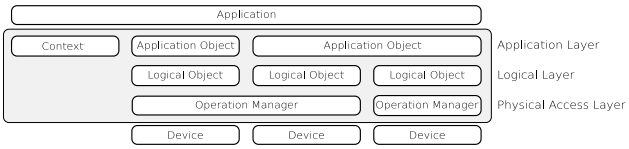

Figure 1: Layered architecture for device abstractions.

The example used in the paper assumes that containers can be traced in two different ways: a GPS/GPRS device for long-range monitoring, and UWB tags for short-range tracking. Every container is an application object that offers the capability of getting its position in an adaptive way by hiding the underlying technology. The container can belong to two different contexts: road and parking area. In the latter case, a container also offers the capabilities of setting and getting its content. This is done through UWB technologies that are not available when the container is traveling. The logical objects we defined are a GPS antenna and a UWB tag. GPS antenna is a logical object that is instantiated by the user when it instantiates the application object and it does not need a unique identifier. It provides operations to switch it on and off to allow the user to control energy consumption. For example, it may switch it off when the container is inside a parking area. UWB tags are read-write tags.
Figure 2 shows the model used for the application scenario. It comprises dedicated middleware infrastructures used to

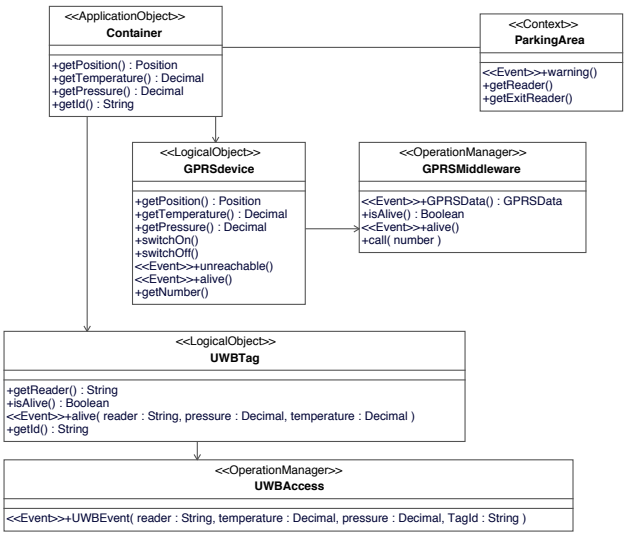

Figure 2: Model for the scenario.

access UWB tags and GPRS devices. Every GPRS device and every UWB tag is represented at runtime as an instance of the respective Logical Object: both Logical Objects offer similar operations. The Application Object `Container` adapts its behavior by means of the Logical Object it is linked to and of the `Role` and `Context` it is associated with.

2.2 Behavioral description

Every component (with the exception of *Operation Managers*) is provided with a dynamic description through a specialization of UML statecharts, which adds a taxonomy of signals and operations to manage issues specific to this architecture, such as attachments to and detachments from Contexts. Statecharts are used to model the behavior of a component when this is instantiated, but there are other processes that are difficult or impossible to define through Statecharts (since they model the behavior of a component in isolation): for examples the processes that describe how objects are created and deleted.

These processes are strongly influenced by the application domain: the instantiation of Operation Managers is mainly a decision at application level, Logical Objects are instantiated as soon as they become reachable through their Operation Managers, and Application Objects are instantiated according to the needs at application level. In this last case, objects may be created from direct user intervention or directly after the instantiation of a proper Logical Object. This implies that in suitable cases Application Objects can be instantiated before "their" physical devices are used.

Logical Objects can only be instantiated when the corresponding physical device can be accessed by the system: this means that we might have Application Objects without any Logical Object associated. If the Application Object is instantiated before the creation of the Logical Object, the problem becomes the association between the two objects. Otherwise the instantiation of Application Objects is similar to that of Logical Objects and it is triggered by the creation of the corresponding Logical Objects. A Logical Object is eventually destroyed whenever the Application Object that uses it is destroyed or upon an action performed by the Application Object itself.

As for Operation Managers, some technologies are active and generate proper events when a new device becomes available (e.g., RFID middleware infrastructures like EPC Global Networks). In other cases, when the infrastructure is passive, we need to adopt a *pull* approach and the query to the Operation Manager must be triggered by an external event such a user request or events generated by other Application Objects.

Moving to the scenario, the Application Object `Container` is instantiated by the user, `UWBtag` is instantiated when a suitable Operation Manager sees it for the first time and the `GPRSdevice` is instantiated as soon as it is connected to the system. Physical devices can temporarily become unreachable (for example when a `Container` identified through a GPRS enters a tunnel). In this case the Logical Object is not destroyed, but it simply disabled.

The application-level policies for creating objects are defined through *event-condition-action* rules that are executed by three *observer* components that are placed at the three abstraction layers. These observers can listen to events from the lower layer (including instantiation events), perform queries on the instance of the model, create new objects, and modify the association between them.

We can assume that the `UWBTag` object is already instantiated, associated, and in use. When `UWBAccess` raises a read event taken from the reader at the exit of the Warehouse the *observer* captures it and queries for the appropriate instance of `UWBTag` to be sure it is properly instantiated. Then, it invokes a *call* operation on the `GPRSMiddleware` with the GPRS number got from the binding database. When a response arrives from the device, the `GPRSMiddleware` raises an appropriate event that is captured by the observer. At this point the observer instantiates the `GPRSDevice`. The second observer intercepts this instantiation event, queries the instance for the particular Container and associates it to the `GPRSDevice`.

3. SERVICE INFRASTRUCTURE

According to the abstractions described in the previous section, the application object `Container` represents a programmatic way to obtain information about the status of a container: position, temperature, pressure. Considering a more complete scenario, a company owns several containers, so we need a system able to properly communicate with the containers and to possibly react in case of emergency.

If we think of implementing the logical architecture presented in the previous section, we need to exploit the technologies provided by the device manufactures for the Operation Managers, but we are free to adopt the technologies we want for the other layers. In this paper, we want to concentrate on the *application level* and think of its objects as if they were Web services for the reasons already discussed. From the perspective of the transportation company, fleet management results in a system able to communicate with a set of Web services each of them corresponding to a logical representation of a container. As a consequence, we can assume that the whole architecture can be organized as shown in Figure 3.

The *Monitor* represents the module managed by the company able to track and trace all the owned containers. The *Monitor* includes a Web service registry, the *Logical Containers Registry*, containing all the Web services correspond-

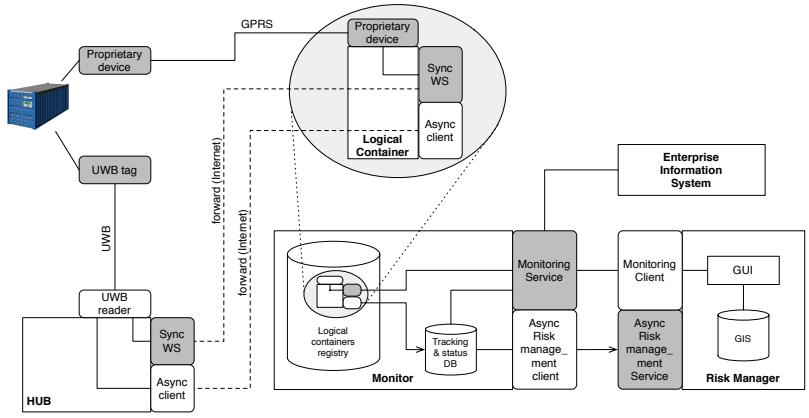

Figure 3: Architecture using logical containers abstraction

ing to all the containers owned by the company. When the company needs to know where a container is, then the proper Web service is retrieved and invoked. As we will deeply discuss in the next sections, a synchronous communication is performed when tracing is required: starting from an explicit request, the system tries to communicate with the container. Communication also occurs asynchronously; in this case the containers periodically transmit their vital data collected in the *Tracking and Status DB*.

As discussed in the previous section, the container Web service is in charge of hiding the physical issues about how a container can retrieve and transmit its status. What really happens behind the scene depends on where physically the container is. In this paper, we consider two main situations: (i) the container is on the road and the communication occurs through a long-range channel such as GPRS; (ii) the container is parked in a hub —ready to be commuted on a train— then it can communicate through a short-range channel, such as Ultra Wide Band (UWB) or Wi-Fi. The information so obtained will be forwarded to the container Web service throughout an HTTP communication over a common internet connection.

Notice that the *Monitor* module is a Web service, and it is able to manage all the fleet of containers owned by the transportation company. In more detail, the *Monitor* module exports the *Monitoring Service*, a synchronous Web service that returns the status given the container identifier. Such a Web service can be used by the existing Enterprise Information System to support new features. In the same way, the *Monitoring Service* can be also invoked by authorized third parties in case of emergency.

The infrastructure presented so far is compliant with the features of current devices able to detect and transmit the status of a container. Current solutions, in fact, are able to measure physical dimensions (e.g., temperature, pressure), and can also be coupled with localization devices such as GPS. At this stage, the communication of such a detected information can occur only by low level protocols: e.g., GPRS, Wi-Fi, UWB, Zigbee. Even if this situation is fine with the transportation company, the hub needs to be properly equipped with devices able to get the status of the containers, e.g., UWB readers or Wi-Fi access points. As a consequence, this approach dependents too heavily on the set of specific protocols adopted by the short-range communication: (i) all the hubs must support the same communication protocols and (ii) once the related infrastructure is set, it is difficult to adopt different short-range protocols. This dependency reflects on the transportation company too: they own the containers and they cannot decide which short-range protocols the container can support.

For this reason, we also propose an alternative solution which in the near future could substitute the previous one. Figure 4 shows this new approach. In this case, we assume that the devices installed on the container have enough capacity in terms of memory, power, and computation to support more complex applications. Currently, several applications have been developed specifically for mobile environments and on entities with limited resources. For instance, there are many low consumption devices (like Sun Spot technology [3] or other devices programmable with Java 2 Micro Edition [4]) that allow the execution of a simple HTTP server or HTTP client. This way, containers became *smart containers*. This way containers moves around together with the software elements that "publish" and advertise these services.

This new architecture will solve the problems previously discussed, but this kind of software solution on tiny devices is not mature enough to achieve our goal. Now, the hubs and the transportation companies need to agree only on the data format to be exchanged and HTTP represents the communication protocol. Each hub is free to adopt any short-range protocol. The only constraints is to support HTTP. In addition, the smart container need to support as many short-range communication protocols as possible to be compliant with all the hubs.

4. EXAMPLE PROCESSES

Regardless of the solution adopted for connecting containers and the *Monitor*, in this section we introduce two macro-processes, namely tracking and tracing of dangerous goods, and risk management to better exemplify the proposal.

4.1 Tracking and tracing

According to the common understanding, *tracking and tracing* is the macro-process of recording the past and present position of a shipment, as it passes through different handlers on its way to its destination, through a distribution network. According to our scenario, the tracking and trac-

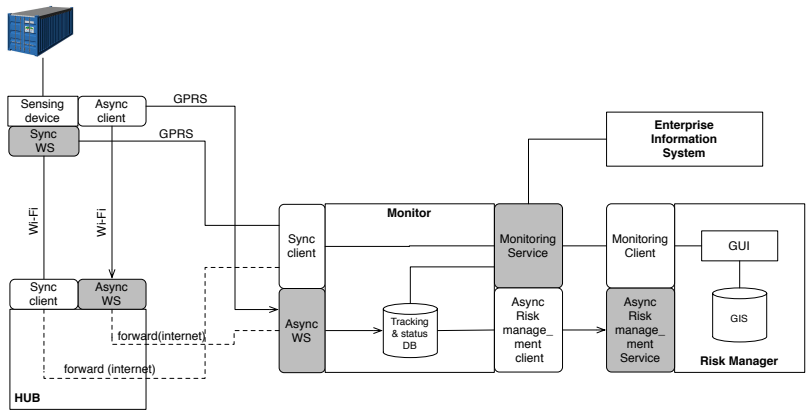

Figure 4: Architecture using smart containers

ing processes need to take into account not only the position but also additional parameters such as temperature and pressure, i.e., the container *status*.
The knowledge of the exact current status of a container (tracking) and capability of reconstructing its way to the current status (tracing) are essential for monitoring the quality of the transportation service and for its improvement.
In our scenario, the main technical difficulty for tracking and tracing is that containers pass through different environments, in which the availability of communication systems vary remarkably, whereas status information should continue flowing as regularly as possible. We identified five processes which involve an information exchange relevant for the main tracking and tracing macro-process:

- Loading of the dangerous goods onto the container, so as to initiate the shipment;
- Unloading of the goods at the receiver's site, so as to terminate the shipment;
- Transportation through the road network;
- Transportation through the railway network;
- Switch from the road (railway) network to the railway (road) network within intermodal hubs, from a road tank truck to a rail tank wagon (or viceversa);

As anticipated, *tracking* is a typically synchronous process, as the Monitoring Service introduced in Section 3 is invoked by any authorized actor whenever knowledge of the current status of a container is required; when an invocation occurs, the *Monitor* always queries the container for a fresh reading of sensing devices and turns to the *Tracking and status DB* only in case of unreachability of the container. A container may undergo a loss or a reduction of its connectivity for *environment obstacles*, that is, passing through a long tunnel or a mountainous region, being stored in a hub or parking area below a thick canopy or other containers, or for *power consumption issues*, that is, the transmitter may spontaneously switch off to a sleep mode so as to save its battery, while nevertheless not interrupting the recording of status variables.
Instead, *tracing* is a typically asynchronous process, as it is up to the container's initiative to periodically send fresh values for status parameters, so as to update the Tracking and status DB, independently of any query. If connectivity is lost for a duration that exceeds the sampling period, the container may store a sequence of observations and their timestamps to be sent when the connection is re-established; in case of saturation of the local memory, only an appropriate selection of observations may be stored.
Among the several actors involved in the process (containers, drivers, customers interested in tracing their shipments), it is worth noting the role of the authority for public security, whose duty in risk management scenarios is to intervene directly or coordinate and monitor the intervention in case of accidents and other emergencies. Tracking the *current* position and status of all containers is indeed a fundamental precondition to effectively support any emergency intervention. However, in case of reduced communication capabilities due to device failures or conformation of geographic areas, tracing a container's trajectory shows helpful for *estimating* the current position with a given precision, which basically depends on how old is the latest bearing, on the direction of the container, and on the recent average speed.

4.2 Risk management

Risk management processes can be triggered by two events. In the first case, the sensing devices placed on the container register a risky pattern of values and, therefore, the Monitor service signals the emergency to the Risk Manager service. In the second case, an external event signals the occurrence of a risky situation, e.g, the driver of a container calls the headquarter of the company, signaling an accident. In both cases, risk management involves some standard activities, such as calling the local fire brigade or specific rescue companies signaled by the producer of the goods. However, the quality of the risk management strongly relies on the quality of the information provided to the entities intervening on the place where the disaster occurred. Firemen and the company performing the intervention need to be precisely informed about the type of transported goods, the exact location of the emergency, and the type of emergency (e.g., a fire or a flood) to best organize and coordinate their efforts. The Risk Manager service allows the transporter company to retrieve all the required information on the container and the transported goods before the actual intervention takes

place. Information, such as the exact position of the container or the degree of flammability of transported goods, is retrieved through the Risk Management service and communicated to firemen and private companies in charges of the actual intervention.
It has to be noticed that the information required by the Risk Manager service is provided only by the Monitoring service. Therefore, the risk management functionality introduced in this section are independent of the two types of implementations of the logical container services described in the previous sections.

The first case requires the asynchronous communication between the Monitoring and the Risk Manager services. In this case, the Asynchronous client of the Monitoring service signals an emergency to the Risk Manager. An emergency can be detected by assessing peculiar patterns of values provided by the sensing devices placed on the container. Examples of peculiar patterns can be:

- increasing temperature on a still container (the container is likely to explode);
- the capsize of the container coupled with a decrease of pressure (in this case, the the goods are likely to have flood outside the container);
- the capsize of the container coupled with an increase of temperature (in this case, we should be dealing with an accident and an explosion).

Since the Monitoring Service collects all the tracking and status variables, the signaling of a risky situation made by the Monitor can be enriched with all the information required to organize a proper intervention, such as the location of the container, temperature and pressure values, and the type and amount of transported goods, which can be easily retrieved through the codes reported on active tag and transmittable to the public security agency. Once transported goods are known, then the responsible of the intervention can easily organize the intervention for different kinds of emergency by consulting specialized databases available on the Internet, or industry specific manuals, commonly owned by specialized companies.

The second case considers an external event, such as a phone call made by the driver of the container, by the local police, or by people passing-by the zone of the accident, to signal an emergency. We assume that the local police department forwards this signal to the transporter. The objective of the Risk Manager is, in this case, to retrieve the correct information about the damaged container (location, transported goods, amount, etc...) in the quickest possible way. This time, the Risk Manager exploits the synchronous communication between our infrastructure and the container services. The Risk Manager can query the Monitor service to identify the damaged container. When the signaling call does not identify the container immediately, the Risk Manager can identify the container by looking for containers currently located in the area in which the emergency has been signaled or by looking for those containers that are not available to be queried, under the assumption that this occurs because of a severe damage. Once the container has been identified, the Risk Manager retrieves the information about the container through the Monitoring service. The *selection* component of the Monitor is in charge of selecting the most suitable way to query the container. If the container cannot be queried because of a severe damage, then the selection component retrieves the most recent data provided by the container and stores them in the Tracking and status DB. Otherwise, the selection component directly queries the container through the abstraction provided by the logical container.

In both case, the Risk Manager relies on a Geographical Information System (GIS) to properly identify the location of the container from the data provided by the container service. The GIS is also useful for providing other important information for the organization of the rescue intervention, such as signaling the presence of a river, groundwater, a public place, or a gas station nearby the location in which the emergency is taking place.

5. RELATED WORK

Contributes to the research on the emergent field of crisis and emergency management derive from a variety of fields, such as decision support systems, pervasive computing, Web services and multiagent systems. Research in this field is mostly focused on medical emergency and disaster management. Specifically, disaster management has emerged as an hot research topic in the last few years, as a consequence of the occurrence of well-known natural disasters, such as the 2004 tsunami in Southern Asia or hurricane Katrina in the US [5]. Generally, researchers highlight two main objectives for IT applications deemed to support emergency management. On one side, it is fundamental to have an information management infrastructure that allows the collection of all the appropriate and required information before the emergency takes place. On the other side, crisis and risk management requires a pervasive IT architecture that can support people on the field once the disruptive event has occurred [6]. Among the two, this second challenge appears as the most critical, and it involves the adoption of emergent technologies, such as lightweight and highly configurable multiagent systems, service oriented solutions in mobile environments, or ad-hoc sensor networks [7, 8].
In the context of logistics and transportation, the research is focused on the first objective of the crisis response management, that is, the gathering of important information that can be useful to face the emergency once it has taken place. Most of the research, in fact, concerns the optimal scheduling of transportation routes, based on resource availability and traffic information [9, 10]. To the best of our knowledge, little has been done, in the specific context of logistics and transportation, concerning the problem of supporting rescue teams on the field in the case of accidents related to dangerous goods transportation. Risk management and its related processes are one of the most critical aspects to be considered in the transportation of dangerous goods. A not properly managed emergency is likely to increase the losses of the transporter organization in terms, for instance, of increased costs for requalifying a contaminated soil and increased costs in facing legal issues with the producer of the transported goods, final customers, or public institutions.
As for the technological support, in the last years several hardware and software pervasive technologies have been presented for logistics environments. Most of them address

RFID technology [11], and we can find many works that solve specific issues of this technology. For example, [12] addresses optimized management of data streams (from RFID readers) to compute complex business events, or [13] tackles localization problems.
Many other hardware technologies have been applied in logistics: Ultra Wide Band technologies [14], for example, can realize networks of devices as well as localization on bigger areas than RFIDs. Wireless Sensor Networks [2] also are gathering increasing success for the development of pervasive systems for monitoring tagged objects.
In these last years also several approaches for integration of data from pervasive devices into conventional information systems have been developed. We can identify two kinds of integration: horizontal integration is aimed to make different pervasive devices technologies interoperate, while vertical integration is aimed to develop abstractions of devices at high level into the information system hiding the interaction with the physical device.
We can mention EPC Global standards [15] for the integration of RFID systems into cross-enterprise information systems. Software section of this set of standard defines a reference architecture to generate application level events starting from data from RFID readers, thus supporting the sharing of data among enterprises along the business chain. Parts of these standards have been implemented by commercial solutions like [16].
There are many database-like approaches for integration both horizontal and vertical. These approaches abstract the pervasive space whose data are gathered as if it were a database. One of the first approaches based on Wireless Sensor Networks is TinyDB [17], but recently some similar approaches have been developed to take also into account the quality of supplied services, the interoperability among different kinds of devices, and the existence of actuators (besides sensors) [18].
Another interesting category of systems are the platforms that perform virtualizations of single pervasive devices to allow the developer to treat physical objects as information system objects and to develop applications using them. [19] proposes a middleware that allows the developer to deploy virtual sensors and treat them as conventional programming primitives, while [20] presents a hardware platform where different devices can be plugged together and a software platform that abstracts devices as services, thus performing both vertical and horizontal integration.

6. CONCLUSIONS AND FUTURE WORK

The paper presents our first experiments on abstracting physical devices with Web services to obtain a flexible service-centered infrastructure for advanced logistics. The first results are promising and the capability of abstracting physical devices to separate the business logic from the actual technology used to sense the elements of interest is much more general than what presented here. We choose to adopt off-the-shelf devices to be installed on containers. In this way, with hundreds of euros per transportation unit, companies can easily afford our solution.
Currently we are refining the architecture, we are concluding the implementation of supporting tools, and we are also trying to apply our ideas on other problem domains where different sensing devices are usually employed. Moreover, we are studying how to use WSDM to augment our infrastructure with well-known standards.

Acknowledgments

This work has been partially funded by Italian MIUR-FAR EASyLog Project.

7. REFERENCES

[1] Baresi, L., Beretta, P., Fraccapani, R., Ghezzi, C., Pacifici, F.: Towards a model-driven approach to develop applications based on physical active objects. apsec **0** (2006) 173–182
[2] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System archtecture directions for networked sensors. ACM SIGPLAN Notices **35**(11) (2000) 93–104
[3] : Sun spot technology website. (`http://www.sunspotworld.com/`)
[4] : Java 2 micro edition website. (`http://java.sun.com/javame/index.jsp`)
[5] Bui, T., Sankaran, S.: Foundations for designing global emergency response systems. In: Proc. 3rd Int. ISCRAM Conference, Newark (NJ). (2006) 72–81
[6] Manoj, B., Baker, A.H.: Communication challenges in emergency response. Communications of the ACM **50**(3) (2007) 51–53
[7] Fiedrich, F., Burghardt, P.: Agent-based systems for disaster management. Communications of the ACM **50**(3) (2007) 41–42
[8] Dwarkanath, S., Daconta, M.: Emergency services enterprise framework: A service oriented approach. In: Proc. 3rd Int. ISCRAM Conference, Newark (NJ). (2006) 298–304
[9] Tomas, V.R., Garcia, L.A.: A cooperative multiagent system for traffic management and control. In: AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. (2005) 52–59
[10] Dorere, K., Calisti, M.: An adaptive solution to dynamic transport optimization. In: Proc. 4th International joint conference on Autonomous agents and multiagent systems. (2005) 45–51
[11] Finkenzeller, K.: RFID handbook. Wiley Hoboken, NJ (2003)
[12] Wang, F., Liu, S., Liu, P., Bai, Y.: Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams. Proceedings of EDBT (2006) 588–607
[13] Liu, X., Corner, M., Shenoy, P.: Ferret: RFID Localization for Pervasive Multimedia. Proc. of the 8th Ubicomp Conf., Sept (2006)
[14] : Ubisense precise real-time location. (`http://www.ubisense.net/`)
[15] : Epc global network. (`http://www.epcglobalinc.com/`)
[16] : Ibm premises server. (`http://www-306.ibm.com/software/pervasive/ws_rfid_premises_server/`)
[17] Madden, S., Franklin, M., Hellerstein, J., Hong, W.: TinyDB: an acquisitional query processing system for sensor networks. ACM Transactions on Database Systems (TODS) **30**(1) (2005) 122–173
[18] Xue, W., Luo, Q., Ni, L.: Systems Support for Pervasive Query Processing. Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)-Volume 00 (2005) 135–144
[19] Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. Proceedings of the 32nd international conference on Very large data bases (2006) 1199–1202
[20] King, J., Bose, R., Pickles, S., Helal, A., Vander Ploeg, S., Russo, J.: Atlas: A Service-Oriented Sensor Platform. Proceedings of the First International Workshop on Practical Issues in Building Sensor Network Applications (in conjunction with LCN 2006), November (2006)