An architecture for distributed, event-driven systems to collect and analyze data in emergency operations and training exercises

Robin Marterer, Matthias Moi, Rainer Koch
University of Paderborn, Germany
{marterer, moi, r.koch}@cik.uni-paderborn.de

ABSTRACT

In order to perform serious research on reliable data from emergency operations and trainings, technological support is essential. Therefore we present an architecture for distributed, event-driven systems for the collection and analysis of data in emergency operations and trainings. The logical as well as the technical architecture will be presented. Most important design decisions, e.g. regarding extensibility, will be described. The architecture has been implemented as a system, which is composed out of a core server and distributed sensors sending data. The system is running since two years in two big European cities.

Keywords

Event-Driven Service-Oriented Architecture, Complex Event Processing, data collection, data analysis, distributed systems, emergency operations, training exercises

INTRODUCTION

In the domain of crisis response, emerging hardware and software systems are available, but rarely integrated. This makes the utilization of theoretically available data unessential complicated. Heterogeneous data from several sources has to be digitalized or synchronized manually during (online) or after (offline) an emergency. In the online use case, data is requested for emergency management, particularly decision making. The offline use case is important for de-briefing purposes, e.g. after action reviews.

In this paper, we present the system architecture and its implementation of the FP7 research project PRONTO. It is based on the principles of Event-Driven Service-Oriented Architectures (ED-SOA) (Mühl, Fiege and Pietzuch, 2006; Taylor, Yochem, Phillips and Martinez, 2009) and Complex Event Processing (CEP) (Luckham, Niblett, 2007; Etzion, 2010). The system has been primarily designed for the two main use cases near-realtime decision support during emergency operations and data collection during emergency operations and training exercises for de-briefing and data analysis. The novelty of PRONTO is, that every change in the real world within an emergency operation context is anticipated as an event. Events are gathered from various kinds of sources, such as hardware devices (e.g. GPS) and user interaction with the PRONTO system as well as from audio and video data streams (Krausz, Bauckhage, 2011). Audio includes speech communication by radio between fire officers as well as surrounding sounds hinting at e.g. dangers. The PRONTO system aggregates events from all these kinds of sources and filters out relevant information for the users. PRONTO assesses the resource management in emergency rescue operations and provides a best available view on the operational picture during operations and training exercises. Different from, other projects like the CRISIS project, PRONTO does not implement a train-on-demand simulation platform based on the concept of serious games. PRONTO captures real existing concepts and processes of current emergency management systems and enriches them with available information from arbitrary events. This also takes effect for the PLAY project. While PLAY is more focused on the observation and the analysis of arbitrary events in different contexts, PRONTO is more focused on the existing processes, especially the resource management and the provision of an adequate operational picture. The main outcome of PRONTO is an improvement of decision support in emergency operations by the integration of novel technologies like CEP.

After presenting the overall architecture the essential Application framework with two selected applications for decision support in emergency rescue operations will be presented. The paper will close with results, including opportunities and hurdles of the system architecture.

SYSTEM ARCHITECTURE

The considered system can be described in terms of server components and client components, which are organized in subsystems (see Figure 1). The Integration subsystem connects all other subsystems with each other. The Application subsystem, which acts as a frontend to the user, is realized as an application framework, capable of managing independent web-based applications. The terminology used in this paper is aligned with the Event Processing Glossary of the Event Processing Technical Society (EPTS) (Luckham and Schulte, 2011). The architecture was designed with respect to seven fundamental requirements: (1) Recognition of events, calculated out of incoming data from several sources, (2) Near real-time performance (occurrence of an event until reaction: < 3 seconds), (3) Robustness, (4) Extensibility, (5) Failure tolerance, (6) Ability to show the UI on arbitrary devices, (7) Having only one (semantic) data store and many different views. Due to its modular design, arbitrary components can be added, replaced or removed without much effort. All components are connected through the Message-Oriented Middleware (MOM) component. Following the publish-subscribe pattern, components can act as event producers or event consumers. The MOM component permits components to be distributed over heterogeneous platforms (operating systems and protocols).

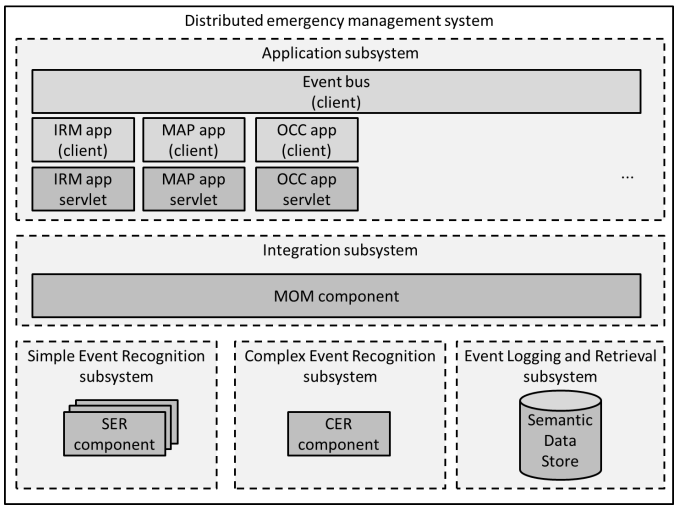

Figure 1: System architecture

Event producer components create simple raw event objects and publish them for further processing via the MOM component to the system. Event producer components can be hardware sensor components (e.g. for GPS, acceleration or CAN-Bus data) or pure software components (see Application subsystem), which implement the event producer interface of the MOM component. Event consumer components can subscribe themselves for selected event types at the MOM component and are then informed about event object occurrences of this event type. Event consumers can also be hardware actuator components (e.g. traffic lights or acoustic warnings systems) or pure software components (see application subsystem)), which implement the event consumer interface of the MOM component. Event processing agents (EPA) calculate simple derived or complex events on input events. Most EPAs calculate simple derived events out of simple raw events. One central EPA (Skarlatidis et al., 2011) takes all kinds of simple events as input and calculates complex events as output. All event objects are being persisted to a central semantic data store, which contains the system's status at all times. The data store and therewith the system's status is also accessible via the MOM component.

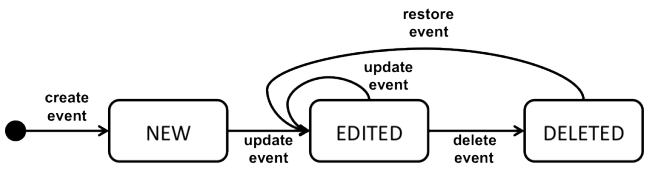

Figure 2: State machine as basis for the naming scheme of event types

Event Types and Event Categories

Each event object in the regarded system architecture, belonging to exactly one event type, contains at least a unique ID and a timestamp. Events, which represent state changes of defined entities also contain a entity ID. Based on the CRUD (create, read, update, delete) concept from relational databases and the ideas from finite state machines (see Figure 2), each entity in the system can be described by its state and by state transitions (i.e.

events). An entity can be in the states {START, NEW, EDITED and DELETED} and can be transferred into another state by {create, update, delete and restore} events. This concept enables the system to store a complete history of events belonging to arbitrary entities.

EVENT-DRIVEN APPLICATION FRAMEWORK

The event-driven application framework for emergency operations and training exercises encapsulates all components, called applications (apps) in the application subsystem (see Figure 1). These apps are directly available to the user through a web-interface. The overall goal of these apps is increasing the awareness of the decision makers and supporting their decision making process as well as supporting de-briefing and training.

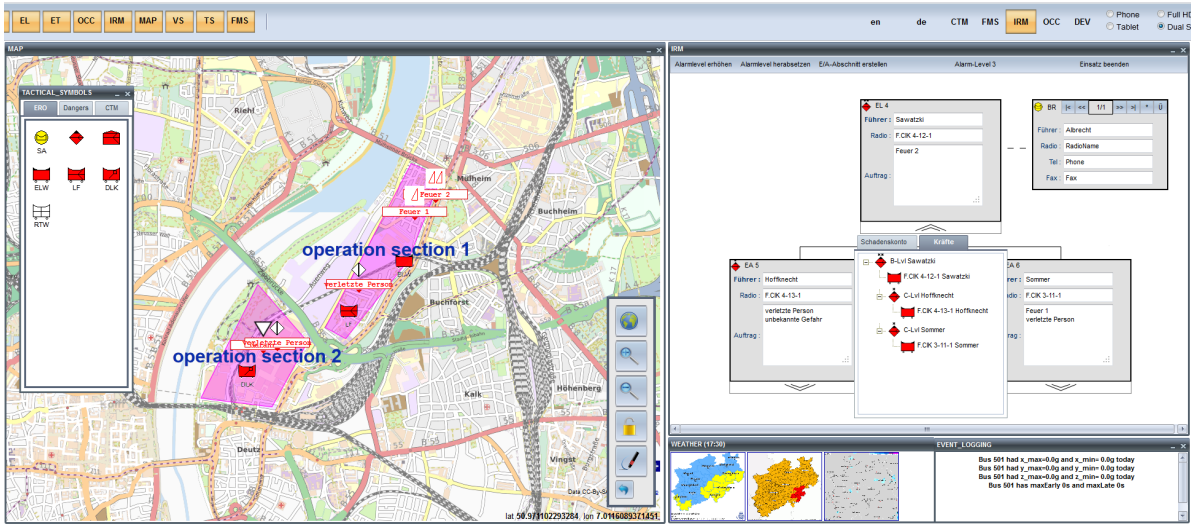

Figure 3: Application framework UI

All apps in the application framework are built on the Google Web Toolkit[2] (GWT) and its smartGWT[3] extension library, which allow the development of rich and complex front-end applications completely written in JAVA (and compiled to Java Servlets, JavaScript, AJAX technology). Each app is divided into a client and a server part, where the client-part communicates with the server-part via RPC. All apps are able to communicate via event messages through the MOM component and via event messages through a client-side event bus. Events sent to the MOM are available system-wide, whereas events sent over the client side event bus will not leave one instance of the application framework. Update events are used for pushing the system state to all client instances continuously. The Operation Control Centre (OCC) app is used for creating operations in the system. The dispatcher (i.e. a defined user role) must enter the location and a keyword describing the emergency as well as an assignment of resources (e.g. fire brigade units, vehicles). After one click, an 'operation initiated event' will be send to the MOM component. The OCC personnel could also use the MAP application in order to get a geographic impression of the emergency. After that, the Officer-in-Charge (OiC) (another user role) for that operation has access to his resources via the Intelligent Resource Management (IRM) app and the MAP app. Other user-role-dependent supporting apps are e.g. the Weather app, which shows the current weather and associated warnings (e.g. storm, heavy rain) for the selected region and the Event Logging app, which realizes an online view of all event objects occurring in the system (e.g. section created, vehicle arrived, vehicle entered operation section etc.).

Intelligent Resource Management Application

The IRM app is an application for high-level organizational and strategic decision support in emergency operations for fire brigades (see Figure 3, right). Its general concept of representing the management structure as well as operation sections during emergency operations is based on the 'tactical organisational table' (german: 'Taktische Arbeitstafel') of the 'State fire service institute North Rhine Westphalia'[4]. The IRM app is a multi-

[2] http://code.google.com/intl/de-DE/webtoolkit/

[3] http://code.google.com/p/smartgwt/

[4] http://www.idf.nrw.de/

user application that represents the structural organisation of fire brigades in emergency rescue operations. Officers are able to manage resources to improve the view on the operational picture and therefore the situational awareness on the logical view of operations. The IRM app implements the event producer (e.g. change in the management structure) and the event consumer interface (e.g. a geo-referenced danger occurs). Each change in the hierarchy, the creation of operation sections as well as occurring dangers, e.g. recognized in the backend, changes the state of the system. E.g. if a unit is assigned to a section, an event object is created that transports this information to inform other components as well as apps in the whole system. As someone already thought, such an action will have an impact on the availability of resources in the whole system. The simplest organisational structure exists only of one unit represented as one node in the hierarchy. By expanding the management structure, the operation is divided into sections. Sections may be geo-referenced or logical (e.g. guarantee water supply). Officers are able to rearrange vehicles or units between different existing management levels. Occurring dangers, which may have been reported automatically, are recursively populated from the bottom (identified geo-referenced section) to the top in the hierarchy. The idea is that the OiC has to have the overview of all existing dangers and notes, while officers at the same management level have disjoint information – only superiors have access to information of subordinated units. If a danger occurs, it will be assigned to a section on the deepest possible level and propagated up to the OiC.

MAP Application

The MAP app (see Figure 4, left) provides basic features as a) jump to predefined locations, b) search for and jump to locations, c) zoom in and zoom out, d) drag the map to locations, e) create, edit and delete tactical symbols on the map by drag and drop, f) create, edit and delete zones with labels, g) connect zones to operational sections, h) create, delete, show and hide different (base) layers (full KML/WMS support), i) show the position of entities, which are connected to the system, e.g. by GPS devices, j) highlight zones, which are entered by entities, k) lock the drag feature, l) show a mini overview map, m) show lon, lat coordinates of the current position, n) filter the presented information by operations. There can be arbitrary many instances of the MAP app, e.g. for the presentation of several locations on a wall with help of a beamer. The MAP app is implemented on the basis of GWT-OpenLayers and fully integrated with GWT/SmartGWT. Each interaction with the MAP app produces one or more events (e.g. ‘create section’, ‘annotate map’). The events are received by other apps and act as input for SER and CER. For example, a ‘create section’ event results in a logical representation in the IRM app and as input for ‘zone detection SER’, i.e. calculating a ‘zone enter event’ out of a ‘vehicle position changed’ and a ‘create section’ event, whenever the GPS position of both overlap. Next, if the vehicle entity and the zone entity belong to the same operation, an ‘arrived at scene’ event is recognized. In addition, danger reports by radio from first responders can be added to the calculations as events.

RESULTS

The evaluation of the PRONTO concepts and the PRONTO system is still ongoing. Fire officers of different grades are encouraged to interact as probands with the system during a predefined realistic scenario and to answer questions in structured interviews. The scenario was also chosen with respect to technical aspects. It guarantees that all hardware and software components of the PRONTO architecture are used. The scenario starts with a smaller fire in an old people’s home, which becomes incrementally bigger, raising the complexity of the situation (with multiple dangers occurring and injured people). The proband acts as OiC and has to manage the situation. As one result of the evaluation, the term ‘event’ must be handled carefully. Most of the probands associate the ‘damaging event’ (i.e. the emergency) with it. But they also agree on a more granular definition of this term. The PRONTO system was adjudged to be very helpful especially in large- and mid-scale emergencies, in order to maintain a common operational picture, enriched by real-time information hinting at, e.g. critical events, which could avoid mismanagement of an confusing emergency. Using the ‘tactical organisational table’ as a basis for system respectively UI design was stated as reasonable. Events must be visualized prominently. A green blinking window containing textual and visual information exposed to be adequate. This window is expected to remain until the user confirms it. The PRONTO system was stated to be user-friendly. This was argued by its minimalistic design, i.e. the number of UI elements is reduced to a minimum. The event-driven approach was also adjudged to be very helpful for de-briefing and follow-up of emergency operations and practical training exercises. It is important to find appropriate filter criteria, which define the event types which are shown to the user and which are not. The PRONTO system archives all events with timestamps, so that fire officers can reflect on past emergency operations or practical training exercises. This data also allows for statistical elicitations. Both was stated as helpful. During the evaluation, the PRONTO system turned out to be suitable as a training system as well. This third use case could be evaluated separately in a next step. The event-driven approach, resulting in the loose coupling of components, enabled us to have a very lightweight, but effective software development process in PRONTO. Around 60 event types have been implemented.

CONCLUSION

The paradigm of ED-SOA and CEP in combination with the presented architecture results in the fact, that each single event is at least tagged with a timestamp. In combination with the CRUD (create, read, update, delete) concept from relational databases and the ideas from finite state machines, each entity in the system can be described by its state and by state transitions (i.e. events), calculated by the SER and CER components and persisted by the Semantic Data Store. PRONTO aggregates events from various kinds of sources and improves emergency management by a strong data base, including events from audio, video, user interaction and hardware sensors. Having events with timestamps enables the user of the system to switch back in time and to reconstruct situations from emergency operations and practical training exercises (e.g. for debriefing purposes). Furthermore, exports to Learning Management Systems (e.g. ILIAS) and long-term analyses as well as the calculation of statistics on the recorded data are possible. The regarded system is based on standard-conform protocols and for that reason extensible by arbitrary hardware and software components. It has great potential for research on reliable and exact data. Extensibility has been proven by several add-on components, implemented as case studies (e.g. a tablet-computer-based event producer solution, export functionalities). A third use case, that has not been evaluated yet is the simulation of emergency operations on the basis of collected or complete artificial data which could be (re-)fed into the system. First evaluation results are available, although the final evaluation of PRONTO is ongoing.

In this paper, we presented an architecture for distributed, event-driven systems for collecting and analyzing data in emergency operations and training exercises. The implemented system is fully functional and partly integrated into the end-users' infrastructures in two big European cities. The installation comprises very different kinds of event producers, which send various types of simple raw events to the system: e.g. GPS, vehicle status data, speed, acceleration, weather and user interactions. The system calculates simple derived and complex events as output. Different versions of the system are running since almost two years processing and archiving event objects. Simple derived events and complex events are computed and visualized in near real-time by several web-based applications, built upon the concepts of the 'tactical organisational table' of the 'State fire service institute North Rhine Westphalia'. Further work will be done concerning the analysis of collected data and the extension of the system by more hardware and software components as well as the final evaluation of the system.

ACKNOWLEDGMENTS

This work has been partly funded by the European Community's Seventh Framework Programme (FP7-ICT-2007-3. Cognitive Systems, Interaction, Robotics – STREP) under grant agreement n° 231738.

REFERENCES

1. Mühl, G., Fiege, L. and Pietzuch, P. R. (2006) – Distributed Event-Based Systems, Springer, Berlin Heidelberg New York.
2. Taylor, H., Yochem, A., Phillips, L. and Martinez, F. (2009) – Event-Driven Architecture: How SOA enables the Real-Time Enterprise, Addison-Wesley, Boston.
3. Luckham, D. (2007) - The power of events: An introduction to complex event processing in distributed enterprise systems, 5th ed., Addison-Wesley, Boston.
4. Etzion, O. and Niblett, P. (2010) - Event Processing in Action, Manning, Stamford.
5. Luckham, D. and Schulte, W. R. (2011) - Event Processing Technical Society: Event Processing Glossary – Version 2.0
6. Pottebaum, J., Artikis, A., Marterer, R., Paliouras, G., Koch, R. (2011) - Event definition for the application of event processing to intelligent resource management, *Proceedings of the 8th International ISCRAM Conference*, Lisbon, Portugal.
7. Friberg, T., Birkhäuser, B., Pottebaum, J. and Koch, R. (2010) - Using Scenarios for the Identification of Real-World Events in an Event-Based System, *Proceedings of the 7th International ISCRAM Conference,* Seattle.
8. A. Skarlatidis, G. Paliouras, G. Vouros, A. Artikis. Probilistic Event Calculus based on Markov Logic Networks, in Proc. Of 5th International Symposium on Rules (RuleML 2011) Part 2, Fort Lauderdale, USA, 03-05 November 2011.
9. Krausz, Barbara; Bauckhage, Christian. *Automatic Detection of Dangerous Motion Behavior in Human Crowds,* in Proc. of AVSS 2011.