Dynamic Emergency Management

STEFFEN HAUSMANN, SIMON BRODT, FRANÇOIS BRY, LUDWIG-MAXIMILIAN UNIVERSITY OF MUNICH, MARCO BETTELINI, AMBERG ENGINEERING AG

Large infrastructures become technologically more and more complex and interdependencies between different infrastructures increase. As a consequence complex infrastructures become more vulnerable to emergencies. The personnel in charge must be supported in the challenging task of identifying and assessing an emergency situation and in choosing an appropriate reaction. This calls for a system which continuously analyses the available sensory data for emergency situations, making predictions for the future development of an emergency and recommending proper reactions to the emergency.

SUMMARY

The interdisciplinary three-year EMILI project has shown that simulations based on simplified physical models, complex event processing and standard hard- and software can be combined into a powerful emergency management system. First the resulting system can timely provide the personnel in charge with instructions and predictions relevant to a current emergency situation and second offers an environment for training the staff for emergency management. Prototypes for three different use-cases, namely a metro station, an airport and a power-grid, have been implemented and successfully tested.

Keywords: Large Infrastructures, Dynamic Emergency Management, Physical Simulation, Complex Event Processing, Training

1. Advantages of Dynamic Emergency Procedures

Large infrastructures, such as airports and large metro stations, become technologically more and more complex. Advanced equipment provides additional, more detailed information, which must be processed in a suitable manner. Interdependencies increase and, as a consequence, the vulnerability of complex infrastructures in emergencies also increases. This evolution calls for a new-generation of SCADA (supervisory control and data acquisition) systems based on new technologies which enable a better emergency management. Today's emergency procedures are reactive but static. They consist in complex and lengthy instructions to be learned in advance by the staff in charge during emergencies. Indeed, in emergency situations, it would be unpractical because it would be much too time consuming to expect the staff to consult such instructions before acting.

Dynamic emergency procedures reacting to situations detected by sensors and/or reported about by equipment, such as staircases or lifts, and expressed as executable software are nowadays possible, though. Dynamic emergency procedures make it possible, firstly to timely provide the personnel in charge with instructions and predictions relevant to the current situation in cases of emergency, and secondly to train this staff for emergency management.

This article reports on the main outcome of a three-year international research project funded by the European Commission, called EMILI (for “Emergency Management In Large Infrastructures)”, within its 7th Framework Programme, which has worked out the conceptual software components for dynamic emergency procedures. The EMILI project has been presented in the article [1].

The suggested approach can be deployed for the fire security of large infrastructures such as airports, train stations, metro stations, concert halls, or theatres. The approach relies on the following technological building blocks, some of which are generic, that is, independent of the infrastructure:

- simplified smoke propagation and egress models delivering real time estimates, the principles of which are generic but that require instances specific to the infrastructure considered,
- an infrastructure of generic sensors of the kind currently used requiring a deployment specific to the infrastructure considered,
- reaction rules in a generic language requiring a programming specific to the infrastructure considered,
- a generic run time system for the (generic) reaction rule language conceived as an extension of a (generic) relational database system.

The proposed approach comes at an undeniable and in most cases significant, cost because of its many aspects dependent on the respective infrastructure. A considerable part of this cost is however necessary for realizing today's static emergency procedures. Furthermore, since the approach proposed to dynamic emergency procedure can be used for both, testing the procedures and for training, part of its costs can be seen as training costs. Finally and very importantly, the approach proposed holds the promise of reducing casualties.

2. Real Time Simulations

2.1 Why Real Time Simulation?

SCADA (supervisory control and data acquisition) systems can process a large amount of information and present it to operators in an ordered manner. This information is static in nature, since it presents instantaneous values for the relevant physical data, such as smoke distribution or operating parameters of the equipment. However, SCADA systems usually provide less or no information on the origin and the evolution of the scenario at hand. Where did a fire originate? Are persons directly and immediately endangered or there is time for a more controlled reaction? What is the likely evolution of fire intensity and smoke propagation? All these issues have a direct and immediate impact on optimum emergency management and should be properly accounted for.

The interpretation of the large bulk of data is frequently left to the operator, who might or might not have an adequate background for this challenging task. As a competent

professional, he will certainly have a proper knowledge of all technical systems, predefined scenarios and standard procedures. His technical background will, in most cases, not include complex and specialized issues such as fire and smoke dynamics. Additionally, his work is greatly complicated by a number of different and urgent tasks to be performed: communicating with users and staff, alerting internal and external emergency services, redirecting persons and vehicles, initializing emergency procedures and much more. There is simply no time and insufficient background for reasoning on causes and likely scenario evolution. Critical decisions related to emergency management are therefore based on a reduced subset of simplified static data and simple rules in most cases.

Optimum emergency management is intimately related to the future evolution of the scenario. In traffic infrastructures, rescue strategies primarily depend on smoke propagation. Therefore smoke-management strategies must be adapted to the selected self-rescue and intervention strategies. The initial strategic choices can, in most cases, no longer be corrected, even if they are far from optimum. This was recognized in the early phases of the EMILI project and this issue was addressed with a specific effort towards the development of physical models for real-time simulation, specifically conceived for emergency management in traffic infrastructures. A similar approach can be adopted for different infrastructures, adapting the model subset to the specific characteristics and requirements.

Emergency scenarios in transport infrastructures must focus on two key processes: person movement and smoke propagation. In the initial phase of emergency in a traffic infrastructure, before the arrival on site of the specialized services, the users are alone. This is the self-rescue phase, which will typically last 5 to 15 minutes, depending on the specific conditions. Some level of communication with the users, through emergency phones, loudspeakers etc., is possible but limited. Proper technical measures, such as train redirection and emergency ventilation, must be adopted immediately and are decisive for the outcome. Time is therefore a key factor. The use of specially conceived physical models allows simulating the likely evolution of the conditions within the critical infrastructure and the position of the escaping persons in real time. Different emergency management scenarios can be analyzed and compared in a very short time. This allows

founding strategic emergency-management decisions on rational bases and represents an important step towards optimum emergency management.

2.2 iSEM – Real Time Simulation for Emergency Management in Traffic Infrastructures

A number of excellent and well-validated software tools are available for analyzing the self-rescue process and for simulating the evolution of fire, smoke and air pollutants. Such tools suffer from two main drawbacks: the need for comprehensive pre- and post-processing and relatively long processing times. The use of standard tools would therefore require a number of interfaces, many adaptations and unrealistically large computational resources. It was therefore decided to develop, within EMILI, the dedicated software package iSEM ("intelligent Simulation for Emergency Management") for this specific purpose. iSEM is based on a simplified but physically sound set of equations for person and train motion, aerodynamics, temperature distribution and smoke propagation.

The geometric discretization is carried out based on a network approach, which allows a very intuitive representation of the geometry at hand and a seamless integration with its ontological model. Figures 1 and 2 show the geometric discretization for a generic metro station. The main output of the simulation is represented by the time-dependent evaluation of the following parameters for every node: number of persons, smoke concentration and visibility, temperature. Details on iSEM's approach and on its validation are provided in [1].

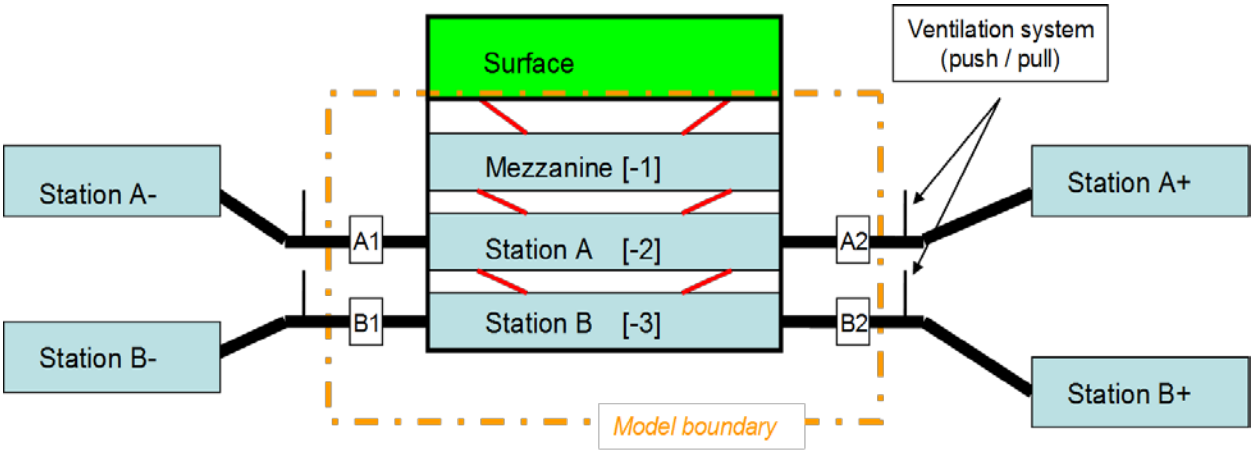

Figure 1: Physical representation of a generic metro station.

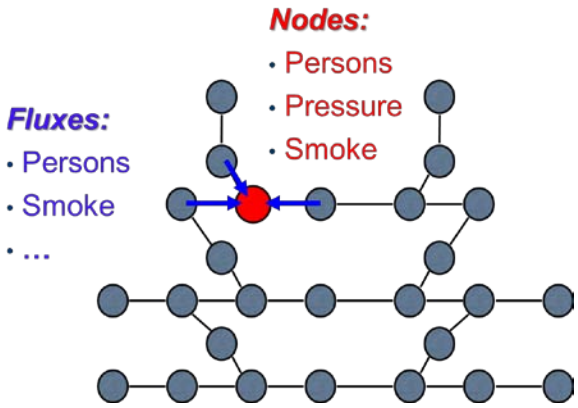

Figure 2: Network representation of a generic metro station.

2.3 Real Time Physical Simulation for Emergency Management

In a first step, physical models are used for the real-time simulation of the most likely evolution of the system, based on a known set of initial conditions, subject to known boundary conditions and evolving according to predefined emergency-management strategies. The main objective is the identification of the best-suited strategies for the specific situation at hand. This involves the following steps:

- Identification of initial conditions, based on the available sensor data
- Simulation of the system's evolution, according to different reaction strategies (e.g. different evacuation strategies combined with appropriate operation modes of the ventilation system and further safety equipment)

- Comparison of results, primarily in terms of minimization of number of victims, and selection of the best-suited emergency-management strategy.

Figure 3 illustrates the integration of real-time simulations in the decision process.

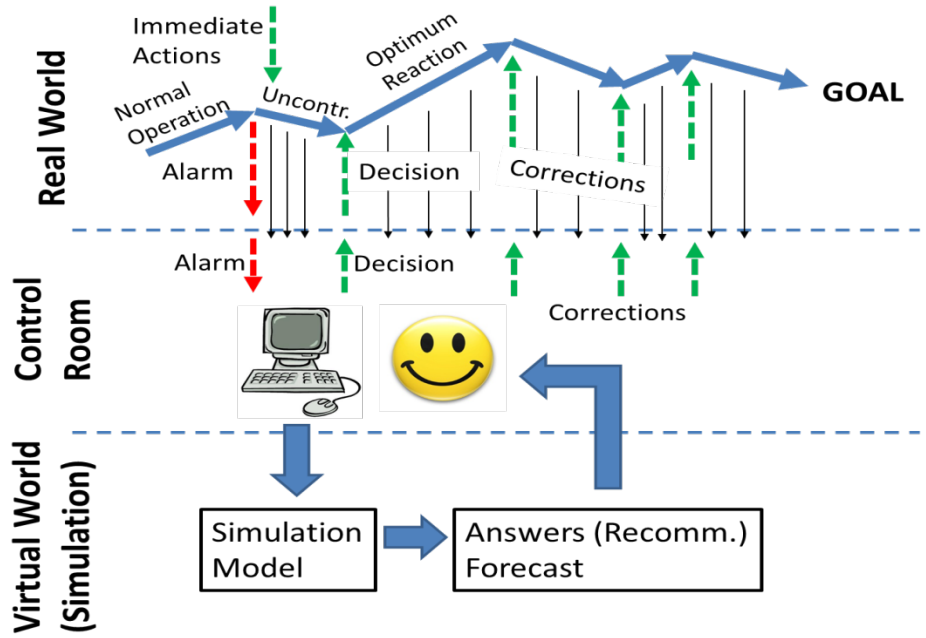

Figure 3: Principle of Real Time Physical Simulation for Emergency Management.

2.4 Real-Time Simulation for Training

Emergency situations are extremely rare in critical infrastructures. The availability of proper training capabilities therefore represents a key factor for the continuous improvement and verification of staff's readiness in case of emergency. iSEM can be used as a "physical engine" for training purposes. The trainees work on predefined emergency scenarios, where new dynamic elements can be introduced any time, and the system's state evolves dynamically according to the trainee's decisions. The consequences of inappropriate or too late decisions manifest themselves in form of unfavourable outcomes. Good decisions result in visibly better results. Owing to the physical models, the system's response is extremely realistic and greatly enhances the learning process. The trainee gains important knowledge also on the critical infrastructure's physical behaviour while running training scenarios. In this way the training leads to a continuous improvement of operators' readiness.

3. The generic reactive language Dura

Emergency management can substantially benefit from a new generation of event processing systems that support operators during exceptional situations and emergency situations. The language Dura [3, 4, 14] is a reactive event processing language that is intended to make a contribution towards more automation in emergency management. It has been designed to capture situation assessment and emergency procedures that are often only available in text form nowadays or implicitly in the form of experience of emergency managers.

3.1 Challenges

Near Real-Time Situation Assessment Emergency management requires quick and reliable situation assessment by means of expressive complex event queries evaluated in a continuous and timely fashion in order to detect relevant patterns in the vast amount of information that is provided by various sensors and actuators. Complex event queries derive higher level events that represent a concise abstraction which is desirable for human operators and suitable as input for the simulators.

Moreover, effective situation assessment needs to incorporate the predictions of simulations to obtain reasonable interpretation of often faulty or biased sensor data and as a reliable basis for the execution of actions. To this end, results of external simulators can be represented by means of events. But to discriminate the time a simulation event is observed by the system and the time at which the event is actually deemed to occur according to the simulation, the event query language must be capable of dealing with multiple independent time lines.

Besides the information that is provided by the sensors and simulators, the assessment of the current situation also needs to incorporate the state of the infrastructure. The interpretation of events may substantially differ based on the context in which they appear, e.g., the current operation mode of the station.

Formalization of Emergency Procedures Emergency management requites a notion for composite actions that are realized by means of basic actions executed by external actuators located in the infrastructure. However, composite actions that are

based on external actions have some rather unusual properties and requirements compared to internal database updates and remote procedure calls that are commonly considered by complex event processing languages.

Reactions are usually composed of several interrelated steps and are intended to accomplish a certain purpose, which cannot be realized by the mere execution of single actions. Therefore, the actually executed external actions are just means to affect the physical state of the infrastructure in a way that is related the purpose of the composite actions. Naturally, the success of the reaction is related to its purpose and not, or only indirectly, related to the success of the applied external actions.

Reactions can certainly fail to accomplish their intended purpose: the corresponding actuators can malfunction, the communication can be faulty, or the executed actions may simply cause unintended effects that fail to achieve the purpose of the action. Moreover, external actions affect the physical state of the infrastructure and are therefore often irreversible and cannot be easily compensated either. Accordingly, reactions must be well chosen before they are executed and thus a reliable situation assessment is mandatory.

Finally, the steps that are taken by an action build on each other and thus cannot be executed in an arbitrary order. To obtain a certain effect it is moreover often crucial to satisfy a certain timing between actions that exceeds the capabilities of simple sequences that are commonly available in imperative languages.

3.2 Emergency Management with Dura in a Nutshell

Dura models concepts related to emergency management by means of events, stateful objects, and actions. Events are used to represent sensor messages and to incorporate the result of simulations, stateful objects model states, e.g., the operation mode of the infrastructure, and (more or less static) topology information, and actions are used to modify stateful objects and to refer to physical actions of external actuators.

Dura is a rule-based language that distinguishes three kinds of rules. Declarative rules and action rules define new (complex) events and actions based on event queries and action specifications, respectively. They resemble database views that are continuously

evaluated and procedures that are automatically executed by the system. Reactive rules specify how to react to detected events. They make the transition between declarative rules that derive a high-level interpretation of the current situation and the corresponding reactions that are executed by external actuators.

Event Queries in Dura Complex event queries in Dura are specifically tailored to the particularities of emergency management. They uniformly integrate queries for events and stateful objects, provide expressive temporal dependencies that are capable of incorporating different time lines and maintain a clear separation of query dimensions which is desirable to obtain a high expressiveness of event queries.

Complex event queries are used in deductive rules to specify when a certain complex event should be derived. The following deductive rule uses a complex event query to derive certain alarms whenever the smoke concentration in an area exceeds 10% and the temperature rises above 50 degrees in the same area within two minutes. Note the clear separation of query dimensions, in particular the separation of the event composition with and and the specification of temporal dependencies with *within*.

```
DETECT
  certain-alarm{ area{var Area} }
ON
  and{
    event e: smoke{ area{var Area}, amount{var C} },
    event f: temp{ area{var Area}, value{var T} }
  } where { {e,f} within 2 min, C > 0.1, T > 50 }
END
```

Complex event queries are also used in reactive rules to specify when to execute certain actions. Under normal conditions, uncertain alarms are ignored. However, if there is an uncertain alarm and there have already been problems in the same area before, a warden is sent to clarify the situation. Accordingly, the following reactive rule initiates an action requesting the inspection of the area, if there is an uncertain alarm and the operation mode of the corresponding are has not been normal throughout the last three minutes. Note how queries of events and stateful objects are freely combined and that the query of the stateful object also considers past characteristics of the operation mode and not only the current one.

```
ON
  and{
```

```
    event e: uncertain-alarm{ area{var Area} },
    not state s: operation-mode{ area{var Area}, mode{"normal"} }
  } where { s valid-during from-end-backward(e, 3 min) }
DO
  action a: request-warden{ area{var Area} }
END
```

Action Specifications in Dura Complex action specifications in Dura are tailored to the properties of external actions. They incorporate expressive temporal dependencies that are discriminating the success and failure of actions to specify a sophisticated timing between actions. The success and failure of actions is determined by means of complex event queries that are not limited to incorporate solely the result of sub-actions but can also incorporate information from sensors. Finally, complex action specifications maintain a clear separation of query dimensions to obtain a high expressiveness.

The following complex action is intended to extract smoke from a certain area. To this end, the fire dampers of this area are opened and subsequently ventilators that generate an airflow that actually pushes the smoke out of the station are activated. Note that the specification of success of the action is directly related to its purpose, the extraction of smoke that manifests in the drop of the smoke concentration in the area below 10% within 1 minute. The event processing system ensures that the specified temporal dependencies in the where part are satisfied during runtime, in this case, that the ventilators are only activated 5 seconds after the fire dampers were opened successfully.

```
FOR
  adapt-ventilation{ var Area }
DO
  compound{
    action a: open-fire-dampers{ var Area },
    action b: activate-ventilators{ var Area }
  } where { succ(a) + 5sec <= init(b) }
    succeeds on {
      event e: smoke{ area{var Area}, amount{var C} }
        where { C < 0.2, end(e)-init(a) <= 60 sec }
    }
END
```

Semantic Analysis for Complex Actions Although desirable, the clear separation of different aspects of actions and the use of inequalities to specify temporal dependencies enable inconsistent action specifications that cause undesirable effects during runtime. To prevent unsound specification while maintaining the expressiveness

of temporal dependencies, a semantic analysis is applied to complex actions at compile time to ensure, e.g., that all sub-actions can be actually executed during runtime [14].

4. The Generic Run Time System Event-Mill for Evaluating Dura Rules

Event-Mill is an evaluation engine for reactive event processing languages like Dura. Event-Mill is based on a so-called “Temporal Stream Algebra” (short TSA) which generalizes the data model and the standard operators of relational algebra thus adapting it to data streams [5,6]. Event-Mill consists of three components: First, a compiler which translates Dura programs into TSA programs. Second, another compiler which transforms TSA programs into a set of incremental relational algebra expressions along with some so-called propagation functions [6]. Propagation functions are needed for proper synchronization and timing during the evaluation of incremental expressions. Third, a run time finally evaluating the original program by continuously executing SQL statements in a relational database like MonetDB [7]. The SQL statements are generated from the incremental relational algebra expressions produced by the compiler. Like Dura, Event-Mill and TSA have been designed to meet the specific requirements of emergency management. As a consequence, Event Mill and TSA significantly depart from other approaches based on, or related to, relational databases.

4.1 Challenges

Temporal Relations instead of Time Windows Emergency procedures for the detection and analysis of emergencies and for the reaction to emergencies usually impose temporal relations between the occurrence of events, the validity of states and the execution time of actions. In fact combining temporally (and frequently also spatially) correlated events into meaningful patterns is the basis for the detection and analysis of emergencies. For example, a fire can be detected accurately by combining the measurements/events from smoke and temperature sensors of an area within the last few seconds (see Section 3.2). Most existing complex event processing systems use time windows with fixed, data-independent bounds for expressing temporal relations. However, such windows can only express basic temporal relations and formulating the temporal relations specified in an emergency procedure by means of time windows

tends to be counter-intuitive and yields programs that, even in simple cases, are difficult to understand. The rules implementing dynamic emergency procedures need to be easy to verify and maintain and thus, temporal relations should be specified directly, that is, preferably without time windows.

Furthermore, windows with fixed, data-independent bounds cannot adapt to relevant “episodes” in the incoming data, for example, defined by the beginning and end of an emergency like a fire. The implementation of dynamic emergency procedures frequently requires analytic queries, like “the maximum smoke concentration within a room during a fire”, which should only be evaluated when actually needed, that is, in case of the emergency they specify. Both the starting time as well as the temporal extension of the analysis are determined by events (the beginning and the end of an emergency) and cannot be specified in advance. Moreover, the precise knowledge of the start and end of some period can be needed, for example in monitoring an evacuation. Thus, data independent window definitions are hardly suited for this kind of queries. General temporal relations should be used instead.

User-defined Timestamps for Composite Events In the area of emergency management the “right” choice for the timestamp of a composite event is not always obvious and may differ from query to query. Consider for example the following two rules: Rule 1: A temperature sensor is considered to be malfunctioning if there is a message from that sensor which is not followed by another message within 30 seconds. Rule 2: A precautionary fire alarm is raised if there is a potential fire detection in an area and no report from the responsible warden is received within 2 minutes after the potential fire detection. Both queries have the same basic structure, a positive event that triggers the rule if it is not followed by another event within a certain time-frame. However for the first query, the timestamp for the ``malfunction sensor'' event should probably be the time of the first missing messages, for example, 10 seconds after the last message from that sensor. By contrast the time for the precautionary alarm should be the time of the (potential) fire detection and not the time of the missing report. Therefore the user should be able to choose the most meaningful definition for the timestamps of a composite event on a query per query basis.

Multiple Time Lines Emergency management requires queries referring to timestamps according to various different timelines. In a simple case these timelines are just application *and* system time. Reconsider the rule – A precautionary fire alarm is raised if there is potential fire detection in an area and no report from the responsible warden is received within 2 minutes after the potential fire detection. The timespan of 2 minutes is defined between the detection of the potential fire, that is, application time, and the reception of the report, that is, system time. So as to ensure that no more than two minutes will elapse between the potential detection and a reaction to that detection, it is crucial that the 2 minutes period is defined in exactly that way, as transmission delays could otherwise block the emergency management system.

Further timelines are needed for simulations that help to predict the evolution of an emergency (see Section 2.1). Beside application and system time, events produced by a simulation carry a timestamp expressing the time at which a prediction is meant to hold. Multiple timelines are also useful for skipping the derivation of composite events which have become irrelevant by lapse of time, for example, due to prioritization in favor of more important queries.

Access to Static Relations Both detection and analytic queries frequently need to access static, or rarely changing, relations for interpreting the incoming events. For example sensor messages usually carry sensor identifiers, but rarely carry data indicating the locations of the issuing sensors. The location is however essential for correlating messages from different sensors. Therefore, location must be retrieved by an access to a database providing, for each sensor identifier, the location of the corresponding sensor. More complex topological information might be needed as well, for example, the neighbouring rooms of a burning area that are threatened by the fire. Identifying those rooms needs to correlate the event of fire detection with the static neighbor relation of rooms through a more involved database query.

Automatic Garbage Collection The evaluation of rules for emergency management needs to buffer, for some time, at least part of the incoming. As new data continuously arrive, old data that have become irrelevant need to be removed, that is, “garbage collected”. Manual garbage collection is error-prone and hard to maintain as it requires

an in-depth knowledge of the underlying evaluation system and a global overview of the whole emergency management program. Thus, garbage collection should be performed in an automatic and transparent way. The conditions distinguishing relevant and irrelevant data mostly depend on the temporal relations specified in the emergency management rules. Therefore, an automatic derivation of these conditions requires a sophisticated analysis of temporal relations.

Workload Bursts An evaluation system for emergency procedures must remain stable under workload bursts. This is especially important as emergency management applications have the property (which is unpleasant for the evaluation system) to require both, high reliability and good response times, at the time when the workload is at its highest, that is, during an emergency. The high workload during an emergency results first from an avalanche of irregular sensor readings which must be correlated to the corresponding emergency and second from the complex analytic tasks triggered by the emergency.

4.2 Temporal Stream Algebra (TSA)

Temporal Stream Algebra (TSA) generalizes the data model and the standard operators of relational algebra like selection, projection, join, set difference and grouping towards data streams [5,6]. Using the operators of relational algebra has a number of advantages: First, the operators are sufficiently expressive for formulating the emergency procedures encountered in all use -cases we examined. Second, the operators provide a good basis for the implementation of the high-level declarative language Dura. Third, data streams can be queried in a manner which is already known from relational databases (especially using the database query language SQL). Fourth, techniques and algorithms for the evaluation and optimization of relational algebra expressions can be reused for, or easily adapted to, data streams. The challenge of applying relational algebra to data streams is that some relational algebra queries cannot be evaluated on data streams, that is, they are invalid. TSA addresses this challenge based on three key observations:

First, data streams usually “make progress” with respect to some of their attributes. More precisely a data stream is “making progress” on an attribute if it eventually

exceeds any value for that attribute. Such an attribute is a so-called “progressing attribute” of the data stream. Usually, some of the timestamp attributes of a data stream are “progressing attributes”.

Second, the set of progressing attributes for a derived stream depends on the progressing attributes of the input streams of the corresponding query *and* on the temporal relations specified in the query. The achievable progress with respect to a progressing attribute of the derived stream depends on the available progress of the input streams and can be computed by so-called “propagation functions” which also depend on the temporal relations specified in the query.

Third, a relational algebra query can be evaluated incrementally on a data stream if its derived stream has at least one progressing attribute.

TSA exploits these observations in the following way: First, TSA propagates constraints on progressing attributes and temporal relations through the operators of relational algebra. Second, TSA provides an analysis of the propagated constraints identifying the progressing attributes of the derived streams of a query and also generating the corresponding propagation functions. Based on this, TSA can decide on the validity of a relational algebra query against data streams and can provide incremental expressions (also expressed in relational algebra) for every valid query. Furthermore, the results of the analysis can be used for garbage collection.

4.3 The Event-Mill Evaluation Run Time

The evaluation run time system Event Mill exploits the power of modern relational database systems like MonetDB [7] for an efficient evaluation of complex event queries. In particular, Event-Mill implements an out-of-order, bulk-wise processing of event queries based on the bulk-processing features of the underlying database system. The bulk-wise processing is essential for coping with bursts in the workload. It also increases the system’s efficiency. The evaluation of Event-Mill is asynchronous in the sense that the progress of different queries only has to be loosely synchronized. This allows for prioritizing the most important queries of an emergency by evaluating the less important queries at a lower frequency, or by even delaying such queries until the end of the emergency. Furthermore, the incremental evaluation of different queries can be

performed in parallel, making use of multi-core systems. Asynchronous processing is also an enabling feature for an efficient distributed processing which might be needed in some use-cases so as to scale with the size of an emergency management application with the size of a metro network.

Basically, the run time system takes an incremental relational algebra expression provided by TSA and transforms it into a semantically equivalent parameterized SQL statement. The parameters are placeholders for so-called progress values indicating the achievable progress of the corresponding derived stream and the available progress of the respective input streams. The actual progress values are computed using the propagation functions also provided by TSA. Each incremental evaluation step for a query consists of three phases: First, the propagation function is used to compute the achievable progress of the derived stream for the current incremental evaluation step based on the currently available progress of the input streams. Second, the parameters of the SQL statement for the query are bound to the obtained progress values and the SQL statement is executed. The execution of the SQL statements directly inserts the resulting tuples into the database table corresponding to the derived stream of the query. Third, the available progress of the derived stream is set to the values of the achievable progress computed in the first phase before the execution of the SQL statement.

5. Conclusion and Perspectives: Towards Composable Software for Modelling Complex Structures

An appealing characteristic of the approach presented in this article is its use of standard hardware and software.

As of hardware, standard smoke and other sensors and standard PCs/PLCs are sufficient. It would make sense for a large infrastructure to rely on two PCs networks fully independent of each other and both running the system. This would make it possible to immediately switch from one PC network another in case one network stops working. A metro station could be processed with a single PC (for each PC network), an

airport with as many PCs as areas like halls that, in case of an emergency would have to be evacuated individually.

As far as software is concerned, standard internet software and the software described in this article would suffice for a large infrastructure like an airport, a train station, or a metro network.

The main limitation of the approach proposed is that it requires a specific programming of sensor-based detection and of reactive emergency rules. A next step which would be worthwhile investigating is a software build up from composable pre-defined, that is generic, components that could be adjusted by the setting of suitable parameters to the specificities of a wide range of large infrastructures. Such an approach seems possible since, provided they are sufficiently recent, large infrastructures are very similar in their structures.

6. Acknowledgements

The work presented in this article is part of the research project EMILI (2010–2012, http://www.emili-project.eu/) funded by the European Commission under grant agreement number 242438 within its 7th Framework Programme.

7. References

[1] M. Bettelini, N. Seifert, and F. Bry: Innovatives Sicherheitssystem für U-Bahn-Stationen" (in German) IM - Fachzeitschrift für Information Management und Consulting, volume 4. 2010

[2] M. Bettelini, S Rigert, and N. Seifert: Optimum Emergency Management Through Physical Simulation - Findings from the EMILI Research Project, Proceedings of the World Tunnel Congress. Geneva. 2013

[3] S. Brodt, S. Hausmann, and F. Bry: Reactive Rules for Emergency Management. EMILI Deliverable D4.2, University of Munich. 2010

[4] S. Hausmann, S. Brodt, and F. Bry: Dura: Concepts and Examples. EMILI Deliverable D4.3, University of Munich. 2011

[5] S. Brodt, S. Hausmann, and F. Bry: Refinement of the implementation. EMILI Deliverable D4.7. 2012

[6] S. Brodt and F. Bry: Analysing Temporal Relations – Beyond Windows, Frames and Predicates. Submitted for publication. 2013

[7] MonetDB, www.monetdb.org

[8] A. Braun, P. Kroner, Y. Leontyeva, and D. Siller: Event Processing in Sensor Networks: A Solution for Integrated Emergency Management, Poster at the 6th ACM International Conference on Distributed Event-Based Systems (DEBS). Berlin. July 2012

[9] V. Janev, V. Mijović, N. Tomašević, L. Kraus, S. Vraneš. Dynamic Workflows For Airport Emergency Management Training, In: Proceddings of the 2nd International Workshop on Information Systems for Situation Awareness and Situation Management (Workshop ISSASiM '12) at the 23rd International Conference on Database and Expert Systems Applications, IEEE Press. 2012

[10] S. Idreos, F. Groffen, N. Nes, S. Manegold, K. Sjoerd Mullender, M. L. Kersten: MonetDB: Two Decades of Research in Column-oriented Database Architectures, IEEE Data Engeenering Bulletin, volume 35, number 1, pages 40-45. 2012

[11] E. Liarou, S. Idreos, S. Manegold, M. L. Kersten: MonetDB/DataCell: Online Analytics in a Streaming Column-Store, PVLDB, volume 5, number 12, pages 1910-1913. 2012

[12] V. Mijović: Application of ECA rules for Emergency Management at Airports, Proceedings of the 2nd International Conference on Information Society Technology (ICIST 2012), Kopaonik, Serbia, February 29 – March 03, pages 638-642. 2012

[13] L. Kraus, V. Janes, S. Vraneš: Different Approaches to ICT-enhanced Emergency Management, Proceedings of the 56th Conference for Electronics, Telecommunications, Computers, Automation, and Nuclear Engineering, Zlatibor, Serbia, June 11 – 14. 2012

[14] S. Hausmann and F. Bry: Towards Complex Actions in Complex Event Processing, To appear in Proceedings of the 7th International Conference on Distributed Event-Based Systems, ACM, 2013

KURZ UND BÜNDIG

Große Infrastruktureinrichtungen werden technologisch immer komplexer und sind immer stärker voneinander abhängig. Als Folge davon werden komplexe Infrastruktureinrichtungen anfälliger für Notfälle. Das verantwortliche Personal muss in der schwierigen Aufgabe unterstützt werden, Notfälle zu erkennen, richtig zu bewerten und passende Gegenmaßnahmen auszuwählen. Daher empfiehlt es sich, ein System zu wählen, das kontinuierlich die verfügbaren Sensordaten auf Notfälle hin analysiert, Vorhersagen über die zukünftige Entwicklung eines Notfalls macht und sinnvolle Reaktionen auf den Notfall vorschlägt.

Stichworte: Große Infrastruktureinrichtungen, Dynamisches Notfallmanagement, Physikalische Simulationen, Complec Event Processing, Training

SERVICE

About the Authors

Steffen Hausmann
Dipl.-Inform.

Steffen Hausmann received his diploma degree with distinction in 2010 from the University of Munich. Currently he is a teaching and research assistant in the group lead by François Bry at the University of Munich. His research interests are related to complex event processing and reactive systems. He investigates new approaches towards high-level event processing languages unifying declarative event queries with expressive composite reactions.

Simon Brodt

Dipl.-Inform.

Simon Brodt is a teaching and research assistant in the group lead by François Bry at the University of Munich. His research interests are the efficient evaluation of expressive complex event queries, the efficient storage and querying of graph data and the evaluation of logic programs. His current focus is the development of an efficient evaluation engine for high-level reactive event processing languages based on in-memory, column-store databases.

Prof. Dr. François Bry

François Bry is a professor at the Institute for Informatics of the Ludwig-Maximilian University of Munich, Germany, heading the research group for programming and modelling languages. He currently investigates methods and applications related to querying answering, search, complex event processing, and human computation. François Bry regularly contributes to scientific conferences and journals as an author, reviewer, and program committee member. Before joining the University of Munich in 1994, he worked in industry in France and Germany, in particular with the research centre ECRC. François Bry devotes his free time to family, travels, and reading literature and history.

Dr. Marco Bettelini

Dr. Marco Bettelini obtained his Master in Mechanical Engineering from ETH Zurich in 1984 and his PhD from the same institution in 1990. After a post-doctoral year at Brown University, he joined ABB Corporate Research. His main fields of activities were applied aerodynamics, combustion and safety. Since 1998, he operates in the fields of infrastructural safety and tunnel ventilation and held positions as Chief Engineer in several leading engineering companies. He is currently head of ventilation and safety with Amberg Engineering Ltd, in Switzerland.

KONTAKT

steffen.hausmann@ifi.lmu.de
simon.brodt@ifi.lmu.de
bry@lmu.de
mbettelini@amberg.ch

Ludwig-Maximilian University of Munich
Institute for Informatics
Oettingenstraße 67
80538 München, Germany
Phone: +49-(0)89-2180-9771
Fax: +49-(0)89-2180-9311

http://www.pms.ifi.lmu.de/mitarbeiter/derzeitige/steffen-hausmann
http://www.pms.ifi.lmu.de/mitarbeiter/derzeitige/simon-brodt/
http://www.pms.ifi.lmu.de/mitarbeiter/derzeitige/francois-bry/
http://www.amberg.ch