A Methodology for Designing Events and Patterns in Fast Data Processing

Dominik Riemer, Nenad Stojanovic, and Ljiljana Stojanovic

FZI Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany
{riemer,nstojano,stojanovic}@fzi.de

Abstract. Complex Event Processing handles processing of a large number of heterogeneous events and pattern detection over multiple event streams in real-time. Situations of interests are modeled using event patterns which describe a specific situation in an event processing language. In order to leverage the usage of event processing in everyday situations, a clear methodology for the identification and definition of events and event patterns is needed. In this paper, we propose an end-to-end methodology for designing event processing systems. This methodology integrates domain knowledge modeled during the setup phase of event processing with a high-level event pattern language which allows users to create specific business-related patterns. In addition, our methodology regards the circumstance that some patterns might have to be defined by technical experts and therefore introduces an actor model. Our approach is validated based on a real use case of a supplier of convenience stores.

1 Introduction

The emergent amount of data created by a large variety of sources like enterprise information systems, embedded sensors or social media leads to a strong demand for efficient processing of such data in order to get meaningful insights into objects of interests [1]. Two main trends can be currently observed, where the main difference is in the purpose of data analytics. Approaches which are optimized on the processing of large amounts of data, commonly described under the term Big Data Analytics. Even though common characteristics of Big Data are usually described as velocity, variety and volume, the analysis of continuous streams of data in order to process time-critical information is not entirely in the scope of Big Data Analytics [2].

Nevertheless, fast processing of data with a strong real-time nature, with the goal of finding relevant situations is a challenge of increasing importance. This so-called Fast Data Analytics (FDA) aims to identify actionable, time-critical situations, which can often only be detected by a combined observation of different sources. Taking a recent example, Google has just introduced a product called Google Now[1], which combines data of different sources from mobile phones (like calendar information,

[1] www.google.com/landing/now/

C. Salinesi, M.C. Norrie, and O. Pastor (Eds.): CAiSE 2013, LNCS 7908, pp. 133–148, 2013.

mail inbox or the GPS sensor) and external information (like traffic data) in order to proactively notify its users about relevant situations (e.g., Google Now automatically extracts your flight plan from your Google Mail Inbox and informs you when it is time to leave the office in order to catch the train to the airport, based on the current traffic situation and train delays).

A technology enabling Fast Data Analytics is Complex Event Processing (CEP) [3], as it provides powerful methods to aggregate, consolidate and detect patterns over continuous, heterogeneous event streams. CEP follows a paradigm as shown in Figure 1, e.g., multiple event sources produce a stream of events, which is consumed by an event processing engine. Users are able to deploy event patterns in the engine that are matched against incoming events. Once a pattern has been fulfilled, a complex event is created and forwarded to a subscribed event consumer.

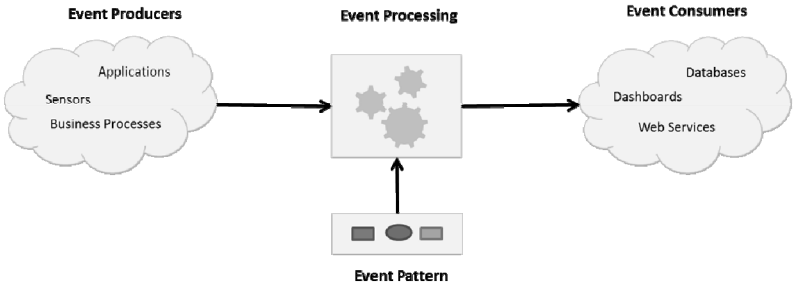

Fig. 1. Basic CEP Architecture

During the last years, the main research efforts in CEP were focused on improving processing performance, adding reasoning capabilities to specialized processing engines [4], or dealing with the uncertain nature of data produced by event sources [5]. Less effort has been put into the management of event processing-based architectures. CEP management basically consists of two parts: The design of the event architecture, meaning the identification of relevant event sources, the definition of specific events that are needed in order to support a specific notification need, and the definition of the data an event should include. Another important part is the management of event patterns. Event patterns are considered the most important asset in event processing [6,7], as they reflect complex situations. Usually, patterns are defined using an event processing language. Many different languages have been proposed, e.g., SQL-like languages, rule-based languages and logic-based rules. Most existing event pattern languages today suffer from a very technical design which makes them highly expressive but restricts the pattern development process to a technically oriented audience. In order to create and facilitate real-time awareness in modern enterprises, there is a need for fast and simple generation and deployment of event patterns which is not only restricted to technical users. Moreover, similar to the problem of identifying useful event sources, only few methodologies have been proposed which help to define relevant patterns based on business goals, which are aligned to the overall strategy [8]. While users often have a goal in mind which they want to implement as

an event pattern, it is not clear how to resolve such a goal to a technical machine-readable pattern. On the other hand, application scenarios for big/fast data require more dynamics in the definition of the relevant situations for the user, leading to an emerging need for better management of event processing.

In this paper, we argue that the acceptance and usage of event processing in everyday situations depends on a clear, end-to-end methodology which takes into account the whole lifecycle of an event processing-based application and which uses domain knowledge in order to provide non-technical users with a simplified way for the definition of situations. Our approach consists of the following contributions: 1) A methodology for identifying event sources and their corresponding events, 2) An ontology-based model used as background knowledge which combines technical-oriented terms with domain-specific knowledge, 3) A methodology which helps breaking down high-level strategic goals into fine-grained, measurable business goals which can be translated into event patterns in a semi-automatic way and 4) A way to model patterns in a more user-friendly way using the predefined background knowledge. Although we present the complete methodology, in this paper our main focus is on the introduction of the general methodology and conceptual modeling of background knowledge during the initial event architecture design.

The rest of this paper is structured as follows: In section 2, we further stress the need for our proposed approach based on a scenario from an electronics retail store. Chapter 3 discussed related work on event processing and event pattern management. Afterwards, we introduce the conceptual model in detail, followed by a validation in section 5, which shows the application of the proposed model in a real-world scenario. Finally, we outline future work in section 6.

2 Motivating Scenario

Our scenario is based in the area of a chain of electronic retail stores. In this domain, a company holds a number of stores that are selling electronic products. Stores operate independently, while procurement is performed by the company's headquarter. Products are bought by a number of suppliers. In addition, the headquarter subcontracts a logistics service provider (LSP) which maintains a central warehouse. Products ordered from a supplier by the headquarter are picked up by the LSP in large trucks and brought to the warehouse. From the warehouse, products are shipped to the retail stores using smaller delivery vehicles. As the retailers do not provide large storing capabilities, short-term subsequent deliveries of products is of high importance.

We consider the presence of several data sources which provide real-time information:

- Vehicles are equipped with GPS sensors which provide position data and the current vehicle speed
- Some customers have the company's mobile app installed on their mobile phone. The app enables customers to preorder products before they are available on the market. In addition, users can enable a function that shares their current position with the company, so that they can receive offers based on their current location.

- Vehicle drivers have mobile devices with integrated barcode reading software, so that products which are delivered to a customer are scanned once they are unloaded.
- The enterprise information systems (such as Customer Relationship Management, Fleet Management and Order Management software) basically support the generation of events after some state of the software (e.g., a route has been created) has changed.
- Recently the company started to use a social media monitoring tool which tracks opinions about the company in social networks and blogs in real-time.

The company aims to make use of such data sources in order to increase the situational awareness and to enable faster reaction on potential threats or opportunities which occur during the execution of business processes. In addition, as the transportation and storage costs as well as lost profits due to sold out products should be minimized, the company wants to track the performance of some Key Performance Indicators (KPI) in real time.

Some example KPIs which might be of interest include:

- The current percentage of delivery vehicles which arrive on time
- Currently trending products
- The importance of products in a specific region, measured by the number of mentions in social networks
- The average revenue per customer category

Potential threats might be:

- Unsatisfied customers. Customers might mention their unsatisfaction through different channels, e.g. in social networks or the customer service center. One situation which helps minimizing this threat would be to contact a customer who publishes a negative mention about the company in a social network proactively by offering support.
- Traffic jams. Traffic jams can be risky for LSPs (because they violate the previously planned daily route), as well as retail stores (because of lost revenue due to sold out products). A possible solution would be early identification of traffic jams on streets which affect an existing route plan through external sources which triggers the rescheduling of the touring plan.

On the other hand, business opportunities, meaning the detection of a situation which might lead to increased revenue, less costs or customer satisfaction can suddenly appear, such as:

- An important customer who has preordered a specific product is notified immediately after the product arrives at the store and the customer is within a fixed radius around the store.
- A truck that is currently on the way to a supplier, might be spontaneously sent to another supplier after a new order was placed by the headquarter.

The examples described above can be represented by corresponding events patterns. Obviously, due to a large number of various event sources, a potentially large number of produced events per event source and the demand for fast definition, deployment and evolution of event patterns, there is a need for a more structured way to define patterns. As many possible patterns presented in this scenario can be clearly assigned to a specific business goal, goal definition approaches can be used to create a hierarchical process which breaks down business strategy to event patterns. Without a methodology, the definition of patterns is error-prone and ad-hoc.

In addition, the expressivity of patterns must be supported by the event architecture in order to facilitate an end-to-end architecture that allows business-level users access to pattern definition in everyday business situations.
The design of the event architecture, meaning the event identification and their properties, heavily depends on the existing application and sensor landscape.

Altogether, this scenario stresses the following challenges in real-time information systems engineering:

- How can sources which produce real-time information be identified, and which events have to be produced by the underlying systems?
- Which methodologies support the formulation of strategy-oriented business goals and how can they broken down to technically-oriented event patterns?
- Which actors are involved in the setup and execution of real-time information systems and how are they related?
- How can the usage of domain knowledge support the goal-oriented pattern definition process?

In the following section, we discuss related work addressing these issues. Afterwards, section 4 presents a methodology for domain-aware design of events and patterns.

3 Related Work

This section discusses related work in the areas of event processing, event pattern management and Business-IT-Alignment.

3.1 Event Processing Networks

An event processing network is described as a collection of event processing agents, event producers and event consumers [9]. An EPN can be described as a graph consisting of event producers, consumers and processing agents as nodes and a set of directed edges representing event channels. An event processing agent processes consumed events and processes the output needed by the event consumer. In this sense, processing can be simple filter operation which outputs a subset of the input events based on a filter condition, but an EPA might also perform a more complex operation like aggregating a sequence of events having a different type over a time window. Etzion defines several different EPAs as described in [10]. EPAs are

hierarchically categorized, starting with the types Filter, Transformation and Pattern Detect. Transformation agents require a single or multiple events as an input and perform an operation on the event payload. For example, an *Enrich* EPA creates a new event which contains additional information compared to the input event. A complete description of EPA types can be found in [9]. In our approach, EPAs are used for the definition of several Business Event Processing Agents (BEPA) that are provided to non-technical users for a more simplified definition of event patterns.

3.2 Event Pattern Management

There are few approaches which take into account the problem of identifying and managing event patterns. Vidakovic et. al. propose a business-oriented methodology for the development of CEP systems [8]. This methodology uses business goals (by applying the Business Motivation Model (BMM) and transfers such goals to a logical, hierarchical perspective where users can define KPIs, situations and reactions. The pattern definition process is completely manual and relies on technical knowledge of a specific event pattern language. An approach for user-oriented rule management has been proposed by Obweger et. al [7, 11]. They distinguish between infrastructural rules, which are defined by technical experts and are provided to business users, which finally use so-called sense-and-respond rules to create meaningful business patterns. This approach improves accessibility of event pattern definition for non-technical users, but does not describe how rules are identified using a structured way. Finally, Sen et. al. present a methodology for the management of complex event patterns [6]. By taking the complete life cycle (generation, refinement, usage and evolution) of event patterns into account, they propose an RDFS-based event pattern representation. In comparison to our approach, this methodology can be seen as a part of the complete process of business-oriented pattern definition, as this approach excludes the problem of how relevant patterns can be identified.

3.3 Business-IT-Alignment

Business-IT-alignment refers to applying Information Technology (IT) in an appropriate and timely way, in harmony with business strategies, goals and needs [12]. In order to improve the alignment between business goals and underlying IT systems, several conceptual models have been proposed: The Business Motivation Model (BMM) is an enterprise architecture model which describes how business requirements can be captured and how operative directives can be modeled. The model consists of the main elements Ends, Means, Directives, Influencers and Assessment. These elements can be used in our approach to break down high-level strategies to fine-grained business goals as the basis for an event pattern definition. Based on the BMM, Veres et. al. propose an ontology-based implementation, which enables validation of defined requirements. The ontology-based BMM is integrated into the B-SCP framework presented by Bleistein [13]. The objective of B-SCP enables alignment of IT requirements with the overall business strategy by introducing a requirements engineering analytical framework. In our

presented methodology, B-SCP will be the basis for the structured definition of event patterns based on strategy-oriented business goals.

4 Conceptual Model

In this chapter, we present a methodology for the identification of relevant events and a structured approach for the definition of useful patterns which enable real-time monitoring and adaption of running processes in accordance with identified threats and opportunities which affect business goals.

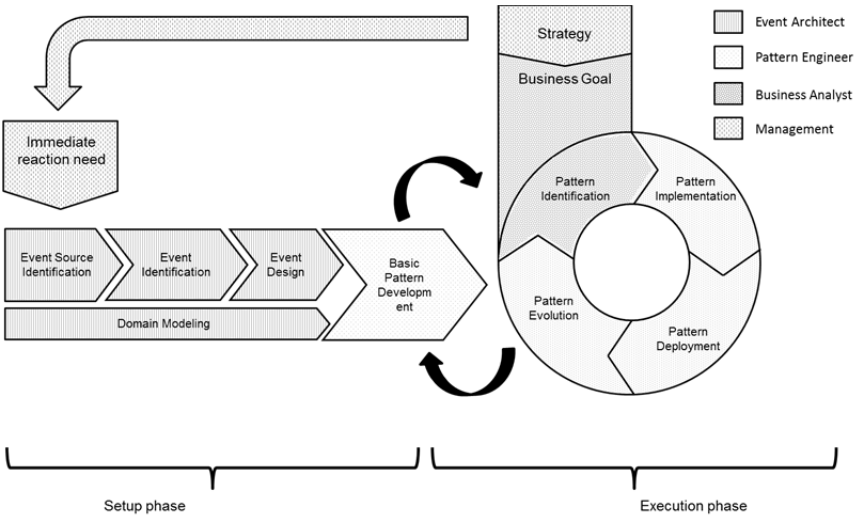

Fig. 2. Overview of the methodology

4.1 Overview of the Methodology

Figure 2 shows the overall methodology of our approach. Basically, the initial decision to make use of event processing is usually triggered by an *immediate reaction need* meaning an organization has decided that immediate reaction on situations detected in real-time is required in order to achieve competitive advantages or to avoid threats. In general, this process can be divided into the Setup Phase, where the event infrastructure is built and the Execution Phase, where relevant situations which need to be captured are identified based on the existing event landscape. Usually, this process starts with the identification of relevant event sources. Afterwards, for each event source, situations are identified which should trigger events based on business processes. This process is followed by the design of all events, which includes the design of event properties, their types as well as the identification of metadata information used as header attributes (like timestamps and identifiers) in events.

In parallel to these processes, a domain model is constructed reflecting the application domain. In our motivating scenario, this would include modeling different actors, their properties and the relation between the actors. At the end of the setup phase, basic event patterns are built which construct the *Event Pattern Pool*, a repository of patterns that can be used by business analysts in order to create business-oriented patterns. This pattern pool can be extended during the execution phase, in the case that new operations should be supported.

4.2 Actor model

In this section, we briefly describe several roles which are used in the setup phase as well as during the execution phase. Later in this chapter, we will further elaborate on the specific tasks an actor is involved in.

Management. The role of managers is the definition of a business strategy and high-level goals that the organization aims to achieve. Some of the objectives defined by managers might be defined by high-level metrics (e.g., increase the revenue by 20 percent).

Business Analyst. In our methodology, business analysts are responsible for the translation of strategies into specific tactics which support the business strategy. They define fine-grained business goals, opportunities which should be detected at runtime as well as threats. For each business goal, KPIs are defined which are able to track the successful achievement of the underlying goal.

Pattern Engineer. This role is the counterpart of the Business Analyst from a technical point of view. Pattern engineers are responsible for the technical implementation of a goal-oriented event pattern. In some cases where a corresponding pattern is very complex, this might be a purely technical process, whereas in most cases pattern definition can be done using graphical tool-based support. Pattern engineers are also responsible for monitoring the pattern execution and for the adaption of event pattern based on changed business requirements.

Event Architect. Event architects build the event infrastructure during the setup-phase. They identify event sources based on the existing application and sensor land-scape, define events and their properties. In addition, event architects are responsible for the definition of background knowledge which defines the domain of interest and leverages business-oriented definition of event patterns.

4.3 Setup Phase

The main outcome after completion of the setup phase is an ontology-based model which is comprised of event sources, events and event properties. This technical model is connected to a business domain model, which is the basis for a higher-level way to express event patterns. In this section, we will describe the tasks that should be

performed during the setup phase in more detail. Afterwards, we present an integrated model. Based on that model, we describe the role and advantages of an ontology-based model.

Event Source Identification. The setup phase starts with the identification of event sources. We assume each source has different characteristics, e.g., different data types which are produced by such sources. Depending on the specific source, different operations during event processing are possible or not. As an example, the calculation of a point approaching another point is only possible if there is at least one point provided by a mobile (moving) sensor data source. Figure 3 presents an exemplary taxonomy of data sources. On the first level, we distinguish between Information Systems, Hardware Sensors and Information Feeds.

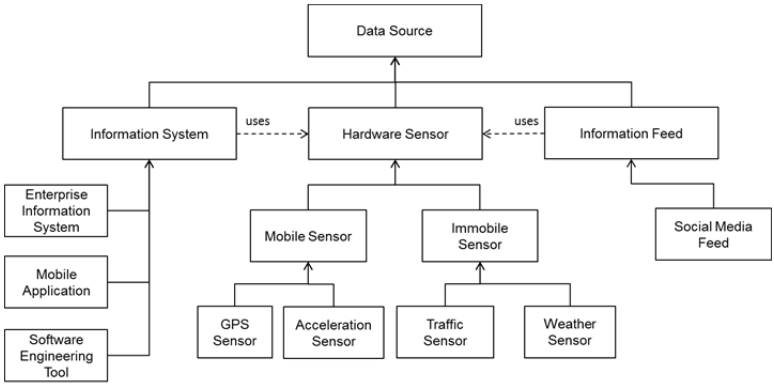

Fig. 3. Model of event sources

Existing Information Systems are identified by using methodologies provided by an enterprise architecture framework like the Zachman Framework [14]. Information systems can be classified based on their purpose as shown in Figure 4. Using the example provided in the motivating scenario, there are several enterprise information systems like Customer Relationship Management, Order Management, Fleet Management and Warehouse Management. Information systems can support the activation of underlying hardware sensors, i.e., a mobile application might be able to send information gathered by the mobile phone's GPS sensor.

Event Identification and Event Design. Upon the model of event sources, a detailed model of corresponding events is generated. By analyzing existing and documented business processes, events can be identified. Especially in the case the event is created by an information system, it is a trigger or the result of a business process. As described in Figure 4, our event model is consisted of events, event properties, property types, actions and event sources as described above.

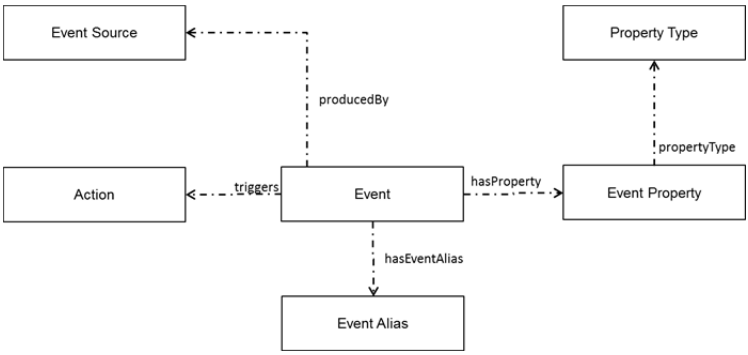

Fig. 4. Event Model

Event properties are of a specific type like a textual representation, a spatial geometry (e.g., a point) or a numerical value which can perform arithmetic operations. The definition of property types is essential in our approach, as the event processing agents created during the setup phase which support a specific event are calculated based on the event source and its event type. An event might trigger an action, e.g. trigger the sending of a notification, the invocation of a web service or the creation of a visualization. In our model, we defined the concept of event aliases, as the occurrence of an event might have a different meaning from the perspective of different actors. As an example, if the order management system creates an event in the case a new order is received, and there is no other event which describes the actual placement of an order, this event would be named "orderReceived" from the perspective of the warehouse and "orderPlaced" from the perspective of the customer despite being the same event.

Domain Modeling. The domain object model describes the domain of interest. It is closely related to real-world objects which are essential from a business user's point of view. Domain objects are assigned to a specific event source which represents general properties of the domain object. In addition, a domain object can have different static properties, which do not change over time. This might be the category of a product or the revenue of a customer. Figure 5 shows the conceptual domain object model. A more detailed example will be provided in the validation.

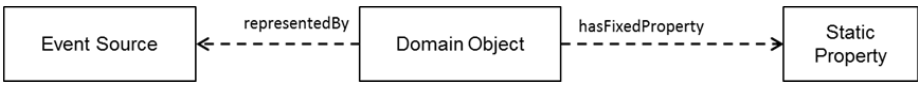

Fig. 5. Domain Object Model

Combined Business/Event/Pattern Model. Once the domain model has been developed and the event identification and design processes have been finished, gathered data is integrated to a combined event/pattern model. This model reflects technical details such as event processing agent types, supported event operators and

aggregation windows as well as the modeled business logic. The combined model is described in Figure 6.

The connection between real-world domain objects and technical-level events allows the development of business event processing agents (BEPA) which provide a self-description of their functionality. BEPAs are connected to one or more events as an event input, a specific EPA type and one or more events which are produced as an output. A formal description of the BEPA structure is given in the following section. BEPAs are modeled hierarchically, meaning the input of a BEPA is either an event or another existing BEPA. In our proposed methodology, BEPAs are modeled by pattern engineers as part of the setup phase. The goal of BEPA design is to provide an abstraction of event processing agents to non-technical users in a way which allows the definition of event patterns from a business perspective. Business users modeling new patterns by using a graphical tool are able to use BEPAs from a business event pattern pool. This pool might be extended on demand by pattern engineers, in the case a new functionality is demanded by business users.

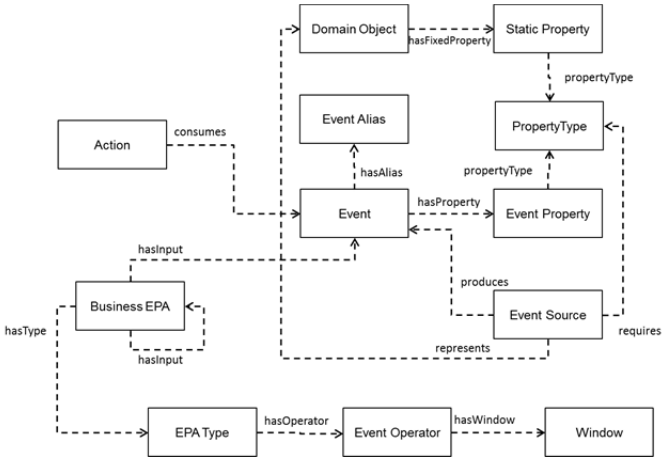

Fig. 6. Combined Business/Event/Pattern model

Modeling of Event Processing Agents. In this section, we specify the structure of a BEPA and describe the implementation of BEPAs in detail how BEPAs are implemented.

We define the following sets: $ES = \{E_1, \dots, E_n\}$ is a set of events, $EPL = \{ep_1, \dots, ep_n\}$ a set of event properties, $PTL = \{pt_1, \dots, pt_n\}$ a set of property types, $S = \{s_1, \dots, s_n\}$ a set of event sources, $D = \{d_1, \dots, d_n\}$ a set of domain objects, $A = \{a_1, \dots, a_n\}$ a set of possible actions that can be triggered, $SP = \{sp_1, \dots, sp_n\}$ a set of static properties, $T = \{t_1, \dots, t_n\}$ a set of event processing agents and $O = \{o_1, \dots, o_n\}$ a set of event operators. B is denoted a set of existing BEPAs.

A Business Event Processing Agent is a tuple $BEPA = (EI, EO, t)$ with $EI \in ES \cap B$ as a set of input events, $EO \in ES$ the set of output events and $t \in T$ as the EPA type. In addition, we assume $\forall ep \in EPL: \exists_1 s \in S$ and $\forall e \in E: \exists ep \in EPL$. Furthermore, an event is a tuple $E = (EP, ET, A, es)$: $es \in S, EP \subseteq EPL, ET \subseteq PTL$. An event source produces an event which has at least the required properties, so that $\forall s \in S: \exists p \in EP$. In this sense, we define a function $valid(E) \rightarrow \{0,1\}$ with:

$$valid(E) = \begin{cases} 1, & EP \cap es \neq \emptyset \\ 0, & else \end{cases}$$

Each BEPA provides a function $f(EI) \rightarrow EO$ which calculates an event output based on the input. An EPA type accepts n events as an input, so that $|EI| = n$. A BEPA performs an operation on an event set if the event which should be attached to the BEPA requires at least one specific property type,$rp \in PT$. An input event is accepted, if its payload contains the property type required by the EPA. In this sense, the input set is restricted to: $\exists ei \in EI: \{rp \cap ei\} \neq \emptyset$. One advantage of our approach is that we do not only take the property type into account, a BEPA only accepts events if they are provided by a specific source type. This means, the function $validBPA(EI) \rightarrow \{0,1\}$ is only valid if:

$$validBEPA(EI) = \begin{cases} 1, & EP \cap es \neq \emptyset \\ 0, & else \end{cases}$$

In words, BEPAs are independent from their underlying event source. Events that are produced by a specific source are required to contain specific property types. New designed events are only accepted by the system if they contain the property defined by the underlying source. The same validation applies for BEPAs. A BEPA is self-describing in a way that it defines which input is needed in order to produce the output. This feature enables automatic validation of a created event pattern. As our approach does not only require a specific property format (e.g., a location data type), but also takes the event source into account, we support an abstract way to define BEPAs like this example: Let $proximity(EI, EO)$ be a pattern which detects if a point approaches another point. Furthermore, the following restrictions apply: $|EI| = 2, |EO| = 1, EI = \{E1, E2\}$. The set of event properties is $EPL = \{id, location\}$, the property type list is $EPT = \{string, geometry\}$ and the set of event sources is $S = \{GPS\ Sensor, TrafficSensor\}$. Using these properties, the BEPA can be configured to accept events with the following properties:

$$\exists s: s \in E1 \wedge s \subseteq MobileSensor,$$
$$\exists t: t \in E2 \wedge t \subseteq HardwareSensor,$$

meaning there must be at least one input event which is produced by a mobile sensor. In addition, due to the link between domain objects and event sources, business users are provided with only those operators that are supported by an event produced by a domain object.

5 Validation

The presented approach has been tested in an EU FP7 project. In this project, we performed the setup phase in a way described in section 4 in the case of a SME company who acts as a supplier of convenience stores. The main requirement for enabling real-time awareness in the partner's IT systems was the intransparency of the current business situation. As an example, before the start of the research project, product return requests of customers were stored on a mobile device and processed during the next business day. After the partial implementation of our approach, the company was able to receive such requests immediately and was able to reroute a delivery vehicle automatically in the case several conditions have been met. As already described, our approach is important for the maintenance of an event-based system, i.e. for the generation of new event patterns.

As follows, we present the application of our methodology on the specific use case and provide instance models for each task described in section 4. Afterwards, initial evaluation results as well as important lessons learned are provided.

The overall strategy was to introduce real-time awareness within the company. This objective should be achieved by establishing an event-driven architecture besides the existing application landscape. In more detail, the main goals were a) to enable real-time responsiveness through semi-automatic adjustment of business processes based on real-time insights and b) to provide real-time information about business process execution. The first task consisted of the identification of relevant event sources. By analyzing the existing application landscape, we identified four relevant information systems (customer relationship management, enterprise resource planning, document management and a geographical information system). In addition, a mobile application was identified (that is used by sales personnel in order to submit orders directly). Finally, we added already existing GPS sensors which are installed in all delivery vehicles of the company to our model.

The second task handled with the identification of events. This was done in two steps, a deep analysis of the company's business processes and a workshop with domain experts. At the end of this process, we implemented 22 events in total which were produced by the identified source types. Events were chosen based on their importance for real-time responsiveness. In parallel, we constructed a basic domain model consisting of the entities Warehouse, Customer, Vehicle, Document and Order and their respective static properties (e.g., the location of the warehouse and the location of all customers). In order to allow a definition of patterns from a less technical point of view, we extended this taxonomy with further information gathered through interviews with domain experts. For instance, we modeled vehicles based on their type (delivery vehicles, trucks, vehicles of sales representatives), customer statuses and products based on their category (e.g., cosmetics, food, electronics).

Design and implementation of basic event patterns started after both models had been constructed. At first, the taxonomy of domain objects was mapped with technical events. We developed a method called semantic requests [15] in order to allow the usage of domain objects instead of events at pattern definition time. Semantic

requests allow abstracting specific event properties by using domain knowledge. As an example, a BEPA that calculates the total revenue of gold customers during the last 2 hours is automatically translated into a machine-readable pattern without processing background knowledge about gold customers in the engine. In addition, we implemented the following BEPAs:

- *proximity(Vehicle, Geometry, x)* detects a vehicle which approaches an area around a specified location and was instantiated with *proximity(DeliveryVehicle, Customer, 10)* in order to notify customers that their order is about to be delivered.
- *trending(Product, Geometry, x)* is used to identify products that are currently within a certain radius around a location. This BEPA was applied in order to improve just-in-time delivery of perishable food based on the current demand.
- *distance(DeliveryVehicle, x)* calculates the total driving distance of delivery vehicles within a certain time period.

Note that these patterns are updated automatically after changes in the knowledge base have been applied as the business logic is isolated from the technical pattern representation. A simplified instantiated domain model and corresponding events are illustrated in figure 7. The conceptual model presented in section 4 is extended with a use case-specific taxonomy (shown in rounded rectangles) and an example instantiation of a proximity-BEPA (illustrated in gray rectangles).

The final model was presented to the end users in order to get feedback on the quality of the model. In another workshop, we analyzed the proposed model and the design of events and provide an initial validation of the approach. In order to assess

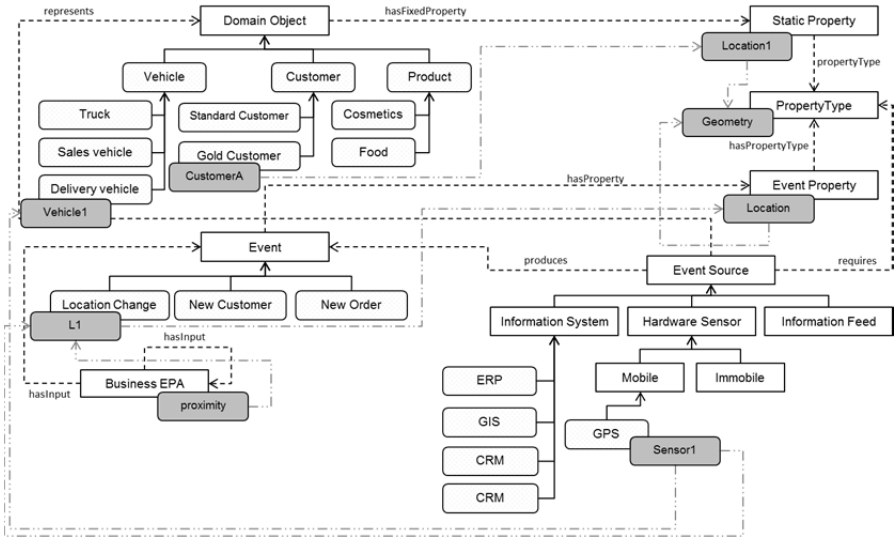

Fig. 7. Excerpt of the instantiated model

the validity of the proposed methodology we have introduced three parameters that can be measured. The goal is to evaluate the suitability of the resulting event model for the selected use case. These metrics are:

1. Completeness: it indicates how complete is the coverage of the event types (i.e. do we have identified all event types that are required for the realization of the use case). Therefore, completeness was measured through the number of additional event *types* that have been added in this discussion. There were three missing types, all were leafs in existing hierarchies. In particular, events produced by mobile applications were missing.
2. Correctness/Soundness: it indicates how relevant the identified event types are (i.e. does every identified event type have a role in at least one complex event pattern). As we focused on the setup phase within this paper, correctness of the model was measured by the number of wrongly defined events. There was only one wrongly defined type which was again a leaf.
3. Complexity: it indicates how difficult it is for end developers to deal with the provided event model (i.e. are all event types presented in an understandable form - this is a subjective category). Complexity of the modeling that was assessed in a rather qualitative way by performing an open discussion. The main conclusion is that the method is quite straightforward (easy to be explained) but its applications requires sophisticated tools.

Currently, we are working on the development of a graphical editor which allows the capabilities of pattern definition in our model. We have planned an additional use case study for validating the modeling toolset after the development of this tool is completed.

6 Conclusions and Future Work

In this paper, we presented a novel methodology for designing events and patterns in fast data processing. Modern organizations need real-time awareness about current business conditions and the various events that occur from multiple and heterogeneous environments. Moreover, the need for flexible processes is big in today's competitive environment as a lost customer, or a missed opportunity to recruit a new customer, may never be recouped. In order to facilitate the broad acceptance and usage of event processing technologies in everyday situations, we identified a need for an event and pattern definition approach which considers requirements of non-technical users. Our future work includes the integration of a business goal definition framework (like the B-SCP model described in section 3) into our methodology. Altogether, the goal is to provide an end-to-end methodology for alignment of business strategy with event processing technologies by bridging the current gap between business-oriented tactics and goals on the one hand and technical-oriented event pattern languages on the other hand. We are further investigating the connection of business event processing agents and semantic web services.

References

1. Bughin, J., Chui, M., Manyika, J.: Clouds, big data and smart assets: Ten tech-enabled business trends to watch. In: McKinseyQuarterly (2010)
2. Zikopoulos, P., Eaton, C.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Osborne Media (2011)
3. Luckham, D.: The power of events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Reading (2002)
4. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Real-Time Complex Event Recognition and Reasoning – A Logic Programming Approach, Applied Artificial Intelligence, vol. 26 (2012), Special Issue on Event Recognition
5. Wasserkrug, S., Gal, A., Etzion, O., Turchin, Y.: Complex event processing over uncertain data. In: Proceedings of the Second International Conference on Distributed Event-Based Systems (DEBS 2008), pp. 253–264. ACM, New York (2008)
6. Sen, S., Stojanovic, N.: GRUVe: A Methodology for Complex Event Pattern Life Cycle Management. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 209–223. Springer, Heidelberg (2010)
7. Obweger, H., Schiefer, J., Suntinger, M., Breier, F., Thullner, R.: Complex Event Processing off the Shelf – Rapid Development of Event-Driven Applications with Solution Templates. In: Proceedings of the 19th Mediterranean Conference on Control and Automation, Corfu, Greece (2011)
8. Vidačković, K., Kellner, I., Donald, J.: Business-oriented development methodology for complex event processing: demonstration of an integrated approach for process monitoring. In: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems (DEBS 2010), pp. 111–112. ACM, New York (2010)
9. Sharon, G., Etzion, O.: Event-processing network model and implementation. IBM Systems Journal 47(2), 321–334 (2008)
10. Etzion, O., Niblett, P.: Event Processing in Action. Manning (2011)
11. Obweger, H., Schiefer, J., Suntinger, M., Kepplinger, P., Rozsnyai, S.: User-oriented rule management for event-based applications. In: Proceedings of the 5th ACM International Conference on Distributed Event-Based System (DEBS 2011), pp. 39–48. ACM, New York (2011)
12. Veres, C, Sampson, J, Cox, K., Bleistein, S., Verner, J.: An ontology based approach for supporting business IT alignment In: Complex Intelligent Systems and Their Applications. Springer Optimization and Its Applications (41). Springer, pp. 21-42.
13. Bleistein, S.: B-SCP an integrated approach for validating alignment of organizational IT requirements with competitive business strategy. PhD Thesis (2006)
14. Sowa, J.F., Zachman, J.: A Extending and formalizing the framework for information systems architecture. IBM Systems Journal 31(3), 590–616 (1992)
15. Riemer, D., Stojanovic, L., Stojanovic, N.: Using Complex Event Processing for Modeling Semantic Requests in Real-Time Social Media Monitoring. In: Sixth International AAAI Conference on Weblogs and Social Media (May 2012)