User-centric Complex Event Modeling and Implementation Based on Ubiquitous Data Service

Feng Gao
Unit of Service Oriented Architecture
Digital Enterprise Research Institute, DERI
Galway, Ireland
email: feng.gao@deri.org

Sami Bhiri
Unit of Service Oriented Architecture
Digital Enterprise Research Institute, DERI
Galway, Ireland
email: sami.bhiri@deri.org

***Abstract*—Current complex event processing systems are often implemented as standalone engines that produce business events and feed process execution environments. Event patterns are defined with rule-based languages. Logical programming and/or stream processing techniques are used to detect matchings for the patterns. However, tremendous technical efforts are required both for the pattern definition and implementation. In this paper, we present a novel framework that provides a user-centric way to define complex event patterns and implement the patterns automatically. We allow the business users to describe their complex events with graphical notations and transform the graphical pattern into a stream query, then, we evaluate the query over primitive sensor data streams to obtain results as complex events.**

Keywords-user-centric; complex event processing; stream reasoning; web service; BEMN.

I. MOTIVATION

Business Process Management (BPM) provides concepts, methodologies and tools to design, implement, execute and reengineer business processes. Today business are demanding more flexibility from BPM to adapt to the fast changing business environment in-time. To this end, the concept of ”BPM 2.0” has been brought up that aims to bring more flexibility into BPM. Among the methodologies used in BPM 2.0, an important idea is to provide automation support for the implementation of business processes [1]. This allows business analysts to test and run the processes they design with minimized technical effort. Current research focus on combining BPM with Service Oriented Architecture to provide automation support for process implementation. In particular, service discovery will find direct matching services for process tasks and simple events, service composition will try to create services for process tasks when there's no direct matching.

However, Complex Event Processing (CEP), an indispensable technique for business process systems [2], has not yet get enough automation support. Most current CEP tasks are delegated to CEP engines. These engines are usually equipped with rule-based languages and engines to define and analyze complex event patterns, and require some programming skills to encapsulate the corresponding event data to communicate with event processing engines. Unfortunately neither rule languages nor programming APIs are friendly enough for business users. As such, companies need significant technical efforts to implement a business process that requires CEP.

On the other hand, development in sensor networks is gaining increasing interests from enterprises. Many efforts have been made to integrate sensor functionalities with enterprise systems to manage business processes more dynamically. A natural use of sensors is to monitor the state changes of the real-world and notify event driven processes to take actions. However business events are complex and sensor events are primitive, thus CEP techniques are crucial for the deriving business events from sensor reading events.

To give an example of complex events derived from sensor readings, let us consider the following scenario: in a supermarket, sensors are deployed on shelves to monitor the numbers of remaining products, when products left on the shelf are insufficient, a sensor will report an out-of-stock event and trigger a replenishment process. If 10 out-of-stock events are captured for the same product during the past week, or a direct request from the manager asks for increasing the storage for the product is received, a complex event will raise to notify the need of increasing the amount of the product in the next purchase order. In this paper, we will demonstrate our work that provides a user-centric and automatic way to define and implement complex event patterns.

The remainder of the paper is organized as follows. Section 2 elaborates our research problems in detail. Section 3 presents the related work. Section 4 gives an overview of the proposed system. Section 5 discusses the graphical notations for complex event definitions. Section 6 briefly describes the algorithm to transform the event patterns into stream queries before we conclude in Section 7.

II. RESEARCH PROBLEM DESCRIPTION

Our research intends to provide means to define and implement complex event patterns in a user centric way. Our goal can be further decomposed into the following.

 ISBN: 978-1-61208-236-3

A. User-centric Complex Event Pattern Definition

We need an intuitive, friendly language for business users to create the complex events they need. Graphical notation is our first choice. Flow-based graphical structure can be used to model the control and data dependency between primitive events. The language should be expressive enough for various business event patterns. Meanwhile, to ensure that the event patterns are operating on the correct event sources, a user-oriented mechanism is required to facilitate primitive event service discovery.

B. Automated Implementation for Complex Event Patterns

Implementing complex event pattern requires evaluating event rules represented by event patterns over streaming data. A Data Stream Management System (DSMS) is usually implemented as a component in a CEP system to process streaming data. Comparing to conventional DSMS employed by current CEP engines that can only process syntactical data, an emerging research area of stream reasoning aims to process continuous semantic data. Our key idea of automated implementation is to transform complex event patterns into declarative stream reasoning queries, so that the complex event patterns defined by the process modelers/business users can be evaluated by the stream reasoning engine and are practically made executable with minimal technical efforts. There exist some stream reasoning languages and engines, our approach can reuse some of them with proper modifications/extensions to evaluate complex event patterns.

III. Related Work

Complex Event Processing (CEP) is a technique to detect complex events in (near) real-time. A complex event represents a set of correlated events called its member events. A complex event pattern describes the temporal relationships, data specifications and other conditions required to derive the complex event from its member events. Most CEP systems describe event patterns in SQL-like languages. Wang et al. [3] and Dunkel et al. [4] use such languages to specify complex events and feed the process engine with derived events. However these languages only offer textual representations. BEMN proposed by Decker et al. [5] is the first work that attempts to provide a graphical and executable event pattern language and is an inspiration to our work, it is able to describe various business event patterns. Still, the BEMN language have some limitations regarding to the difficulty of implementation, scalability and expressiveness, which we will discuss in details later. All the approaches described above can only process the dynamic data in the stream and do not go beyond syntactical processing.

Stream Reasoning is an emerging research area that tries to enable reasoning on continuous data and support processing for both dynamic data and static background knowledge. Anicic et al. [6], proposes a prolog-based framework is to transform continuous triples as logic facts and stream query as rules, so that the processing of complex events can make use of both dynamic event data and static background knowledge. Le-Phuoc et al. [7] use a "white box" approach and support native stream reasoning operators, they also provide means to optimize the query plan so that the evaluation of stream query can be more efficient. We will transform our user-centric complex event definition into the executable stream query language defined in [7] (with some extensions) to enable automated implementation of complex event patterns.

Semantic CEP is discussed in several works. Moser et al. [8]elaborate the benefits to extend syntactical event correlation to semantic event correlations. In the paper they define 3 kinds of semantic correlations on event attributes: equivalence, inheritance and relation-based. Li et al. [9] use an ontology to describe event rules as well as context-aware devices (sensors), and an event hierarchy is used to model the causal relationship between different levels of events. In the framework propsed by Taylor et al. [10], users can select and correlate sensors based on the semantic sensor description, then, a semantic middleware will translate the users' requirements into the internal language used by the CEP engine (e.g. EPL) and program the sensors to prepare the streams. Semantic query and reasoning in this approach will not affect the run-time processing so that it can guarantee high throughput of the CEP engine. However, it does not give a formalization of event patterns and have limited expressiveness in terms of AND and OR patterns and aggregations. Moreover, it relies on programmable sensors. Hasan et al. [11] propose a dynamic enrichment of the stream sensor data so that the information sensed can be correlated to background knowledge based on the interested situation at runtime. This approach is different to ours, where we load the static knowledge before runtime.

IV. System Architecture

In this section, we will introduce the architecture of our proposed system by presenting the overview of the framework and functional descriptions of its components.

A. Overview

An architectural design of our system is depicted in Figure 1. The system aims to realize user-centric and automated complex event implementation for business users in a service-oriented environment based on publish/subscribe messaging paradigm and semantic web technology.

B. Functionalities of Components

Due to the limited space we will only introduce the crucial components in the system framework in the followings.

- **Process Modeling Environment** provides utilities for business process designers to create, update, retrieve and delete process models. The process modeling tool will support graphical notations for describing complex

 ISBN: 978-1-61208-236-3

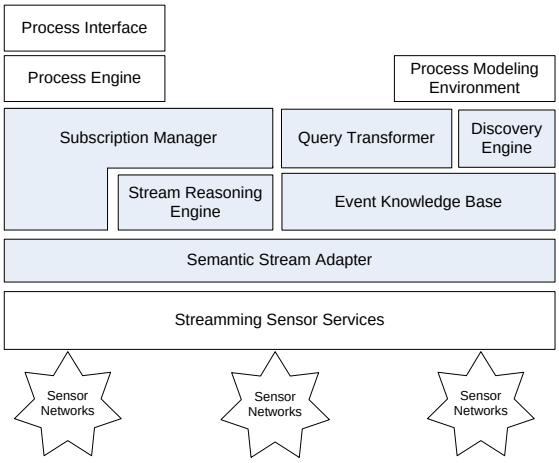

Figure 1. System Architectural Overview

event patterns. The graphical notations can be made compatible to the standardardized Business Process Modeling Notation (BPMN).

- **Event Service Discovery Engine** helps the process designer to select primitive sensor event services based on semantic service descriptions.
- **Query Transformer** will take the complex event descriptions as inputs to create stream queries over the sensor data streams. We will briefly describe the strategy of the algorithms used in the transformer later.
- **Event Knowledge Base** stores the top-level event ontology, domain-specific ontology and datasets (including sensor service descriptions).
- **Semantic Stream Adapter** is the direct consumer of the sensor services. It will subscribe to sensor services and receive messages from them. Then the adapter will convert these messages into RDF triples using mapping schemas defined in service descriptions and construct semantic streams for the stream reasoner.
- **Stream Reasoner** is the query engine for the RDF streams. It will evaluate a stream query over a specified window on semantic streams. Results of the stream query will be consumed by the process engine as business events.
- **Streaming Sensor Services** are the set of web services that produce notification messages used to construct streams. Currently we adopt the WS-Notification protocols to wrap functionalities of sensors into web services and produce primitive events.

V. Graphical Notations for Complex Events

BEMN [5] intends to provide a graphical representation for the event composition languages beyond conventional textual language. BEMN diagram can be integrated into BPMN process models seamlessly to facilitate complex event description in business processes. The complex event pattern in BEMN for the example given in Section 1 is shown in Figure 2.

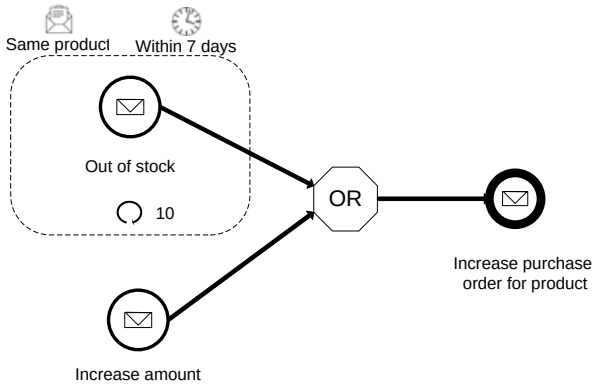

Figure 2. Example event pattern

Despite that the formal semantics of the language are defined and the execution environments are described, the original language did not take into account how current stream processing technique can be integrated to enable the execution of event models. As a result, some simplified matching functions are defined in the execution semantics to make only core event composition models executable. This will limit the expressiveness of executable event composition models and bring the overhead of translating general (non-core) models into core models. In our work, we align the semantics of event models with stream reasoning languages to make the models executable without such overhead.

Furthermore, BEMN execution environment exposes all the events to all the event patterns through a single event stream. This will bring efficiency issues as the matching functions need to filter out events from irrelevant sources. In our approach, we use the pub/sub messaging paradigm based on asynchronous web service interactions to create different event streams on-demand using WS-notification protocol, thus irrelevant events are filtered out before entering the event processing engine.

Moreover, BEMN language does not provide specification on data structure of event declarations. It is left to the programmers who implement the event streams. In this way, technical details are hidden from business users, with the price of compromising automation support for event pattern implementation. Also, it is difficult for the business users to create good event composition models without knowing what the primitive events really mean. We propose to use semantic web technology to help business users discover the primitive events they need, as well as create filters upon primitive event data. To this end, we refine the abstract syntax of event declarations and filters to allow more detailed definition of them.

 ISBN: 978-1-61208-236-3

VI. Pattern to Query Transformation

We will provide an algorithm based on Program Structure Tree (PST) to parse the event composition models and transform them into stream reasoning queries in a 'Divide-and-Conquer' style. First, we do not take parallel inhibition relation into consideration (an inhibition is considered parallel if when removed, source and target of this inhibition is still connected to start and end node, otherwise it is considered sequential), thus event patterns will be Directed-Acyclic-Graphs (DAGs), and we will traverse the DAG to find embedded Single-Entrance-Single-Exit (SESE) regions. Each SESE region can be sequential or branched. Each component in sequential SESE regions is connected with *SEQ* operators (as in [6]). Components in branched regions are translated into *GroupGraphPattern* (as in [7]) or *OPTIONAL* query patterns with filters on their occurrences (*bound()* filters). Then we deal with the inhibitions. Inhibition in sequential events will be transformed into optional patterns with *!bound()* filters. Inhibition in parallel will be defined using filters on timestamps. Finally, we will build aggregations and put filters to the correct scope. The query for the example in Figure 2 is listed in Listing 1.

Listing 1. Sample query for event pattern in Figure 2

```
SELECT ?pid, count(?x) as ?cnt
WHERE{
{STREAM <http://example.org/OutOfStock> [7 Days]
{?x a ces:OutOfStock;
            Evt: hasPayload [hasID ?pid]}}
UNION
{STREAM <http://example.org/IncreaseStorage> [NOW]
{?y a ces:IncreaseStorage;
             Evt:hasPayload [hasID ?pid]}}}
GROUPBY ?pid HAVING (count(?x)>10)
```

VII. Conclusions and Future Work

In this paper, we present a novel framework to facilitate user-centric definition and automated implementation of complex events based on ubiquitous data service. The user-centricity is achieved by revising a graphical notation of complex events called BEMN [5] which can be seamlessly integrated with BPMN and is targeted to business users. The automated implementation is realized by allowing detailed description of primitive events in event patterns and translating event patterns defined by business users to a stream reasoning query, so that they can be evaluated immediately without further coding.

Apart from the implementation and evaluation of the proposed system, future works may be explored in the following 3 aspects: support for multiple stream reasoning systems, navigation of sensor functionalities and creation of complex event hierarchies. We intend to provide support for multiple stream reasoning systems by creating profiles for the BEMN revision to align its semantics with different query languages. Primitive event service discovery is one of the key enabling techniques of automatic implementation. We aim to provide a navigation based discovery by modeling service capabilities and their relationships to construct a service capability hierarchy/graph, which can be navigated by business users. Currently, we assume all member events are primitive sensor events and we are not able to create event causal hierarchy, we intend to break this assumption to support more comprehensive event models in the future.

Acknowledgments

This work is supported by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2),

References

[1] M. Kurz and A. Fleischmann, "Bpm 2.0: Business process management meets empowerment," in *Subject-Oriented Business Process Management*, 2011.

[2] D. Luckham, "The power of events: An introduction to complex event processing in distributed enterprise systems," in *Rule Representation, Interchange and Reasoning on the Web*, 2008.

[3] F. Wang, S. Liu, P. Liu, and Y. Bai, "Bridging physical and virtual worlds: Complex event processing for rfid data streams," in *Advances in Database Technology - EDBT 2006*, 2006, pp. 588–607.

[4] J. Dunkel, "On complex event processing for sensor networks," in *International Symposium on Autonomous Decentralized Systems, 2009.*, 2009, pp. 1 –6.

[5] G. Decker, A. Grosskopf, and A. Barros, "A graphical notation for modeling complex events in business processes," in *EDOC 2007*, 2007, p. 27.

[6] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic, "Ep-sparql: a unified language for event processing and stream reasoning," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 635–644.

[7] D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth, "A native and adaptive approach for unified processing of linked streams and linked data," in *Proceedings of the 10th international conference on The semantic web - Volume Part I*, 2011, pp. 370–388.

[8] T. Moser, H. Roth, S. Rozsnyai, R. Mordinyi, and S. Biffl, "Semantic event correlation using ontologies," *On the Move to Meaningful Internet Systems OTM 2009*, pp. 1087–1094, 2009.

[9] Z. Li, C.-H. Chu, W. Yao, and R. a. Behr, "Ontology-Driven Event Detection and Indexing in Smart Spaces," *2010 IEEE Fourth International Conference on Semantic Computing*, pp. 285–292, 2010.

[10] K. Taylor, "Ontology-driven complex event processing in heterogeneous sensor networks," *The Semanic Web: Research and Applications*, pp. 285–299, 2011.

[11] S. Hasan, E. Curry, and M. Banduk, "Toward Situation Awareness for the Semantic Sensor Web: Complex Event Processing with Dynamic Linked Data Enrichment," *Semantic Sensor*, 2011.

 ISBN: 978-1-61208-236-3