A Framework for Event Processing on the Semantic Web

Srdjan Komazec
Supervised by Dieter Fensel
Phase II Submission

Semantic Technology Institute (STI) - Innsbruck
ICT Technologiepark, Technikerstrasse 21a, 6020 Innsbruck, Austria
srdjan.komazec@sti2.at

Abstract. The ubiquity of the Web has reached the point at which people and devices are producing numerous and overwhelming streams of information in the form of events. Early and intelligent detection of event pattern occurrences complemented with appropriate reactions can enable just-in-time responses to the situations of interest. This thesis is exploring the problems and possibilities stemming from the integration of event-driven reactivity techniques applied to the Semantic Web domain.

1 Problem Statement

Event processing stands for an established computing paradigm (best recognized by the seminal work of David Luckham [11]) which provides approaches and techniques to process event[1] streams and provide responses in a timely fashion. The main concerns of event processing comprise issues of exploring causal, temporal, and aggregative relationships between events with a strong emphasis on the real-time processing aspect. Established event processing application areas include manufacturing monitoring and control systems, location based services, financial data systems, defense intelligence, web analytics, and medical systems.

The introduction of the Web and its recent trends towards the Internet of Services and the Internet of Devices call for an application shift where event processing approaches are moving from closed and strictly controlled enterprise and military spaces towards the open, loosely governed and heterogeneous Web environment. In particular, the Web enables a dramatic increase in the number of observable events. Seen as a common platform to easily and cheaply exchange information, the Web of today is in fact already overwhelmed with events such as dissemination of new multimedia content, readings from the Sensor Web, and social networking activity notifications. Those events can be observed through various channels such as emails, RSS feeds and RESTful Web services and are

[1] According to [5] *an event represents anything that happens.* We are adopting this rather relaxed definition of event because it goes inline with the assumption-free characteristics of the Web.

targeted primarily for human consumption. The impact of a potential solution to tame the Web of Events can be substantial since there are already too many events on the Web and too less meaning of them.

Event processing at the Web scale is suffering from the genuine Web issues of heterogeneity and scalability, and stresses even more problems peculiar to the event processing itself such as correlation of events coming from different sources, identification and discovery of event sources. The Semantic Web, as an extension of the Web, provides a promising foundation to solve some of the issues of event processing over the Web. In particular, the problem of event heterogeneity reconciliation can be solved by the inclusion of machine processable data in the event description and the application of data mediation techniques. On top of it, powerful reasoning algorithms can be applied by the event processing agents in order to foster automation of the common event processing tasks, including event pattern and constraint matching, rule evaluation, aggregation computing, and proper time management.

In order to bring the vision of event processing on the Semantic Web to the reality, two fundamental challenges need to be met. First, event producers should enrich their event descriptions with machine processable semantics compliant to the Semantic Web languages. Second, event processing solutions need to integrate the Semantic Web technologies and languages.

The first challenge is not in focus of the thesis, since there are already first signs that semantically annotated events are finding their way towards the (Semantic) Web. With recent Twitter announcement of support to annotated tweets[2], adoption of the Open Graph Protocol[3] by Facebook, and wider usage of semantic technologies in various forms of sensor networks, the semantically annotated events are starting to pop-up.

The primary focus of this thesis is on the second challenge, where event processing techniques meet the Semantic Web. Both Semantic Web and Event Processing communities have recently shown an interest in integrating their results at various levels. The most prominent approaches stemming from the Semantic Web community are ETALIS [1, 2], ECA-LP and ECA-RuleML [15, 16], MARS [12], and recently C-SPARQL [3], while as for the Event Processing community the main stream of work is represented by XChange [7] and RDFTL [14].

Survey of the existing results has shown that presented solutions are partially solving some of the problems of event processing on the Semantic Web but that there exist no complete and sound framework to bring the power of event processing on the Semantic Web to its full potential. In particular, the solutions are rarely discussing the effects of underlying model theories and effect of

[2] `http://apiwiki.twitter.com/Annotations-Overview`
[3] `http://opengraphprotocol.org/`

materialization of implicit knowledge onto the mechanisms employed to exhibit event processing behavior. Furthermore, the solutions are almost exclusively approaching to the problem of event pattern detection in a backward-chaining fashion which includes serious performance penalties. It is also visible that current solutions are not tackling some phenomena peculiar to the event processing such as *parameter context*[4].

The gap between the Semantic Web and Event Processing can be filled with a comprehensive framework for event processing on the Semantic Web which builds upon best of breed from both areas and provides their deeper integration. The framework should cover the whole event processing life-cycle starting from the event pattern detection phase, checking for additional constraints and responding with proper activities. Such a framework would be useful in various application scenarios, ranging from filtering and processing social networking event streams, over integrating and aggregating events across the (Semantic) Web, towards managing semantically enriched enterprise SOA solutions.

2 Main Questions

A possible synthesis of the Semantic Web and Event Processing solutions should yield an approach which acknowledges the peculiarities of the Semantic Web in terms of existence of implicit knowledge and usage of reasoning techniques while providing a performant and scalable detection of event pattern occurrences and successive processing of those occurrences. In that sense, a possible solution represents a tradeoff between the expressivity of adopted Semantic Web language and retained performance and scalability. A general research question this thesis aims at answering is the following:

How are the underlying model theories of the Semantic Web languages coupled with the peculiarities of the event processing paradigm affecting the event pattern detection and event-driven reactivity mechanisms?

The effect of the presence of implicit knowledge on the complex event detection techniques has not been discussed thoroughly in the previous work. The process of knowledge base closure materialization can be time consuming (thus not amenable for real-time appliances) and can interfere with the event detection mechanisms. Solutions such as ETALIS [1, 2] and ECA-LP [15] are basically promoting the backward-chaining approach in which reactive rules are homogenized with deductive rules (i.e. rules capable of materializing new facts). In such a setting, the materialization of derived knowledge comes alongside with the detection of an event. However, the used backward-chaining approaches have proved not to be particularly suitable for event-driven applications, since rule

[4] Parameter context stands for a set of approaches to reduce consumed space and computational overhead by providing a mechanism to choose a meaningful subset of event occurrences.

evaluation is a time consuming and repetitive process (in particular, it depends on rule set interdependencies) and as such does not promote incremental evaluation of event patterns. Proper performance and scalability can be achieved only in terms of forward-chaining detection of event patterns where suitable techniques should be employed to inject support for incremental maintenance of materializations.

A set of more precise questions this thesis is targeting to answer can be derived from the general one:

1. *How to maintain truth inside of event knowledge base in the context of parameter contexts?* The usage of parameter context is enabling filtering of received events according to the particular need governed by an application domain. Since a detection of an event pattern can have multiple interpretations, the event processing engine must be fine-tuned to react only to recent occurrences of events, chronicle occurrences, cumulative occurrences, etc. It comes with surprise that none of the existing approaches is taking into account the issue of parameter contexts. Coupled with the possible existence of derived facts materialized through closure computation, the problem is getting even more evident as the inferred knowledge may influence the proper management of parameter context.
2. *How to maintain truth inside of event knowledge base in the context of timing windows?* The notion of timing windows provides a way to focus only on a time-bounded subset of recently harvested events. As such, proper management of materialized knowledge in course of timing window changes represents a sensitive issue, since expiration of a (possibly derived) statement truth in a knowledge base should be continuously monitored. Solutions such as C-SPARQL [3] are providing an approach to compute incremental changes in a secondary RDF store and then use this store for query evaluation. The problem with this approach is in the coarse grained time-frames during which the truth is maintained. The rapid arrivals of new events and short expiration times can impose a significant overhead in course of a proper query evaluation. This thesis aims to attack the problem directly on the primary structure used to incrementally evaluate event patterns, since this could improve the overall performance and maintain the truth more precisely than the previous approach.
3. *How to optimize event pattern detection structures in terms of multiple queries and derived knowledge?* The surveyed solutions are not in general discussing the opportunities stemming from the fact that interdependencies between multiple event patterns provide an opportunity to save computational resources when evaluating the intermediate results used by more than one pattern. When it comes to the forward-chaining handling of event patterns and network-based detection of event occurrences (like in RETE approach [6]) we can run into the same issues as in the case of question 1, i.e., computing the deductive closure can potentially extend the knowledge

base coverage and introduce a new range of difficulties in terms of sharing of intermediate event-detection results.

3 General Approach

In the heart of any event processing solution lay a language capable of declaratively describing arbitrary event patterns and an engine capable of interpreting such declarations and detecting occurrences of events fulfilling the patterns. As a matter of completeness and practical applicability, an event processing solution is usually an integral part of a more general event-driven reactivity framework which often follows the Event-Condition-Action[5] (ECA) paradigm. The framework plays the role of an interface point through which a particular system is monitored, observations about its behavior are collected in the form of events and appropriate activities are enacted in order to change the system state or influence its behavior.

Without major deviations from the aforementioned approach, the artifacts developed in course of this thesis are following the very same path. The central part of the thesis is related to the event pattern language and the engine capable of interpreting event pattern descriptions in order to actively search for the event occurrences fulfilling the defined pattern. The language will be built on top of the proven Semantic Web languages RDF(S) and SPARQL, and where applicable recent results in the area of RDF data stream querying will be used (such as C-SPARQL, which provides a mean to handle transient streams of RDF triples through the notions of timing windows, aggregate functions, etc). However, the detection engine will not rely on common approaches to query RDF repository but rather on the forward-chaining production rule-like mechanisms such as RETE [6], and its successors THREAT [13] or LEAPS [4]. The engine will integrate all the necessary features to properly address the research questions presented in Section 2. In order to address them, the thesis will follow the path established by [17] of incremental maintenance of ontology materializations computed as a result of deductive closures upon the event arrivals and try to apply them over the event detection network-structures.

In order to enable useful application of the complex event processing solution, an appropriate ECA rule engine is developed. While the *Event part* reuses the developed event pattern language and supporting detection engine, the *Condition part* is devoted to the traditional query evaluation over the persistent and volatile data detected in the previous step. The *Action part* is enabling enactment of different types of activities such as updates over data repositories or execution of management procedures over the observed system. The overall approach is presented in Figure 1.

[5] `http://en.wikipedia.org/wiki/Event_condition_action`

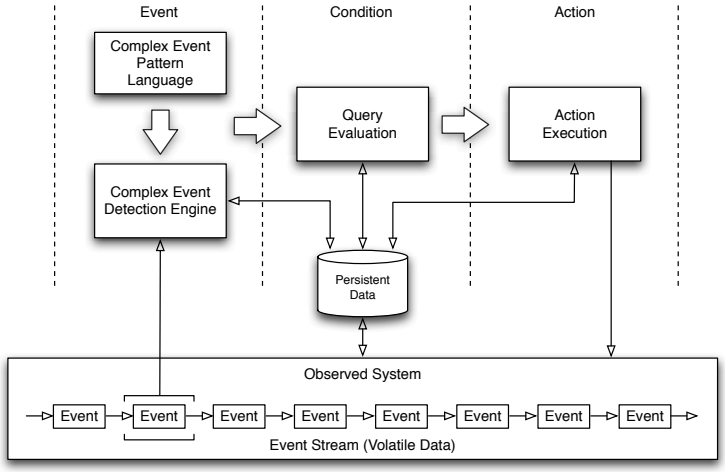

Fig. 1. The artifacts covered in course of the thesis.

4 Proposed Solution

The solution so far has focused on building a general ECA framework which provides a testbed for further extensions, quick implementation of the use cases, but above all the possibility to integrate and test the innovative solutions presented in Section 2.

The work related to this thesis has been conducted so far in course of two projects: Service Bundler[6], and COIN[7].

In Service Bundler project the ECA engine is used to process the information collected from a set of probes dedicated to observing behavior of single Web Services and enact appropriate activities upon detection of particular situations (e.g., updating aggregated statistics and recovering from Web service faults). In Service Bundler deliverable 3.3 [9] the syntax of ECA language has been defined where the *Event* and *Condition* part are represented as SPARQL SELECT and ASK queries, while the *Action* part provides for a possibility to create, update and delete data in an RDF storage, invoke external Web Services and generate and disseminate new events over an existing channel. Alongside with the syntax, the semantics of the langauge has been described, and an appropriate RDFS ontology governing the concepts related to the Web service monitoring had been developed. In addition, the API supporting object-oriented representation of ECA rules has been presented.

[6] `http://seekda.com/en/research/service-bundler`
[7] `http://www.coin-ip.eu`

In COIN project the very same ECA framework is used to monitor and react over the events emitted by the Semantic Execution Environment alongside with the execution of Semantic Web Services. The implemented ECA rules are again computing aggregated statistical values (like average/max/min invocation time, and overall/per user invocation counting) but also communicating to the external systems through Web service interfaces (i.e., sending notifications to a service reputation manager).

The realization of the core research issues related to semantically enhanced event pattern detection is still pending and will be conducted in the near future.

5 Evaluation

Evaluation of the thesis results targets to study and demonstrate usefulness of the solution by applying it to two application domains. In addition, the completeness and expressivity of the solution are evaluated through an implementation of the reference use-case in the event processing domain. At the end a performance study will be conducted.

5.1 Study of language expressivity in the context of reference use-case

In [8] Etzion et al. have presented an informal specification of a reference use-case which challenges various aspects of an event processing solution. The problem is build on top of the fast flower delivery use case in which various parties must be orchestrated at run-time by exchanging and reacting to the particular events in order to ensure high quality of the service. The use-case challenges expressivity of a complex event processing solution in several ways. First, it requires rather expressive language to describe event patterns which supports time windowing, correlations of events, and aggregation support. Second, since the overall system includes various event prosumers (flower storage, delivery vehicles, and the management system run by flower store association) certain support for interoperability and scalability is expected. Third, the system is also expected to take into account both volatile and persistent data (e.g., when reevaluating drivers rankings). Some existing solutions are already claiming to have an implementation of the reference use-case (like ETALIS[8]). A comparison of the solution provided in this thesis and the existing solutions is also considered.

5.2 Empirical study of implementation effectiveness

The effectiveness of the language, detection engine, and rule engine developed in the course of this thesis will be empirically studied through the implementation of two use cases coming from different domains.

[8] `http://code.google.com/p/etalis`

The first domain concerns with filtering and reactivity over semantically annotated streams of events produced by a social networking solutions such as Twitter. The possible application of the even-driven reactivity and event processing is in the domain of detecting particular social trends. The setup for the study is under development.

The second domain of application targets Semantically-enabled Service Oriented Architectures (SESAs) and in particular Semantic Execution Environment (SEE). So far SESA has neglected the importance of events generated by it. A SEE implementation could be improved by introducing event-based monitoring and reactivity in terms of auditing, run-time performance optimization, adaptation, and resilience to failures. As in the case of the previous application domain, the evaluation will cover study of the solution effectiveness when it faces the peculiarities of a SEE implementation, such as distributed execution, heterogeneous event descriptions, and performant event processing in case of burst event streams.

A part of the second use case has already been implemented and published in Komazec et al. [10]. As presented in Figure 2, all the SEE brokers are instrumented, thus capable of emitting events. The events are registered in the RDF repository governed by an appropriate ontology. Decisions regarding the actions taken upon sensing/deriving knowledge about the system are delegated to the Monitoring and Complex Event Processing component. The component detects (complex) situations of interest by consulting the registered events, analyzing them in some broader context (e.g. consulting additional knowledge) and selects/executes appropriate actions, which will be enacted over the system.

5.3 Complex event detection engine performance evaluation

An evaluation of the event detection engine performance will be carried out. Where applicable, the same measurements will be carried out over the matching solutions. A quantitative analysis such as throughput and scalability in the context of different parameters (allocated memory, frequency of events and complexity of event pattern descriptions) will be performed over the recorded measurements.

6 Conclusions and Future Work

This thesis is targeting to provide a solution for event processing in the context of the Semantic Web. In contrast to the previous approaches the thesis is aiming to provide a comprehensive event processing framework consisting of an expressive event pattern language and a novel event detection engine which is capable of maintaining the truth in the event knowledge base in course of rapidly changing data. The event pattern detection mechanism is complemented by an ECA rule-based solution, which enables application of the event processing in

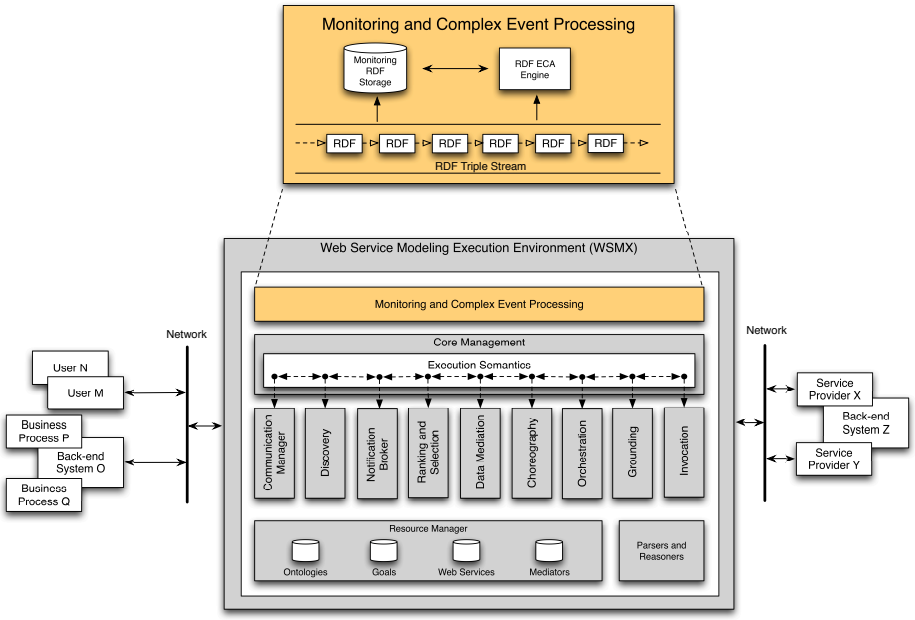

Fig. 2. A SEE implementation enriched with the ECA rule engine.

various domains such as social networking and semantically enriched SOA system monitoring.

Work on the thesis has reached the point in which the working environment has been set up and elaboration of answers is in the initial stage. In particular, the near future will yield the work focusing on the core thesis problems related to the influence of advanced aspects, such as timing windows, parameter contexts and multiple query optimizations, on the event pattern detection process in presence of derived knowledge.

Acknowledgments This work was supported by the COIN project (EU FP7 Project 216256; www.coin-ip.eu) funded by the European Community within the IST-Programme of the 7th Framework Research Programme.

References

1. Darko Anicic, Paul Fodor, Roland Stuhmer, and Nenad Stojanovic. Efficient Logic-Based Complex Event Processing and Reactivity Handling. Technical report, FZI Forschungszentrum Informatik, 76131 Karlsruhe, Germany and State University of New York at Stony Brook, USA, 2009.
2. Darko Anicic, Paul Fodor, Roland Stuhmer, and Nenad Stojanovic. Event-Driven Approach for Logic-Based Complex Event Processing. In *CSE '09: Proceedings*

of the 2009 International Conference on Computational Science and Engineering, pages 56–63, Washington, DC, USA, 2009. IEEE Computer Society.
3. Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. volume 6088/2010 of *Lecture Notes in Computer Science*, chapter Incremental Reasoning on Streams and Rich Background Knowledge, pages 1–15. Springer Berlin / Heidelberg, 2010.
4. Don Batory. The leaps algorithm. Technical report, Austin, TX, USA, 1994.
5. K. Chandy and W. Schulte. *Event Processing: Designing IT Systems for Agile Companies*. McGraw-Hill, Inc., New York, NY, USA, 2010.
6. Robert B. Doorenbos. *Production Matching for Large Learning Systems*. PhD in Informatics, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1995.
7. Michael Eckert and François Bry. Aktuelles Schlagwort: Complex Event Processing (CEP). *Informatik Spektrum*, 32(2):163–167, 2009.
8. Opher Etzion and Peter Niblett, editors. *Event Processing in Action*. Manning Publications Co., Berlin, Heidelberg, 2010.
9. Srdjan Komazec. Deliverable 3.3 - Monitoring Language and API. Technical Report 3.3, Semantic Technology Institute, University of Innsbruck, 2010.
10. Srdjan Komazec and Federico Michele Facca. Towards a reactive semantic execution environment. In Robert Meersman, Pilar Herrero, and Tharam S. Dillon, editors, *OTM Workshops*, volume 5872 of *Lecture Notes in Computer Science*, pages 877–887. Springer, 2009.
11. David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
12. Wolfgang May, Franz Schenk, and Elke von Lienen. Extending an OWL Web Node with Reactive Behavior. In Jos Jlio Alferes, James Bailey, Wolfgang May, and Uta Schwertel, editors, *Proceedings of Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2006)*, volume 4187 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2006.
13. Daniel P. Miranker. Treat: A better match algorithm for ai production systems; long version. Technical report, Austin, TX, USA, 1987.
14. George Papamarkos, Alexandra Poulovassilis, and Peter T. Wood. RDFTL : An Event-Condition-Action Language for RDF. In *In Proc. 3rd Int. Workshop on Web Dynamics (in conjunction with WWW2004*, pages 223–248, 2004.
15. Adrian Paschke. ECA-RuleML: An Approach combining ECA Rules with temporal interval-based KR Event/Action Logics an Transactional Update Logics. Technical Report 11/2005, IBIS, Technische Universitaet Muenchen, 2005.
16. Adrian Paschke. ECA-LP / ECA-RuleML: A Homogeneous Event-Condition-Action Logic Programming Language. *CoRR*, abs/cs/0609143, 2006.
17. Raphael Volz, Steffen Staab, and Boris Motik. Incrementally Maintaining Materializations of Ontologies Stored in Logic Databases. *Data Semantics II-LCNS*, 3360:1–34, 2004.