Twelve Theses on Reactive Rules for the Web

François Bry and Michael Eckert

University of Munich, Institute for Informatics
Oettingenstr. 67, D-80538 München
{bry, eckert}@pms.ifi.lmu.de
http://www.pms.ifi.lmu.de

Abstract. Reactivity, the ability to detect and react to events, is an essential functionality in many information systems. In particular, Web systems such as online marketplaces, adaptive (e.g., recommender) systems, and Web services, react to events such as Web page updates or data posted to a server.
This article investigates issues of relevance in designing high-level programming languages dedicated to reactivity on the Web. It presents twelve theses on features desirable for a language of reactive rules tuned to programming event-driven Web and Semantic Web applications.

This is an extended abstract of a talk given during the Dagstuhl Seminar "Event Processing." A longer version of this work, which explains and justifies each thesis in detail, has been presented with the same title at the workshop "Reactivity on the Web" at EDBT 2007, see [1].

1 Introduction

A common perception of the Web is that of a distributed repository of hypermedia documents with clients (in general browsers) that download documents, and servers that store and update documents. Although reflecting a widespread use of the Web, this perception is not completely accurate.

In fact, many Web applications build upon servers or clients updating data in reaction to events or to messages exchanged on the Web. Examples are online marketplaces, adaptive (e.g., recommender) systems, and Web services. The Web's communication protocol, HTTP, provides an infrastructure for exchanging events or messages. In addition, SOAP provides conventions for exchanging structured and typed information on the Web as XML messages. For transport of messages between Web nodes, SOAP can use HTTP (or other protocols).

This article first argues that complementing HTTP and SOAP with high-level languages for updates and reactivity is needed for both standard Web and Semantic Web applications. It then presents twelve theses on features desirable for a language of reactive rules tuned to programming Web applications.

The views reported about in this article have emerged during the design of the Web and Semantic Web query language Xcerpt [2, 3] and of the reactive Web language XChange [4, 5], as well as from experiences with programming in Xcerpt and XChange [6, 7].

2 Motivation and Background

Many Web applications, such as online marketplaces, e-learning systems, recommender systems, and communication platforms (Wikis, forums, etc.), build upon complex reactions to messages or events as well as updates to Web data (HTML, XML, RDF, OWL, etc.). Cost-efficient development of such Web applications requires high-level languages tailored to updates and reactivity. Although HTTP and SOAP provide a communication infrastructure that helps implementing reactivity on the Web, more abstract and higher-level languages are needed that

- abstract away network communication and system issues,
- ease the specification of complex updates of Web resources (e.g., XML, RDF, and OWL data),
- are convenient for specifying complex flows of actions and reactions on the Web.

The need for high-level Web update and reactive languages is similar to the need for high-level (Semantic) Web query languages (see [8] for a survey). High-level reactive languages will complement, not replace, HTTP and SOAP.

3 The Need for ECA Rules on the Web

Thesis 1: High-level reactive languages are needed on the Web. Event-Condition-Action rules are well-suited to specify reactivity on the Web. In particular, they are better suited than production rules for a large class of Web applications.

A rule-based approach to reactivity on the Web provides the following benefits over the conventional approach using (imperative or object-oriented) general purpose programming languages:

- Rules are easy to understand for humans. Requirements specification often already comes in the form of rules expressed either in a natural or formal language.
- Rule-based specifications are flexible, therefore easy to adapt, alter, and maintain as requirements change, which is quite frequently the case with business rules.
- Rules are well-suited for processing and analyzing by machines. Methods for automatic optimization, verification, and transformation into other types of rules (e.g., derive ECA rules from integrity constraints) have been well-studied and applied successfully in the past.
- Rules can be managed in a single rule base as well as in several rule bases possibly distributed over the Web.

On the Web, reactive rules explicitly referring to events, i.e., Event-Condition-Action (ECA) rules, are more appropriate than production rules without explicit reference to events for the following reasons:

– "Real-world reactive rules" often come with an explicit specification of an event, for example: "a credit card application (event) will be granted (action) if the applicant has a monthly income of more than EUR 1 500 and no outstanding debts (condition)."
– Events exchanged as messages between Web nodes are a natural, high-level communication paradigm, also exploited in Service-Oriented and Event-Driven Architecture.
– Events can carry data between Web nodes that is relevant for the condition and action part of a rule.
– ECA rules allow an easy handling of errors and exceptional situations that can conveniently be expressed as (special) events.

Thesis 2: Given the loosely coupled nature of the Web, reactive rules should be processed locally. They act globally through event-based communication and access to persistent Web data.

4 Event Communication Paradigms

Thesis 3: Events are best exchanged directly between Web sites in a push manner.

Periodical polling, where interested Web sites retrieve remote Web resources periodically to check if an event has happened, is less favorable since it causes more network traffic, increases reaction time, and requires more local resources.

5 Specifying Composite Events: Towards High-Level Event Query Languages

Thesis 4: Events are volatile data and should be kept distinct from persistent data.

On a reactive Web, there are two kinds of data: "normal" data from Web resources such as XML or RDF documents ("persistent data") and data from events ("volatile data"). "Normal" Web data is retrieved upon request in a pull manner, *persistent*, and can be *modified*. It typically signifies a *state* of (an abstraction of) the world. Event data is communicated between Web nodes (typically in a push manner), *volatile*, and *not modifiable*. It is typically used to signal *changes in state*. Due to their different nature, there should be a clean separation of persistent Web data and volatile event data in a reactive language.

Thesis 5: Recognizing composite (or complex) events is essential for a reactive Web language. Composite events are conveniently specified by (event) queries. There are (at least) four complementary dimensions to event queries: data extraction, event composition, temporal conditions, and event accumulation.

Thesis 6: A data-driven, incremental evaluation of event queries is the approach of choice.

Incremental evaluation avoids re-computation of intermediate results whenever new events arrive. In contrast, a standard, "query-driven" approach would perform a full re-evaluation of the query against the history of events received so far for each new event.

6 Specifying Conditions: Embedding a Web Query Language

Thesis 7: Data from persistent Web resources plays an essential role for Web reactivity. A reactive language thus should embed or build upon a Web query language.

A reactive Web language has to integrate in the current Web of retrievable, persistent data sources. Programmers must be able to easily access and query persistent Web data. Instead of "reinventing the wheel," a reactive language should embed or build upon an existing Web query language.

7 Specifying State-Changing Actions

Thesis 8: The Web is a dynamic, state-changing system. Reactions to state changes (events) through reactive rules are state-changing actions such as updates to persistent data. Reactive rules are needed where compound actions can be constructed from primitive actions.

8 Structuring Rules and Rule Programs

Thesis 9: Like in any other programming language, Development and maintenance of reactive rule programs can be considerably supported by structuring mechanisms such as: branching in rules, deductive rules for event queries and Web queries, procedural abstractions for actions, and grouping of rules.

9 Miscellanea

Thesis 10: Identity of data items is an issue for reactive languages due to their ability to react to changes of data objects on the Web.

Reactive languages with the ability to monitor data items (or objects) and react to their changes need to deal with identity of the data items. There are basically two approaches to identity: extensional identity (as part of the data, e.g., primary keys in relational databases) and surrogate identity (independently from the data, e.g., pointers or references to objects in object-oriented programming languages).

For monitoring changes of objects, surrogate identity is advantageous. However, to communicate with remote Web sites, surrogate identity has to become part of the data, i.e., made extensional. Even worse, Web resources such as XML or RDF documents usually do not provide a surrogate identity for their data at all and only rarely provide auxiliary identity-defining attributes (keys such as `xml:id` attributes) as part of the extension.

Thesis 11: Meta-programming and meta-circularity, that is, the ability to use rules to exchange and evaluate (other) rules, are needed in some important cases.

A particular example of this are (automatic) policy-based trust negotiations [9]. Negotiating partners do not give out all their policies at once, since policies themselves can be sensitive information and thus only given out when a certain stage in the negotiation (e.g., trust level) has been reached. Instead they exchange policies reactively during the course of the trust negotiation.

Thesis 12: Reactivity in the Web's open and uncontrolled world requires language support for authentication, authorization, and accounting.

10 Conclusion

In this article we have presented twelve theses on reactive rules for the Web. We have argued that reactivity in the Web needs reactive rules, in particular ECA rules, and established a list of desiderata for reactive, ECA-rule-based languages.

Many of the desiderata postulated in this article are very general. They apply not only to reactive languages based on ECA rules, but also to other rule-based reactive languages (e.g., based on production rules) and even languages, frameworks, and program libraries not based on rules at all.

Acknowledgments

The ideas expressed in this article have been significantly influenced by the research project REWERSE (Reasoning on the Web with Rules and Semantics, http://rewerse.net) and the W3C RIF Working Group (http://w3.org/2005/rules). The authors thank their colleagues of REWERSE and of the W3C RIF Working Group for many fruitful exchanges on the subject of this article.

The authors thank Tim Furche, Paula-Lavinia Pătrânjan, and Inna Romanenko for their insights and numerous discussions.

References

1. Bry, F., Eckert, M.: Twelve theses on reactive rules for the Web. In: Proc. Int. Workshop Reactivity on the Web at EDBT 2006. Volume 4254 of LNCS, Springer (2006)

2. Schaffert, S., Bry, F.: Querying the Web reconsidered: A practical introduction to Xcerpt. In: Proc. Extreme Markup Languages. (2004)
3. Xcerpt. http://xcerpt.org (2006)
4. Bailey, J., Bry, F., Eckert, M., Pătrânjan, P.L.: Flavours of XChange, a rule-based reactive language for the (Semantic) Web. In: Proc. Intl. Conf. on Rules and Rule Markup Languages for the Semantic Web. Volume 3791 of LNCS, Springer (2005)
5. Bry, F., Eckert, M., Pătrânjan, P.L.: Reactivity on the Web: Paradigms and applications of the language XChange. J. of Web Engineering **5**(1) (2006) 3–24
6. Kraus, S.: Use Cases für Xcerpt: Eine positionelle Anfrage- und Transformationssprache für das Web. Master's thesis (in German), Inst. for Informatics, Univ. of Munich (2004)
7. Romanenko, I.: Use cases for reactivity on the Web: Using ECA rules for business process modeling. Master's thesis, Inst. for Informatics, Univ. of Munich (2006)
8. Bailey, J., Bry, F., Furche, T., Schaffert, S.: Web and Semantic Web query languages: A survey. In: Reasoning Web, Int. Summer School. Volume 3564 of LNCS, Springer (2005) 35–133
9. Winslett, M.: An introduction to trust negotiation. In: Proc. Int. Conf. on Trust Management (iTrust). Volume 2692 of LNCS, Springer (2003) 275–283