Evolution of Distributed Web Data: An Application of the Reactive Language XChange

François Bry Michael Eckert Hendrik Grallert Paula-Lavinia Pătrânjan
University of Munich, Institute for Informatics
http://www.pms.ifi.lmu.de, {bry,eckert,grallert,patranjan}@pms.ifi.lmu.de

Abstract

Many data sources on the Web evolve: they change their content over time, typically as reactions to events. Such changes often need to be mirrored in data on other Web nodes: updates need to be propagated. To respond to the need for evolution and reactivity both locally and globally, the language XChange has been developed. We demonstrate its applicability in a concrete scenario of distributed Web sites of a scientific community with mutual data dependencies.

1. Introduction

Many data sources on the Web and Semantic Web are evolving in the sense that they change their content over time in reaction to events bringing new information [2]. Often, such changes must be mirrored in data on other Web nodes: updates need to be propagated.

The reactive, rule-based language XChange [3] has been developed to respond to the need for both local (at a single Web node) and global (distributed over several Web nodes) evolution and reactivity on the Web. Borrowing ideas from active database systems, XChange is a language of Event-Condition-Action (ECA) rules. XChange is tailored for the distributed nature of the Web and for common Web data formats by allowing event-based communication and by embedding the versatile Web query language Xcerpt [6, 7].

Our demonstration shows how XChange can be applied to programming reactive Web sites where data evolves locally and, through mutual dependencies, globally. The setting we consider are several distributed Web sites of a fictitious scientific community of historians called the Eighteenth Century Studies Society (ECSS). ECSS is subdivided into participating universities, thematic working groups, and project management. Universities, working groups, and management have each their own Web site. The different Web sites are autonomous, but cooperate to evolve together and mirror relevant changes from other Web sites. For example, Web sites maintain data about members; a change of member data at a university entails further changes at the Web sites of the management and some working groups.

2. The Language XChange

We refer to [3] for a full introduction to XChange, and only describe its foundations and benefits here:

(i) XChange programs consist of ECA rules. These allow programming on a high abstraction level and are easy to analyze for both humans and machines.

(ii) XChange embeds the versatile Web query language Xcerpt [6, 7] to access and reason with Web data, and also provides an integrated update language (based on Xcerpt) for modifying Web data. All parts of a rule follow the same paradigm of specifying patterns for tree- or graph-structured data (e.g., XML, RDF) that is queried, newly constructed, or updated. This makes XChange elegant and easy to learn.

(iii) Both atomic and composite events can be detected and relevant data extracted from events [3, 1].

(iv) XChange enforces a clear separation of persistent data (Web resources, relating to *state*) and volatile data (events, relating to *changes in state*).

(v) XChange's high abstraction level and its powerful constructs allow for short and compact code.

An XChange program is located at one Web node and consists of rules of the form `ON` *Event query* `IF` *Web query* `DO` *Action*. Such an ECA rule means: when events answering the event query are received and the Web query evaluates successfully, perform the action. Both event and Web query can extract data through variable bindings, which can then be used in the action. With this, we can see that both event and Web queries serve a double purpose of detecting *when* to react and, through binding variables, *how* to react.

For distributed applications XChange programs at different Web sites coordinate by sending and receiving events as XML messages in a push-manner. We see this as an important advantage of ECA rules over production rules [2].

3. Description of our Demonstration

Our demonstration applies the ECA rule language XChange to the distributed Web data of ECSS, as described in Section 1. Data evolves locally and updates are propagated globally by means of event messages. The Web sites of universities, working groups, and management are autonomous, but cooperate to evolve together and mirror relevant changes.

$r1$: ON change member
DO update LMU data
$r2$: ON change member
DO forward to management
$r3$: ON change member
DO update management data
$r4$: ON change member (w/WG3)
IF was not member of WG3
DO send add member to WG3
$r5$: ON change member (w/o WG2)
IF was member of WG2
DO send remove member to WG2
$r6$: ON remove member
DO update WG2 data
$r7$: ON add member
DO update WG3 data

Figure 1. Example: Affected Rules

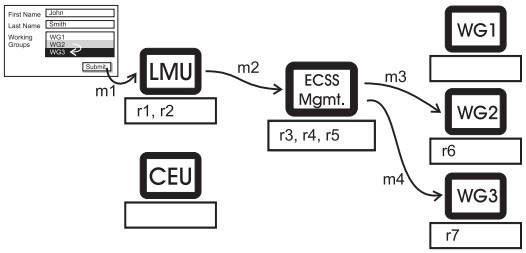

Figure 2. Example: Distributed Event Flow

The different Web sites maintain XML data about members, publications, meetings, library books, and newsletters. Data is often shared, for example a member's personal data is present at his home university, at the management node, and in the working groups he participates in. Such shared data needs to be kept consistent among different nodes; this is realized by communicating changes as events between the different nodes using XChange ECA rules.

Events that occur in this community include changes in the personal data of members, keeping track of the inventory of the community-owned library, or simply announcing information from email newsletters to interested working groups. These events require reactions such as updates, deletion, alteration, or propagation of data, which are implemented using XChange rules. The rules run locally at the different Web nodes of the community, allowing for the processing of local and remote events.

Figure 1 gives an example of the rules triggered when a member's data, including working group affiliation, is changed. The distribution of the rules and the flow of events between the different Web nodes is shown in Fig. 2.

The initial change is made with a Web form at the member's home institution, sending event $m1$ to the LMU node. There, rule $r1$ reacts and locally updates the member's data at LMU accordingly, while rule $r2$ forwards the change to the management node as event $m2$. The management node has rules for updating its own local data about the member ($r3$) and for propagating the change to the affected working groups ($r4$ for adding, $r5$ for deleting a member). The working groups finally each have rules ($r6$, $r7$) reacting to deletion and insertion events ($m3$, $m4$).

Apart from this simple example of managing distributed member data, our demonstration realizes a community-owned and distributed virtual library, meeting organization, and newsletter distribution. It is fully described in [5]. For presentation purposes, the facilities for displaying the rules of each node and logging received and sent events are available.

4. Conclusions

While a similar behavior as the one in the demo could be obtained with conventional programming languages, XChange provides an elegant and easy solution. Evolution of data and reactivity on the Web are easily arranged for by using readable and intuitive ECA rules. Moreover, by employing and extending Xcerpt as a query language, XChange integrates reactivity to events, querying of Web resources, and updating those resources in a single, easy-to-learn language.

The demo presented here has been implemented in the framework of a three months independent study project. The student has had no prior experience with XChange, Xcerpt, and rule-based programming (including ECA rules). Out of the three month, large parts were dedicated to designing the use case from scratch; the actual rule authoring consumed less than one month. Judging from the learning curve, we estimate that adding a new task to the demo (such as managing reports that are delivered to the funding agency of the ECSS) could be done within only two or three days now.

XChange is an ongoing research project [8]; a prototype implementation is available and used to run the demonstration. Development focuses on further use cases and structuring rule sets [4], automatic generation of ECA rules, querying of composite events [1], and efficient evaluation.

Acknowledgments

This research has been funded by the European Commission and the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (`http://rewerse.net`).

References

[1] F. Bry and M. Eckert. A high-level query language for events. In *Proc. Int. Workshop on Event-driven Architecture, Processing and Systems at Int. Conf. on Web Services*. IEEE, 2006.
[2] F. Bry and M. Eckert. Twelve theses on reactive rules for the Web. In *Workshop "Reactivity on the Web" at Int. Conf. Extending Database Technology*. Springer, 2006. (Invited paper).
[3] F. Bry, M. Eckert, and P.-L. Pătrânjan. Reactivity on the Web: Paradigms and applications of the language XChange. *Journal of Web Engineering*, 5(1):3–24, 2006.
[4] F. Bry, M. Eckert, P.-L. Pătrânjan, and I. Romanenko. Realizing business processes with ECA rules: Benefits, challenges, limits. In *Proc. Int. Workshop on Principles and Practice of Semantic Web*. Springer, 2006.
[5] H. Grallert. Propagation of updates in distributed web data: A use case for the language XChange. Project thesis, Inst. f. Informatics, U. of Munich, 2006.
[6] S. Schaffert and F. Bry. Querying the Web reconsidered: A practical introduction to Xcerpt. In *Proc. of Extreme Markup Languages Conf.*, 2004.
[7] Xcerpt. http://xcerpt.org.
[8] XChange. http://www.reactiveweb.org/xchange.