Processing Distributed Event Data and Multifaceted Knowledge in a Collaboration Federation

Stanley Y.W. Su, Seema Degwekar, Amit Sardesai
Database Systems R&D Center
Department of Computer and Information Science and Engineering
University of Florida
{su,spd,as5}@cise.ufl.edu

Howard Beck
Agricultural and Biological Engineering Department
Institute of Food and Agricultural Sciences
University of Florida
hwb@ufl.edu

Abstract

All nations are facing many global problems, the solutions to which require efficient and effective sharing of distributed, heterogeneous data, knowledge and application systems. We present a way to capture organizations' multi-faceted knowledge by three popular rule types and rule structures, and wrap them as Web Services for their registration, discovery and invocation. Distributed data associated with events defined by these organizations are transmitted through a distributed event infrastructure to those sites that contain applicable rules. The processing of heterogeneous rules and application operations specified in rules may add to or modify the event data to produce a dynamic event data set that can be used to support collaborating organizations' decision-making and problem solving. A peer-to-peer architecture of a distributed system and the functions of its components are described. Issues and approaches related to event data evolution and distributed rule and trigger processing are also discussed.

1. Introduction

All nations in the world are facing global problems such as border control and immigration, terrorism, plant disease and pest diagnostics, and others. The solutions to these complex problems require many organizations within a country as well as across national boundaries to effectively and efficiently share, not only heterogeneous data, but also knowledge rules and application systems in highly distributed and dynamic environments. Although techniques for accessing heterogeneous data stored in distributed databases have been well studied [1, 2] and centralized knowledge management systems designed for processing a specific type of knowledge rules also exist, techniques, tools and collaborative systems for specifying, processing and managing what we shall call "**event data**" and "**multifaceted knowledge**" in a "**Collaboration Federation**" are still lacking.

An **event** is anything of interest (e.g., a sensor signal, user input, or a real-world incident) to collaborating organizations that occurs at a particular point in time. It usually has data associated with each of its occurrences. The data needs to be sent to subscribing organizations by an event notification mechanism. Upon the receipt of an event notification, each organization may want to invoke some knowledge rules and automated or manual operations to operate on the data. The rules and operations may generate more data or modify the received data. By **event data**, we mean a dynamic data set that contains the initial data associated with an event occurrence and the data evolved by applying different types of rules and application operations.

By **multifaceted knowledge**, we mean an organization's knowledge specified in terms of different types of knowledge rules and rule structures. Human knowledge can be specified in different formal knowledge representations [3], which differ in their intended use, expressive power, and features. Among them, three general knowledge representations are widely used in implemented systems: constraint rule, logic-based derivation rule and event-condition-action (ECA) rule [4, 5]. The specification and processing of different types of rules and rule structures is important for the following reasons. First, rules are declarative specifications of organizations' policies, regulations and constraints, which are easier to understand and modify than program code that implements them. Second, it is very difficult and, in many cases, impossible to express the semantics of different types of rules using a single knowledge representation to be processed by a single rule engine.

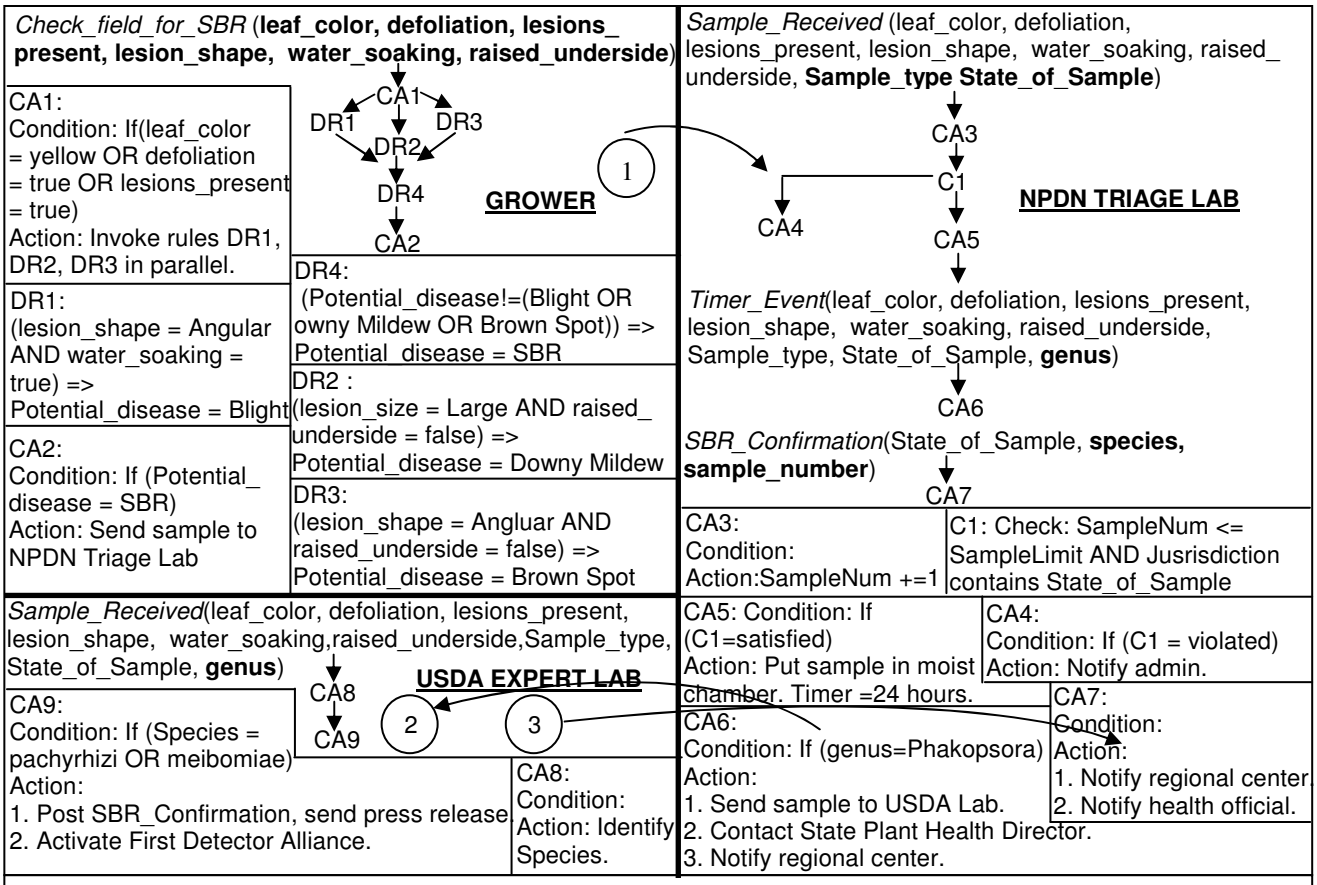

Figure 1. Example scenario of soybean rust confirmation.

Third, a structure captures the order in which these rules should be processed.

By **collaboration federation**, we mean a number of collaborating but autonomous organizations that are connected by the Internet and have agreed to share event data, multifaceted knowledge, and application resources to coordinate their activities and perform their functions. Each federation may publish and subscribe to distributed events, and define, publish, exchange and apply distributed, heterogeneous knowledge rules and rule structures. The goal of this R&D effort is the creation of an infrastructure and the development of technologies to process and manage distributed events, multifaceted knowledge and application operations in a collaboration federation.

2. Collaboration Federation and Scenario

A good example of a collaboration federation is a federation of organizations that the authors have been involved in. The U.S. Department of Agriculture Cooperative State Research, Education and Extension Service launched a multi-year national project in May 2002 to build the National Plant Diagnostic Network (NPDN) to strengthen the homeland security protection for the nation's food and agriculture by facilitating quick and accurate detection of disease and pest outbreaks in crops. Such outbreaks can occur as foreign pathogens are introduced into the U.S. either through accidental importation, by wind currents that traverse entire continents, or by an intentional act of bioterrorism [6].

In this nation-wide effort, the University of Florida (UF) was selected as the southern regional center of NPDN, responsible for building a Southern Plant Diagnostic Network (SPDN). SPDN consists of a regional center system developed at UF, which connects the systems in 11 other states and Puerto Rico. Data collected from the region is sent to the NPDN national data center at Purdue University. Since plant samples and analysis results need to be transmitted to many organizations for analyses and use, there is a need for event notification, automatic delivery of event data, and distributed processing of multifaceted knowledge and application operations among these organizations. A prototype system, which uses an Event-Trigger-Rule Server [7, 8] to process condition-action rules in the NPDN environment, has been reported in [9]. This work is an extension of the prototype.

One of the recent concerns, which can benefit from the technologies being developed, is the identification, diagnosis and management of one particular disease – Soybean Rust. The USDA has established general procedures for alert notification and response to soybean rust, and each state is working on standard operating procedures to handle outbreaks. We shall use this as an example to demonstrate our system.

For this scenario of Soybean Rust (SBR) identification, diagnosis and management, the three collaborating organizations of importance are the Grower, the NPDN Triage Lab and the USDA Expert Lab. The grower initiates SBR identification by checking for known SBR symptoms in the field. If he/she cannot eliminate the possibility of SBR, the field sample is sent to the NPDN Triage Lab for testing. In this example, the event is named *Check_field_for_SBR* and its event data are specified by the list of attributes shown in Figure 1. The procedure that the grower follows is specified by a rule structure shown in the figure. These rules are given in a notation that is easier for the reader to understand; they are not specified in the rule markup language we have defined. We note here that, in a distributed environment, we allow events and condition-action rules to be defined independently by collaborating organizations to facilitate their reuse. Triggers are specifications that link distributed events to distributed rules. They can also be independently defined and processed in a distributed manner.

The condition-action rule CA1 checks for some initial symptoms. The derivation rules DR1, DR2, and DR3 are fired in parallel to determine if the symptoms match with those of the known but non-threatening diseases – Blight, Downy Mildew, or Brown Spot, respectively (used to eliminate the possibility of SBR). The event data generated by these rules are used by the derivation rule DR4. The CA2 rule that follows calls for sending the sample to the NPDN Triage Lab for testing. This flow of information is labeled by (1) in Figure 1.

The Triage Lab's process can again be specified by a rule structure. When the NPDN Triage Lab receives the sample, it posts the event *Sample_Received*, which invokes CA3 to increment the sample count, and the constraint rule C1 to make sure that the lab constraints (whether the sample is within the number that can be handled by the lab, and if it is from the state within its jurisdiction) are satisfied. If not, the administrator is notified (CA4). Otherwise, the sample is put in a moist chamber and a timer is set to 24 hours (CA5). This is done to allow the pathogen to multiply to facilitate identification and testing. After 24 hours, a timer event is posted to trigger CA6, which verifies if the genus [10] is Phakopsora. If so, further specification is necessary to confirm that it is indeed a soybean rust-causing pathogen, and the sample is forwarded to a USDA expert lab (information flow 2). Upon receiving the sample, the USDA Expert Lab identifies the species of the pathogen (CA8), and if SBR is confirmed, posts the SBR-Confirmation event, sends out a press release and activates the First Detector Alliance (CA9). The event will also trigger the processing of CA7 at the Triage Lab to send out notifications to regional centers and health officials (information flow 3). At each stage, new event data (shown in bold font in the figure) are added to the initial event data and continue to evolve by the invocation of related rules and operations. By using the scenario given above, we have shown 1) procedures and policies of collaborating organizations can be specified at a high-level using different types of rules and rule structures, 2) an event can trigger the processing of distributed rules, which may post other events to trigger other rules, 3) event data, which are dynamically generated by rules and operations invoked by rules, constitute all the required information that can be used by collaborating organizations for decision making and problem solving, and 4) event notifications can keep all relevant organizations informed and coordinated.

3. System Architecture, Issues and Approaches

The collaboration network being developed has a peer-to-peer architecture. The replicas of the software components and tools shown in Figure 2 will be installed at all collaboration sites. A network of microcomputers that run the software and tools form an overlay network in a wide-area network like Internet, to which application systems of these organizations are connected. The components shown at the top left corner of the figure form Subsystem 1, which is replicated at all collaborating sites. The components shown at the bottom left corner form Subsystem 2, which are installed at the host site of the federation. Each collaborating organization will use the user interface tools to define parameterized events, corresponding event filter templates (used by subscribers to specify their event filters), rules and triggers.

A typical runtime scenario is as follows. An event occurs at a collaborating site. It is published by calling the local Event Server, which has the information about the event subscribers and their event filters (downloaded from the host site either at the event subscription time or, in some cases, at runtime). The Event Server, which serves as the coordinator of the event, would process the information and send the initial event data to all those subscribers whose filtering conditions are satisfied (i.e., the participants). The participant Event Servers will also be notified. They invoke the corresponding local Rule Servers to process those rules that are tied to the event by trigger specifications (see Section 3.2 for more details about triggers). New data produced by derivation rules

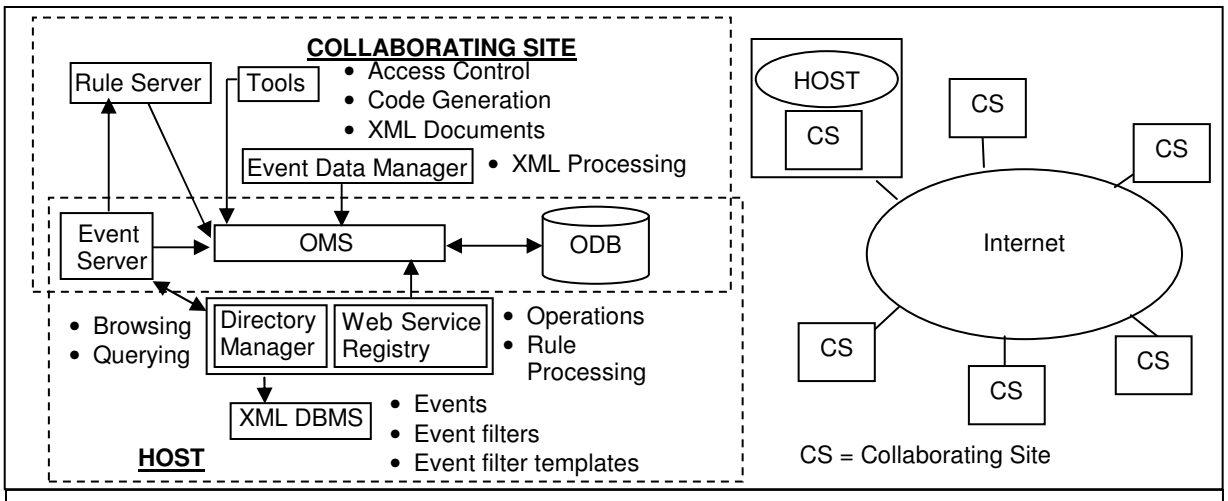

Figure 2. Architecture of the collaboration federation

and condition-action rules at each subscriber site is returned to the coordinator, which then aggregates/merges the new data with the initial event data to form a new version of event data. This new version may make other distributed rules applicable and thus would trigger another round of event notification and rule invocation. Several iterations of the above process may take place until no new event data is generated and no modification is made to the last version by rules. At this time, all subscribers of the event would have received all the relevant event data.

Events, rules, and triggers can either be private to an organization or shared within the federation. Shared event and filter template specifications in XML format are registered with the Directory Manager of subsystem 2, and managed by an XML database management system (the Apache Xindice). Authorized users of collaborating organizations can search or browse the event directory, subscribe to shared events and define event filters. In the scenario explained above, event data is transmitted by recording it in an XML document, which is constantly updated as relevant rules are processed. Private events are visible only to the organization that defines them. They will trigger the processing of private rules.

Similarly, rules as well as triggers can be private or shared. Sharable rules are translated into programs, the input and output specifications are determined based on the rule specifications. They are wrapped as Web Services with a given name and an access point, and registered with the Web Service Registry of the host site. The rule structure specified in a shared trigger can thus make references to sharable rules. At runtime the Rule Server invokes these distributed Web Services. In effect, individual shared rules become simple Web Services and a structure of rules specified in a shared trigger forms a composite Web Service, which is also registered with the Registry. Authorized users of collaborating organizations can search and browse the registry and incorporate them in their triggers to build larger granules of knowledge. By transforming all types of rules and triggers into Web Services, and using the event notification service and the Web Service infrastructure, multifaceted knowledge of collaborating organizations can be uniformly processed without having to use multiple types of rule engines.

Private rules and triggers are private to the organization that defines them. They can be invoked by shared event(s) or private event(s) and the effects of their processing **will not** contribute to the event data being transmitted to other sites for security reasons. However, they are visible to local applications.

3.1. Distributed Events and Event Data

Distributed events defined by collaborating organizations are called organization-defined events. Some events are system-defined events such as the detection of a constraint violation, the deletion of a defined event, an exception condition of an operation, etc. Each event type has a set of parameters (i.e., attributes of data entities that form the initial event data). For each event type, authorized collaborating organizations can publish its occurrences. Since multiple sites may have rules applicable to the event, the Event Server of the event-originating site needs to know which sites to send the event data to. In our approach, the rule user interface at each site that defines rules would check the parameters of each event registered at the host site. If these parameters form a superset of the rule input data, the rule is considered as applicable and that site becomes an "implicit subscriber" of the event. The Event Server at the event-originating site will make asynchronous calls to the Event Servers of explicit subscribers and synchronous calls to Event Servers of implicit subscribers.

Another problem arises if an organization has subscribed to multiple event types and these events occur

concurrently. Different sets of event data associated with these occurrences will have to be combined for the processing of private rules for local decision support and shared rules for global decision support. This is because rules defined by an organization may involve data contained in different event data sets.

3.2. Distributed Triggers

A trigger specifies what alternative events will trigger the evaluation of a composite event expression (e.g., Event E1 occurred before E2 or E1 and E2 have occurred in a specified time window) and the invocation of a rule structure if the composite event expression is evaluated to *true*. Triggers are stored and processed by the local Rule Server at the site of definition. In our previous work [7, 8], we have developed a trigger language, which allows the specification of a linear, tree or lattice structure of condition-action rules. In this work, we extend it to allow the specification of rule structures that make references to all three types of rules. Our triggers are different from the triggers in active databases. Triggers in active databases are event-condition-action (ECA) rules, each of which has three components: E, C and A. In our work, events and CA-rules are separately defined entities. A trigger links any event or a number of alternative events to not only condition-action rules but also other types of rules defined independently by collaborating organizations. An event can be any incident of interest to collaborating organizations as stated in the introduction section, not just insert, delete, update and select operations in databases. The action part of our CA-rules is not limited to database operations. It can include Web Service calls and RMI calls to application programs.

A couple of issues related to triggers need to be addressed. First, a structure of different types of rules captures the multifaceted knowledge of an organization that can be shared and used to build more complex rule structures by other collaborating organizations. Can the rule structure make reference to both shared and private rules and/or rule structures? We believe that this should be allowed. However, for security reasons, if any private rules are involved, the derived data and/or updates of the old version should **not** become a part of the event data to be transmitted to other sites. Second, since events and rules are defined by different organizations, the terms used by different organizations to name data entities and attributes may have semantic discrepancies. Manually mapping terms used in event specifications to those in rule specifications would be rather tedious and insufficient. In this work, we are investigating the use of domain ontologies managed by an ontology management system (OMS) [11, 12] to either fully or semi-automatically deal with the ontological problem by reasoning on the underlying concepts of the terms used in event and rule specifications. The Web Services of shared rules and triggers published in the Web Service Registry are stored in the ontology database (ODB) and managed by OMS of subsystem 2. To improve the quality of service discovery, we are also making use of our previous work [13] on a constraint-based UDDI registry.

3.3. Distributed Rules and Rule Processing

Multiple event types can be defined by collaborating organizations and each event type can have multiple occurrences. Each occurrence can start multiple rounds of event notifications and rule invocations as the dynamic event data set evolves. The network system needs a way of tracking all the processing initiated by an event occurrence. In this work, we treat an event occurrence as the start of a distributed transaction. The site that initiates the event occurrence serves as the coordinator of the transaction and all other sites that have applicable rules and rule structures are the participants. Thus, the distributed transaction management techniques introduced in the database management field can be adapted for distributed rule processing after accounting for the following two differences. First, in our system, separate event data sets are maintained for different event occurrences. Thus, operations on these data sets by different transactions do not interfere with each other. Second, the traditional distributed database management systems transfer database operations to different sites to process the distributed data fragments. In our system, dynamic event data sets are transmitted to different sites to trigger the processing of distributed rules as well as operations invoked by rules.

Distributed rules may form a cycle that causes a non-termination problem. Our approach calls for the coordinator of an event occurrence to check if the same derived event data and/or updates to event data have been returned by the same group of implicit subscribers. This would indicate that distributed rules triggered by the event form a cycle. Organizations that collectively possess the cyclic rules should be immediately informed so that some rule can be removed or modified to break the cycle.

4. Related Research

Several distributed event notification/service systems such as DREAM [14], Herald [15], Yeast [16], provide features similar to our distributed event infrastructure [7, 8]. Of these, only DREAM and Yeast, couple event notification with rule processing, but use only one rule type – ECA. There are many rule-based systems [17] that process rules with a single rule representation. There are a few systems which process ECA rules on distributed data [18, 19].

There are three major differences between our system and the systems mentioned above. First, event notification is not linked with the processing of heterogeneous, distributed rules. Second, they deal with the processing of only one type of rules. Third, they decompose rules into sub-rules and transmit them to multiple sites for processing against distributed data, whereas we transmit event data to those sites that contain applicable rules and application operations.

5. Summary

In this paper, we have presented the motivation of our R&D effort on processing distributed, dynamic event data and multi-faceted knowledge in a collaboration federation environment. We used some rules and rule structures from the NPDN environment to show how they can be invoked by events to operate on the evolving event data in a distributed event infrastructure. The architecture of a distributed network system and the functions of its components have been presented. We also discussed several issues relating to distributed event, rule and trigger processing and presented our approaches to deal with them. The infrastructure and technologies introduced in this work allow collaborating but autonomous organizations to define, publish, apply, modify, and remove shared and private events, rules, triggers and application operations for supporting their decision making and problem solving.

6. References

[1] H. Garcia-Molina et al, "The TSIMMIS approach to Mediation: Data Models and Languages", *Journal of Intelligent Information Systems* 8(2), Kluwer, USA, 1997, pp. 117-132.

[2] A. Sheth, "Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics", in M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman, eds., *Interoperating Geographic Information Systems*, Kluwer, USA, 1998, pp. 5-30.

[3] J. F. Sowa, *Knowledge Representation: Logical, Philosophical and Computational Foundations,* Brooks Cole, USA, 2000.

[4] GUIDE Business Rule Project final report, 1997.

[5] I. Rouvellou et al, "Combining Different Business Rules Technologies: A Rationalization", *Proc. OOPSLA 2000 Workshop on Best-practices in Business Rule Design and Implementation*, Minnesota, USA, 2000.

[6] J. Tucker, "Historical Trends Related to Bioterrorism: An Empirical Analysis", Special Issue, *Emerging Infectious Diseases* 5(4), Centers for Disease Control and Prevention, USA, 1999, pp. 498 – 504.

[7] M. Lee, S. Y. W. Su, and H. Lam, "Event and Rule Services for Achieving a Web-based Knowledge Network", *Proc. International Conference on Web Intelligence*, Maebashi City, Japan, 2001, pp. 205-216.

[8] M. Lee, S. Y. W. Su and H. Lam, "A Web-based Knowledge Network for Supporting Emerging Internet Applications", *WWW Journal* 4(1/2), Kluwer, USA, 2001, pp. 121-140.

[9] S. Degwekar et al, "Application of An Event-Trigger-Rule System to Agricultural Homeland Security", *Proc. International Conference on Knowledge Sharing and Collaborative Engineering,* St. Thomas, US Virgin Islands, 2004, pp. 50 – 56.

[10] http://aquat1.ifas.ufl.edu/genspe.html

[11] H. W. Beck, and H. S. Pinto, "Overview of Approach, Methodologies, Standard and Tools for Ontologies", http://www.fao.org/agris/aos/Documents/BackgroundAOS.html, United Nations Food and Agricultural Organization, 2002.

[12] H. W. Beck "The Role of Ontologies in eLearning", To appear in *Educational Technology Magazine,* 2005. (In Press)

[13] S. Degwekar, S. Y. W. Su, and H. Lam, "Constraint Specification and Processing in Web Services Publication and Discovery", *Proc. International Conference on Web Services*, California, USA, 2004, pp. 210 – 217.

[14] A. Buchmann et al, "DREAM: Distributed Reliable Event-Based Application Management", *Web Dynamics Adapting to Change in Content, Size, Topology and Use*, Springer Verlag, (Eds: Mark Levene and Alexandra Poulovassilis), Germany, 2004, pp. 319-352.

[15] L. F. Cabrera, M. B. Jones, and M. Theimer, "Herald: Achieving a Global Event Notification Service", *Proc. Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Elmau, Germany, 2001.

[16] K. Balachander and D. Rosenblum, "Yeast: A General Purpose Event-Action System", *IEEE Transactions on Software Engineering* 21(10), IEEE Computer Society, USA, 1995, pp. 845-857.

[17] J. Widom and S. Ceri, *Active Database Systems, Triggers and Rules for Advanced Database Processing*, Morgan Kaufmann, USA, 1996.

[18] V. Kantere et al., "Coordinating Peer Databases Using ECA Rules", *Proc. International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, Berlin, Germany, 2003, pp. 108-122.

[19] T. Heimrich and G. Specht**,** "Enhancing ECA Rules for Distributed Active Database Systems", *Web, Web-Services, and Database Systems*, Springer Verlag, Germany, 2002, pp. 199-205.