Integrated Specification and Processing of Knowledge and Process for Achieving Knowledge Sharing among Collaborating Organizations

Xuelian Xiao, Jeff DePree, Seema Degwekar, Stanley Y.W. Su
DB Systems R&D Center
Department of Computer and Inform. Science and Engineering
University of Florida
Gainesville, FL USA
{xxiao,jdepree,spd,su}@cise.ufl.edu

Howard Beck
Agricultural and Biological Engineering Department
Institute of Food and Agricultural Science
University of Florida
Gainesville, FL USA
hwb@ufl.edu

Abstract

Today's government and business organizations face many complex problems and challenges. Effective resource sharing, collaboration and coordination among organizations are needed. Collaborating organizations need to share not only data, but also human and organizational knowledge embedded in organizational and inter-organizational policies, regulations, constraints, processes and operating procedures. This paper presents an XML-based knowledge and process specification language, a user interface tool, and a distributed Event-Triggered Knowledge Network (ETKnet). The language allows policies, regulations and constraints to be specified in terms of three types of rules and structures composed of these rules. The action clause of a condition-action rule allows a process or an operating procedure to be specified in the form of a structure of manual and automated operations having various constructs. The user-interface is meant to ease the user's task of defining events of common interest, along with rules and rule structures, and outputting the captured knowledge in the specification language. The rules and rule structures in the XML format are automatically translated into program code, wrapped as web services and installed at the defining organizations' sites. When an event occurs, data associated with the event is sent to the sites that contain applicable rules. ETKnet processes these distributed rules and rule structures uniformly as web services, thus achieving event-triggered knowledge sharing.

Keywords

Event-triggered Knowledge Sharing, Collaboration, Knowledge and Process Specification, Distributed Knowledge Rule Processing

1. Introduction

Government agencies of today are facing complex global problems such as border control, illegal immigration, terrorism and bio-security threats. Business organizations are also facing fierce competition and challenges in the global economy. An individual organization alone may not possess the information and know-how required to solve a complex problem. Therefore, effective resource sharing, collaboration and coordination among organizations hold the key for solving the problem successfully. One aspect of effective collaboration is to share not only data, but also human and organizational knowledge useful for decision support, problem solving and activity coordination. The technology of sharing distributed, heterogeneous data has been extensively studied, but an effective way of sharing knowledge among collaborating organizations is still lacking. Our aim in this work is to represent and share knowledge among collaborating organizations.

First, we are interested in capturing the multifaceted knowledge embedded in organizational as well as inter-organizational policies, regulations, constraints, processes and operating procedures. A common way of representing knowledge is to use knowledge rules [13]. Three types of knowledge rules have been found to be useful in many applications: integrity constraints [16], logic-based derivation rules [17], and action-oriented rules [18]. In our previous works [4, 6, 15], we have found that all three types of rules and structures composed of these rules are useful for specifying policies, regulations, constraints, processes and operating procedures in both e-business and e-government application domains. In this work, we have incorporated the control constructs used in workflow and business processes (WPDL [19] and BPEL [1]) in a high-level XML-based knowledge and process specification language to formally specify multifaceted knowledge for integrated processing as well as knowledge exchange. We have also implemented a user interface to ease organizations' task of defining such knowledge.

Secondly, we would like to investigate the technique and infrastructure for sharing distributed, heterogeneous data, multi-faceted knowledge and application operations in a uniform manner

when events of interest occur. Traditionally, different types of knowledge rules are defined by different rule languages and processed by different types of rule engines. Business processes and workflow processes are defined by different languages (e.g., [1, 19]) and processed by business process and workflow management systems. The installation and maintenance of several types of rule engines and a workflow/process management system for our purpose will be very costly. What is more, their internal processing and external interface are rather different. It is difficult to achieve their interoperation. In this work, we translate multi-faceted knowledge, which is defined by collaborating organizations using the user interface and formally represented by the specification language, into program code and wrap this code as web services for uniform processing and interoperation in an Event-triggered Knowledge Network (ETKnet [5, 6]). Thus, different types of rules and processes can interoperate seamlessly. For example, a logic-based derivation rule defined by one organization may deduce some data for use by an action-oriented rule defined by another organization. The latter may activate a collaborative operating procedure to update some data, which can then be verified by a constraint rule of yet another organization. These rules can also be defined by an organization in a rule structure. Collaborating organizations do not have to write application code to implement multi-faceted knowledge, or use multiple rule engines and workflow/process management systems to process them.

Our work is different from [2, 3] in that they use a single rule type (ECA rule) for workflow/business process modeling, and application integration. Interoperation of different types of rules is achieved in [12] by generating web service interfaces as the communication mechanism for rule engines that manage different types of rules. We, alternatively, start with knowledge and process specification and automatically generate web services for their uniform processing without having to use different types of rule engines and a workflow/process management system. The existing rule languages proposed or developed for business process and semantic web applications (e.g., [7, 8, 9, 11]) do not have the same expressive power as the language and the user interface presented in this paper because they do not unify and integrate knowledge specification (defined using three different types of rules and rule structures) with process specification (defined using a rich set of structural constructs).

The rest of the paper is organized as follows. Section 2 describes the knowledge and process specification language. Section 3 presents the user interface. In section 4, we explain how we process distributed knowledge rules and processes in ETKnet. Section 5 concludes the paper.

2. Knowledge and Process Specification Language

In this language, we integrate the specifications of different types of rules and rule structures and the structural constructs specified in WPDL and BPEL in a single XML-based language. We adopt some constructs from RuleML for derivation rules. The concept used for defining rule structures is based on our earlier work [10]. In our action-oriented rule, the action that is to be executed can be a single operation or a structure of operations composed by a set of operation constructs, which can be used to specify a collaborative process or operating procedure. Due to space limitations, we cannot describe the entire language. We give a brief description of each rule type and rule structure here. Interested readers can access the complete language specification at http://www.cise.ufl.edu/research/ETKnet/RuleBase.xsd.

2.1 Integrity Constraints

Integrity constraints model the constraints or conditions to which data should adhere, which are dictated by requirements of an application. Since we use an object-oriented data model for modeling data associated with an event occurrence (i.e., event data), constraints can be specified on an object's attributes and relationships between attributes. Our system supports two types of integrity constraints. Constraints of the first type are called *attribute constraints*. They limit the acceptable values that any data attribute may have at any point in time. Such a constraint is of the form

$$x\ \theta\ n, \text{ or } x\ (in\ /\ not\ in)\ \{\ n_1,n_2,\ldots,n_a\}$$

where x is an object attribute, n is a value from x's domain, θ is one of the six arithmetic comparison operators (>,>=,<,<=,=,≠), and $\{n_1,n_2,\ldots,n_a\}$ represents a set of enumerated values from x's domain.

Constraints of the second type are *inter-attribute* constraints. They are used to constrain relationships among attributes. The relationship can be modeled in either a mathematical way or conditional way. The former allows for so-called formula constraints, which are of the form

$$f_1(x_1,x_2, \ldots, x_b)\ \theta\ f_2(y_1, y_2, \ldots y_c)$$

where $f_1(x_1,x_2, \ldots, x_b)$ and $f_2(y_1, y_2, \ldots y_c)$ are mathematical formulas relating the object attributes $x_1,x_2, \ldots, x_b$, and $y_1, y_2, \ldots y_c$, respectively. The conditional constraints are of the form

$$\text{If } (P_1\ \alpha\ P_2\ \alpha \ldots \alpha\ P_d) \text{ then } (Q_1\ \alpha\ Q_2\ \alpha \ldots \alpha\ Q_e)$$

where $P_1\ \alpha\ P_2\ \alpha \ldots \alpha\ P_d$ and $Q_1\ \alpha\ Q_2\ \alpha \ldots \alpha\ Q_e$ are predicate expressions of the form $f_1(x_1,x_2, \ldots, x_b)\ \theta\ f_2(y_1, y_2, \ldots y_c)$ connected by the logical operator α in {AND,OR}. Each of $P_1, P_2, \ldots P_d$ and $Q_1, Q_2, \ldots, Q_e$ can be in its asserted or negated form. All of the attributes of entities referenced in a constraint rule are the rule's input data. The output of the rule is the truth value indicating whether the constraint is satisfied or not.

2.2 Derivation Rules

Derivation rules are also known as inference rules or deductive rules. They provide new data if some premises on existing data are satisfied. They are of the form

$$P \rightarrow Q, \text{ or } P \Rightarrow Q$$

which means that, given that the premise P evaluates to true, the conclusion Q is also true. In their general forms, P can be a set of predicates linked by logical operators AND and/or OR, and Q can be a set of predicates linked by the logical operator AND.

The attributes of entities referenced in P are the rule's input data and those referenced in Q are its output data.

2.3 Action-Oriented Rules

Action-oriented rules are generally expressed in the form of an event-condition-action (ECA rule) or just an event-action (EA rule). The general format of the rule is

On E, If C then execute A

which means that, when the event E occurs, action A is executed if condition C evaluates to true. In our work, we separate the event specification from CA specification to allow an event defined by an organization to trigger the processing of CA rule(s) defined by any other organization(s). We adopt the format of condition-action-alternative-action (i.e., If C then A else B) used in [10,14] for action-oriented rule specification. In this rule type, attributes referenced in C as well as the input to the conditions and operations specified in A and B form the rule's input data, and the result of performing the operations in A or B is the output data.

An action is either a primitive automated/manual operation, or a structure of operations that can be decomposed and specified as a set of operation constructs. The latter is used to specify a collaborative process or operating procedure showing the execution relationships among operations. By integrating the process patterns defined in WPDL and BPEL, we include the following constructs in our specification language. These constructs can be used recursively to express more complex operation structures.

2.3.1 Sequential construct

This construct is used to define a sequential execution order between two operations. An operation is activated after the completion of another one.

2.3.2 Switched construct

This construct specifies a conditional processing of operations. It is similar to the switch statement in a programming language. The conditions in branches are evaluated in the order in which they appear, and the first one whose condition is evaluated to true will be executed. If no conditional branch is taken, then a default branch is taken.

2.3.3 Unordered construct

The operations listed in this construct can be executed in an unspecified order, but not concurrently. Execution and completion of all operations is required.

2.3.4 Selective construct

This construct allows a number of operations to be selected randomly from a set of operations for execution in an unordered fashion. The exact number is specified in the NUM field in this construct.

2.3.5 Repeated construct

This construct is used for structured looping. The loop body can be a single operation or a set of structured operations.

2.3.6 Split construct

This construct is used to specify that several operations can be executed in parallel. There are three types of split: AND-Split, OR-Split, and XOR-Split. The AND-Split is used to specify that all successor operations can be executed in parallel after the completion of the predecessor. The OR-Split is used to state that a specified number of operations can start their execution in parallel

after the completion of the predecessor. The number is given in the NUM field of the construct. These operations have to satisfy the data conditions specified for their processing. The XOR-Split is very similar to the *switched construct*. The only difference is that, in the XOR-Split, there is no default branch. Thus, zero or one of the successor operations can be executed.

2.3.7 Join construct

This construct is used to specify a synchronization point. There are two types of Join constructs: AND-Join and OR-Join. In AND-Join, the successor operation can only be executed after the completion of all of its predecessors. In OR-Join, the successor operation can only be executed after the completion of a specified number of predecessors. The number here is given in the NUM field of the construct. The successor operation has to satisfy the data conditions specified on each chosen incoming transition.

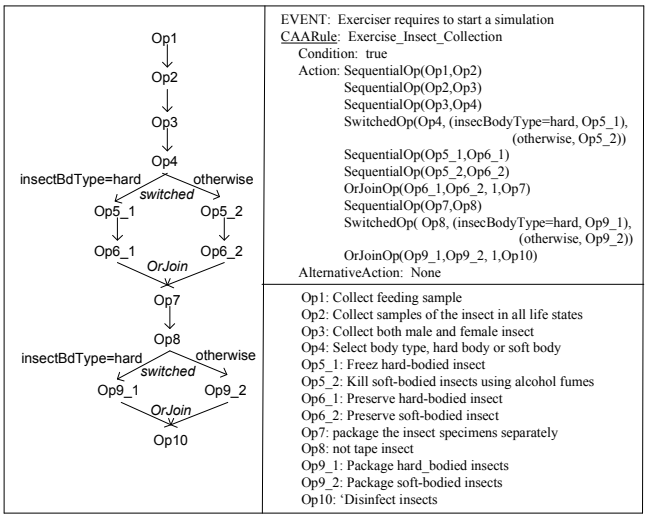

Figure 1: An example operation structure

2.3.8 An example of operation structure

One of the applications for testing our research outcome is the National Plant Diagnostic Network (NPDN, http://www.npdn.org). The goal of NPDN is to provide surveillance and rapid detection of disease and insect outbreaks that threaten crops. We show in Figure 1 an operation structure, which models the standard operating procedure defined in the NPDN environment for collecting an insect specimen. The structure is specified in the action clause of a CAA rule triggered by an event in a simulation exercise.

2.4 Rule Structure

In both e-government [6] and e-business [4] applications, we have found that it is useful to use a structure for organizing dissimilar rules to capture organizational knowledge. A rule structure specifies the structural relationships among rules. Since a rule is a higher level of specification than a manual or automated operation, the structure of rules can be simpler than the structure of low-level operation defined in the action clause of an action-oriented rule. The relationships captured in our rule structure include **link**, **split**, **and-join**, and **or-join** relationships. A **link** relationship is used to state that two rules should be executed in sequence. A **split** relationship is used to state that a set of rules can be executed in parallel. An **and-join** relationship between a collection of rules $r_1, r_2, \ldots, r_n$ and rule *s* states that rule s can be executed only after the completion of $r_1, r_2, \ldots, r_n$. An **or-join** relationship between a collection of rules $r_1, r_2, \ldots, r_n$ and rule *s* states that rule *s* can be executed after a specified number of rules in the collection complete theirs. The number here is specified in the construct.

2.5 Event and Trigger

An event is anything of interest to collaborating organizations that occurs at a particular point of time. Collaborating organizations may define any events of common interest and triggers to connect events to rules and rule structures. Each trigger explicitly specifies the rule(s) or rule structure(s) that need to be processed on the occurrence of a particular event. A trigger can also be automatically generated by the system to link an event to a rule or rule structure that is "applicable" to the event. By applicability, we mean that the set of objects and attributes referenced in the event data is a superset of those that form the input of the rule or rule structure. This means that the event data provides the data needed to process the rule or rule structure.

Collaborating organizations in a distributed, collaborative environment can define rules, rule structures, events and triggers independently. They can decide whether they would like to share what have been defined. Shared rules, rule structures and events are registered at a host site, and other organizations can use them to achieve inter-organizational sharing and collaboration (see Section 4 for details).

3. User Interface

In order to ease the tasks of users, who have not necessarily studied computer science, in defining events, rules, rule structures, triggers and operations, it is necessary to provide them with an easy-to-use interface which enables them to define the above elements through the manipulation of graphical widgets rather than through the use of a specification language or the writing of code. The interface, which can be instantly accessed via any web browser, provides an intuitive way to manage existing elements as well as define new ones. No knowledge of Java, XML, or our specification language is required of the user who is assisted by auto-complete boxes and an extensive help system. In the rest of this section, we will describe the central features of the interface.

3.1 Event and Knowledge Rule Specification

Each event is uniquely identified by its name. A description for each event can be provided during the definition. An event can be classified as local or global depending on whether an organization wants to treat the event data as private or shared respectively. Occurrence of a global event will cause the event data to be sent out to all subscribing organizations (i.e., explicit subscribers) and to those that have applicable rules (i.e., implicit subscribers). The event registration feature allows the user to select one or more global events and upload them to the host; this allows all organizations using the system to explicitly subscribe to these events and be notified of their occurrences.

Each rule has a name, a description and a state. The state of a rule can be active or suspended. Each rule belongs to one of the three rule types mentioned in Section 2. Depending on the rule type, the interface dynamically generates an HTML form to allow the user to enter the parameters specific to that type. Some rules may involve rather complicated expressions. The user interface provides the facility to define, display and hide components of these expressions. Defined rules are automatically translated into program code and wrapped as web services. The meta information for these web services is registered in a service registry at the host site to make them available for use by collaborating organizations.

The user interface also has the capability of defining a rule structure by enabling the user to create a diagram that specifies the relationships among rules. The diagramming feature gives the user the ability to select and place multiple rules, and then specify connections among them. A connection can either be sequential, parallel (i.e., split construct) or synchronized (i.e., join construct) as described in Section 2.

3.2 Process Specification

The operation definition screen is shown in Figure 2. Each operation has certain characteristics such as its name and input and output parameters. An operation can be automated or manual. An automated operation is made available to the system via a published web service; the URL of the WSDL document that describes the service is all that is needed during definition. The definition of a manual operation specifies the contact person(s), the email addresses and/or cell phone numbers, along with a message to instruct the persons on what operation to perform and how to inform the system after the operation has been performed.

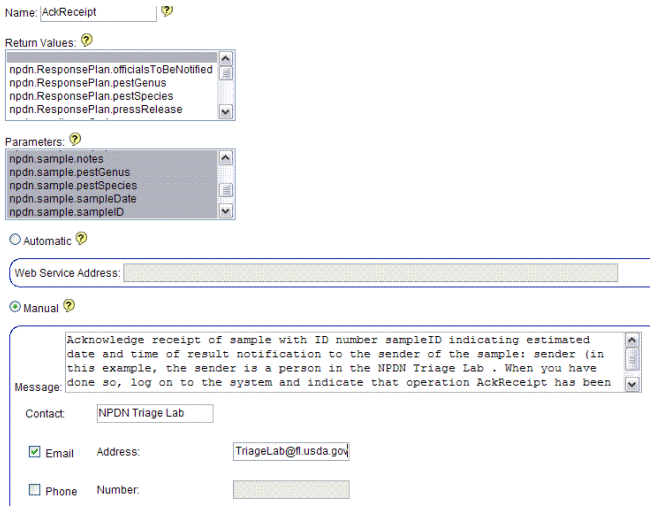

Figure 2: Operation definition screen.

The action definition screen is shown in Figure 3. This part of the interface is similar to that used to create rule structures. It allows the user to graphically add and connect operations in order to form arbitrarily complex operation structures. The user first selects from a list of previously defined operations, and places the selected items on the canvas. Connections can then be specified between any two operations by selecting each of them. Depending on the existing connections, each new link may result in any one of the seven constructs described in Section 2. By clicking on a connection, the user may specify a necessary condition for a given construct using the same interface that is used for the definition of

rules. The end product is an easy-to-follow diagram representing the flow of execution for an operation structure.

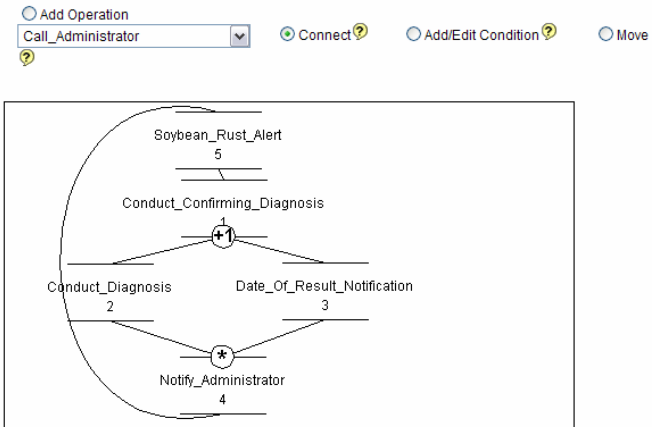

Figure 3: Action definition screen.

4. Processing Distributed Knowledge Rules and Processes

In this section, we explain how distributed events, rules (including processes), rule structures, operations and triggers are processed in our event-triggered knowledge network (ETKnet).

4.1 System Architecture

ETKnet has a peer-to-peer server structure (Figure 4). Each collaborating organization has a subsystem installed at its site, which is shown as CS (collaborating site) in the figure. The structure of all CSs is identical. One of the CSs is designated as the host site (HOST), which stores the meta-information of all shared events and web services that implement rules, rule structures and operations.

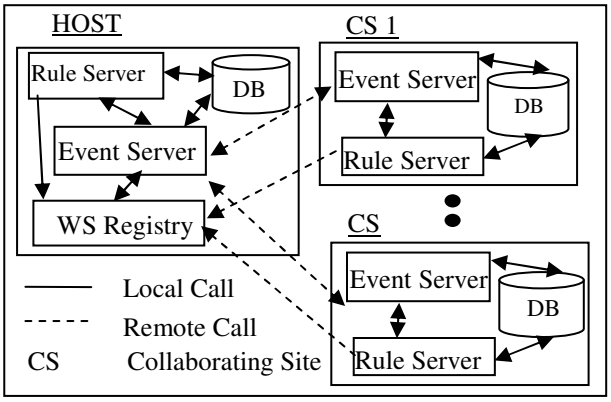

Figure 4. System Architecture

Each collaborating site creates and manages its own events, rules, rule structures, operations and triggers, and registers the shared ones at the host site. When an event occurs, information about explicit and implicit subscribers of the event is downloaded to the site of occurrence used by its Event Server for event notification and event data transmission. Web services generated for rules and rule structures are registered at the host but the rule code is stored at the site that defined them and processed by its Rule Server at run-time. Information about triggers is stored at the site that defines them and used by the Rule Server to process appropriate rules and rule structures upon the occurrence of an event.

4.2 Rules and Rule Structures as Web Service

In our specification language, the three types of rules have very different syntax and semantics. One possible approach to process these heterogeneous rules is to use multiple rule engines, each processing a specific type of rules, and build wrappers to convert the output of one rule engine to the input of another to achieve rule interoperability. However, the disadvantage of this approach is obvious. It is costly to install and maintain multiple rule systems at each site. What is more, we extend action-oriented rules to include complex structures to specify the collaborative processes or operating procedures among organizations. No single existing rule engine, business process management system or workflow management system has the power to parse and process integrated rule and process specifications.

Our approach can avoid these disadvantages. At each site, after the user defines a shared rule or rule structure using our user interface, it is translated into Java code, and wrapped as a web service. The web service is registered at the host site and the rule code is installed at the definition site. At runtime, all the distributed web services are processed in a web service infrastructure. By doing so, rules can now interoperate without using different types of rule engines. The interoperation of heterogeneous rules and rule structures can thus be achieved by invoking the *rule web services*. The algorithm used to convert rules and rule structures into *rule web services* has been given in [6].

The Java code for an action-oriented rule is more complicated than the derivation and constraint rules because the action clause can specify an operation structure that models a process or procedure. Each operation used in an action-oriented rule should have been deployed as a web service, which we shall call an *operation web service (OWS)*. An action-oriented rule is translated

into a program that activates a structure of OWSs based on its structural constructs.

A *sequential* construct requires the successor OWS to be invoked after the predecessor OWS. A *switched* construct is translated into a switch statement. The first OWS whose corresponding condition is satisfied will be invoked. In an unordered relationship, OWSs can be invoked in a random order but in sequence. A *selective* construct requires the invocation of a random number of OWSs in sequence. A *repeated* construct is translated into a structured loop to repeatedly invoke the OWSs in the loop according to their specified relationship. A *split* construct requires the creation of multiple threads to process either all or a given number of the indicated successor OWSs in parallel. The *join* construct requires the use of thread synchronization to ensure that either all or a given number of predecessor OWSs have been executed.

The web service for a rule structure can be generated in a similar manner. The only requirement is that all rules used in a rule structure should have been deployed as web services before, i.e. a rule structure can only refer to existing rules.

4.3 Distributed Event and Rule Processing

In this section, we explain how our system achieves the event-triggered processing of distributed rules, rule structures and operations. When an event occurs at a collaborating site, which becomes the coordinator for this particular knowledge sharing session, the event data associated with that event are sent to all explicit and implicit subscribers' sites in an XML document. Implicit subscribers contain applicable rules or rule structures. Event notification and event data transmission to all subscribers is carried out by the Event Server of the coordinator site. Event Servers at the subscribers' sites activate their corresponding Rule Servers to process their rules and/or rule structures based on the trigger information stored at these sites. The rules and rule structures at each site may add new data and/or modify the event data. They may also use the event data or part of it as input to process their operations without producing and modifying the event data. After finishing this round of rule or rule structure processing, those sites that have modified the event data would send the coordinator the updated files. The coordinator then merges the returned files to produce a new version of event data, which may cause some other rules and rules structures to become applicable. A new round of event data transmission and rule or rule structure processing would take place. Multiple rounds of event data transmission and rule processing can take place until no rule or rule structure is applicable. The last version of event data would contain all the data items pertaining to the event occurrence. All notified sites can use the data for further decision-making, problem solving and activity coordination.

We note here that, in the same knowledge session, a rule or rule structure, once processed, will not be activated again unless its input data specification refers to at least one attribute whose value has been updated in the last round of event and rule processing. Also, event data returned from different sites may contain conflicting or contradictory information, and distributed rule processing may get into an endless loop. These issues and solutions are presented in [5, 6].

5. Conclusion

The technology for integrated specification and processing of knowledge and process presented in this paper allows a number of collaborating organizations to define events of common interest and specify organizational and inter-organizational policies, regulations, data constraints, processes and operating procedures in terms of three general types of knowledge rules and rule structures. The action clause of a condition-action rule can specify an operation structure consisting of constructs useful for modeling a collaborative process or operating procedure. The defined events are used for event subscription, notification and event data transmission. The specified knowledge rules and rule structures are automatically translated into program code and uniformly processed as web services by ETKnet in a distributed fashion. This research outcome is important because: 1) high-level specifications of events and knowledge rules and processes are easier for collaborating organizations to understand, modify and use; thus organizations do not have to write application code to implement them, 2) the developed distributed event-trigger-rule processing technique and the network system allow different types of rules and complex processes/procedures specified in action-oriented rules and rule structures to be uniformly processed as web services using replicas of a single rule server without having to use multiple rule engines and workflow/process management systems, 3) through event notification, event data transmission and interoperation of distributed rules and rule structures, collaborating organizations can

"connect the dots (i.e., aggregated event data)", receive guided assistance on appropriate emergency response, and establish better communication, coordination and collaboration among them, and 4) the user-interface tool, the specification language and its translator, and the ETKnet developed in this project are web-based and written in the Java programming language; they can be installed on PCs that are connected to the Internet for broad participation of individuals from any group, in any geographic area, and of any organization.

Acknowledgements: This projected is supported by NSF grant no. IIS-0534065

6. References

[1]. BPEL,*"Business Process Execution Language"*, Accessed December 2007, http://www.oasis-open.org/specs/index.php

[2]. Bry, F., Eckert, M., Pătrânjan, P. and Romanenko, I. (2006) *"Realizing Business Processes with ECA Rules: Benefits, Challenges, Limits"* Proc. Int. Workshop on Principles and Practice of Semantic Web Reasoning, *pp. 48-62*

[3]. Chen, L., Li, M. L., and Cao, J. (2006) *"ECA Rule-Based Workflow Modeling and Implementation for Service Composition"* IEICE Transactions on Information and Systems, Volume E89-D, Issue 2, *pp. 624-630*

[4]. Degwekar, S. and Su, S. Y. W. (2006) *"Knowledge Sharing in a Collaborative Business Environment"* Proceedings of the Fifth Workshop on e-Business, Milwaukee, Wisconsin, (abstract on page 60 and paper on CD, 12 pages)

[5]. Degwekar, S. (2007) "*ETKnet: A Distributed Event- and Rule-based System for Knowledge sharing in a Collaborative Federation"*, Ph. D. Dissertation, Department of Computer and Information Science and Engineering, University of Florida.

[6]. Degwekar, S., DePree, J., Beck, H., Thomas, C.S., and Su, S. Y. W. (2007) *"Event-triggered Data and Knowledge Sharing among Collaborating Government Organizations"* Proceedings of the Conference on Digital Government Research, Philadelphia, PA., *pp.102-111*.

[7]. Gandon, F., Sheshagiri, M., Sadeh, N. (2004) *"Rule Language in OWL"* Carnegie Mellon University. Accessed December 2007, http://www.daml.org/2004/05/devday-rules/

[8]. Horrocks, I., et al. (2004) *"SWRL: A Semantic Web Rule Language Combining OWL and RuleML"* Accessed December 2007, http://www.w3.org/Submission/SWRL/

[9]. Hirtle, D., et al. (2006) *"Schema Specification of RuleML 0.91"* The Rule Markup Initiative. Accessed Jan 2007, http://www.ruleml.org

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

[11]. Cover Pages, (2001) *"Simple Rule Markup Language"*, Accessed Dec. 2007, http://xml.coverpages.org/srml.html

[12]. Rosenberg, F., and Dustdar, S. (2005) *"Design and Implementation of a Service-oriented Business Rule Broker",* Proceedings of the 7th International IEEE Conference on E-Commerce Technology Workshops, *pp. 55-63*

[13]. Sowa, J. (2000), *"Knowledge Representation: Logical, Philosophical and Computational Foundations"*, Pacific Grove, CA: Brooks Cole.

[14]. Su, S. Y. W., et al. (2005) *"Transnational Information Sharing, Event Notification, Rule Enforcement and Process Coordination"*. International Journal of Electronic Government Research, 1(2), April-June 2005, Ideagroup Publishing, *pp 1-26.*

[15]. Su, S. Y. W., Degwekar, S., Sardesai, A. and Beck, H. (2005*) "Processing Distributed Event Data and Multifaceted Knowledge in a Collaboration Federation"*, Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration (IEEE IRI-2005), Las Vegas, Nevada, *pp. 524-529*.

[16]. Ullman, J. (1982), *"Principles of Database System"*, 2nd Ed, Rockville, MD: Computer Science Press.

[17]. Ullman, J. (1988), *"Principles of Database and Knowledge-Base Systems"*, Rockville, MD: Computer Science Press.

[18]. Widom, J. and Ceri, S. (1996), "*Active Database Systems. Triggers and Rules for Advanced Database Processing*", San Mateo, CA: Morgan Kaufmann.

[19]. WfMC, (1998) *"Workflow Management Coalition Interface 1: Process Definition Interchange Process Model"*, Accessed Dec. 2007, http://www.wfmc.org/standards/docs/if19807m.pdf