Rule-Based Cooperation of Distributed EC Systems

Dongwoo Lee
Department of Computer Information Science
Woosong University, Daejeon, 300-718, Korea

ABSTRACT

Emergent requests or urgent information among enterprises require their intimate collaboration in B2B EC (electronic commerce). This paper analyzes the needs of intimate cooperation of distributed EC systems in terms of business contracts and presents an active rule-based methodology of close cooperation among EC systems and an active rule component to support it. Since the rule component provides high level rule patterns and event-based immediate processing, system administrators and programmers can easily program and maintain intimate cooperation of distributed EC systems independently to the application logic. The proposed active rule component facilitates HTTP protocol. Its prototype is implemented in B2B EC environment and evaluated using basic trigger facility of a commercial DBMS.

Keywords*: Distributed EC System, Rule-Based Cooperation, Active Rule Component, Distributed Processing.*

1. INTRODUCTION

The distributed EC systems need to be coordinated and integrated for enterprises to gain their common business goals. Especially emergent requests or urgent information among them require their intimate cooperation and should be processed in an immediate mode. For example, consider that a shopping mall becomes short of an item suddenly and requests a partner supplier to provide it. Then the supplier should provide the item quickly within a predefined time period according to business contracts between them. If, however, the item is out of stock in the supplier's warehouse and not able to be supplied within the predefined time period, it should be notified to the shopping mall promptly so that the shopping mall can try to find an alternate supplier to fill the item. Most current systems, however, due to the systems' security and autonomy, cannot handle these requirements appropriately, but handle them in a batch processing mode or *ad hoc* manners [1].

In this paper the needs of intimate cooperation of distributed EC systems are analyzed in terms of business contracts and an active rule component based on active database abstraction [2], [3] is proposed to provide the distributed EC systems with flexible coordination and immediate processing in WWW environment. Since high level rule programming is supported by the component, the close cooperation of distributed EC systems and event-based immediate processing can be implemented independently to application logic.

The active rule component is designed and integrated into distributed EC systems using HTTP protocol to be applied through firewalls. Thus the security and autonomy of distributed EC systems of individual enterprise are assured. Since most of business systems have database systems to store persistent data and commercial DBMSs provide basic trigger functionalities of active capability, the component is designed and implemented using a commercial DBMS for practical purpose.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the need for intimate cooperation of distributed EC systems, the proposed mechanism to meet it, and requirements to provide the mechanism. In section 4, one of the requirements, elements of an active rule program and examples of active rule programs are presented. Section 5 discusses architecture of the active rule component and implementation and evaluation of a pilot system is described. Finally in section 6, we conclude with future work.

2. RELATED WORK

The advent of the internet, WWW, and distributed computing technologies has been enabled business organizations to conduct business electronically. And a lot of researches on B2B E-Commerce have been carried on [4]. But the most of researches have been mainly focused on interoperability problems among distributed business systems. The issues of intimate cooperation among distributed EC systems have not been addressed comprehensively.

There are many researches on exception handling issues on business processes [5]-[7]. The exception is defined as deviation from the normal workflow, such as system errors or failures that interrupt normal processing of workflows. The

* *Corresponding author. E-mail : dwlee@wsu.ac.kr*

Manuscript received Aug. 10, 2009 ; accepted Aug. 22, 2009

exceptions are classified into basic failure on system level, application failure, expected exception on workflow level, and unexpected exception. Especially [5], [7] propose rule based exception handling methods. However, these researches are focused on normal processing of workflows with the exceptions, i.e., fault-tolerant workflow processing. That is different from intimate cooperation issues in this paper.

In summary, previous work addressed either general interoperability issues of businesses or exception handling of business processes' failures. Few of them comprehensively address close cooperation issues among EC systems in WWW.

3. INTIMATE COOPERATION AMONG DISTRIBUTED EC SYSTEMS

3.1 The Need for Intimate Cooperation

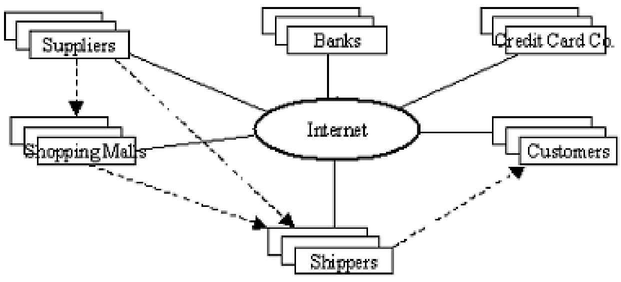

Fig. 1. Typical B2B E-Commerce

Consider typical B2B E-Commerce environment shown in Fig.1 as a motivating example. All participants are connected by solid lines which denote internet and information flow. The dotted arrows denote material flow among participants. Each participant does business with each other by its own B2B and B2C systems. That is, a shopping mall takes customers' orders and provides services by a B2C system, while it places orders to suppliers, requests delivery of items to shippers and money transfer to banks, inquires customers' credit to credit card company based on the agreement or contracts which were made with other participants. A supplier receives orders from shopping malls by its B2B system.

In Fig. 1 each participant has fire-wall for security and is connected each other via the fire-walls and internet. In general, the fire-walls close most of ports except special ports such as HTTP 80. Therefore the systems of each enterprise should use HTTP protocol and fixed ports to interact with other organizations' systems [8].

In the above B2B E-Commerce there are two kinds of job processing modes in enterprises. One is a batch-processing mode in which an enterprise collects jobs and processes them at a time. The other is an immediate processing mode in which an enterprise processes jobs promptly when they come in. In the former mode human or systems work efficiently while the processing time of each job becomes longer. In the latter mode human or systems should always wait for a job while each job is processed promptly. The choice of processing modes depends on the characteristics of jobs, policy of an enterprise, and contracts between enterprises.

Intimate cooperation among EC systems, i.e. immediate request - immediate cooperation, can be seen as exceptions out of enterprises' normal cooperation. That is, emergency request or critical information among EC systems should be transmitted to partners' systems promptly and processed by the systems in an immediate mode. They are not frequent, but once they occur they may require special treatment and affect customers' or enterprises' profits in a large degree. Since enterprises cooperate with each other by contract fulfillment, the cases for the intimate cooperation can be classified as following in terms of service contracts:

1. unable to fulfill a normal contract service
2. need to modify or compensate a normal contract service
3. need to cancel a normal contract service
4. need a special service instead of a normal contract service

For the above cases, new contracts, which require intimate cooperation, can be added into systems incrementally. Most current systems, however, due to the systems' security and autonomy, cannot handle these requirements appropriately, but handle them in a batch processing mode or *ad hoc* manners [1]. That is, they use login method by allowed users or Email. Or low-level *ad hoc* programs, which are coded into application logic, handle the cases. It causes software modularity problems.

3.2 Intimate Cooperation by Active Rule Paradigm

We derived the intimate cooperation procedure among distributed EC systems shown in Fig. 2, which consists of 4 phases:

1. Detection: a phase to detect that an enterprise wants to make emergency requests for partner's cooperation or critical information occurs, which should be transmitted promptly.
2. Transmission: a phase to transmit the detected situation to a partner promptly.
3. Evaluation: a phase to evaluate cooperation constraints whether they should be processed in an immediate mode. There are two kinds of constraints, time constraints and resource constraints.
4. Processing: a phase to execute or process the requested job or the information in an immediate mode.

As shown above, in order to cooperate in an immediate mode, the situation for intimate cooperation should be detected, notified or transmitted to each other, evaluated and recognized, and processed promptly. It shows that close cooperation among distributed EC systems is suitable application to active rule mechanism [2], [3]. That is, the intimate cooperation can be represented in active rules, such as occurrence of the situation for close cooperation as event, cooperation constraints as condition, and the processing of the job as action. Then, an active rule component, which processes active rules, detects automatically the occurrences of the event and notifies the occurrence to partner's system. The component of the partner evaluates the condition. If the condition is satisfied, then it executes the action promptly for intimate cooperation. That is, the cooperation among EC systems can be processed in an immediate mode without interference of application or users.

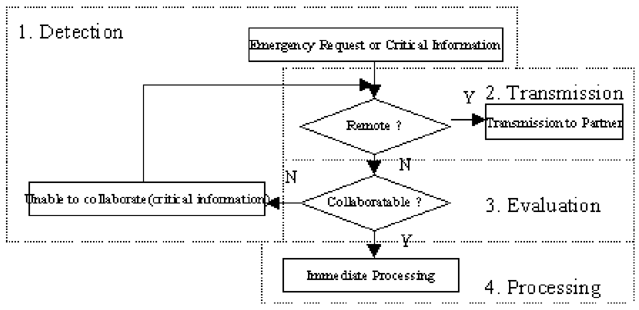

Fig. 2. Intimate Cooperation Procedure among Distributed EC Systems

The intimate cooperation among distributed EC systems can be supported by the following two main elements;

1. Active rule programming facility: an interface or programming facility which a system administrator or programmers can use to program each EC system to cooperate with others. That is, facility to represent event, condition, and action as a rule for close cooperation.
2. Active rule component: which includes capability to detect and manage events, evaluates condition, and executes action so that it executes and manages active rules. In addition, it utilizes communication infrastructure accessible through HTTP protocol for implementing cross-organizational interactions via fire-walls. In the next sections active rule programming and active rule component are described.

4. ACTIVE RULE PROGRAMMING FOR INTIMATE COOPERATION

In this research new active rule language is not developed. Instead an existent ECA (event condition action) rule language is extended at the minimum to express intimate cooperation. We adopt ECAA (Event Condition Action Alternative Action) rule language which is one of ECA rule languages [2], [3]. Fig. 3 depicts active rule structure. Alternative Action part is for flexible representation of close cooperation. Even though an enterprise makes an intimate cooperation request, its partner may not be able to cooperate because of the cooperation constraints. In this case the partner should notify the inability for cooperation to requester as an alternative action. Therefore it is optional.

```
Rule rule-name;
    Event event-expression;
    Condition condition-expression;
    Action Begin action-block End
    [Alternative Action Begin alternative-action-block End]
```

Fig. 3. Active Rule Structure

The rule-name does not have a major role in its execution since it is triggered by an event. Rule names are used mainly for management purpose.

Event : A rule is triggered by detection of occurrence of an event described in event-expression. The need for the intimate cooperation among EC systems is represented as an event. In this paper the events are classified into local and remote in terms of occurrence and subscription. If a local event occurs and is subscribed by a remote system, it is transmitted to the system. This information is registered into Event-Schema-Table of Event-Manager in an active rule component when it is defined. In terms of contents, the events are furthermore classified as request event and notification event as in [Table 1]. Since events for close cooperation are application-related, the wrapper codes are required to generate them and transfer to Event-Manager. The syntax of the event-expression is

event-expression ::= event-name(type1 par1, type2 par2, ...);

Table 1. Events and Actions for Intimate Cooperation

Request Events & Notification Events	Immediate Cooperative Action
Notify-Able-Service	No-Action
Notify-Unable-Service	Find-Alternate-Service
Request-Modify-Service	Modify-Service
Request-Cancel-Service	Cancel-Service
Request-Special-Service	Special-Service

Condition : Once an event has been detected, a condition part is evaluated. For close cooperation, cooperation constraints should be checked. Even though an EC system requests intimate cooperation to its partner system, the partner may not be able to cooperate because of cooperation constraints. There are two kinds of constraints. One is time constraints that the partner system should provide requested service within a predefined time period. Another is resource constraints that the partner should have resources such as man power, required system, or items to fulfill the requested service.

Action : If the condition is satisfied, the action block is invoked. An action block consists of a set of actions or call statements which execute the requested service for intimate cooperation. [Table 1] shows kinds of requested events and notification events and corresponding action types. Some actions may generate events again.

Alternative Action : If the condition is not satisfied, which means the partner system cannot cooperate immediately, the alternative action block is invoked. It is used to notify the inability of cooperation to the requester or remedy it.

Example of active rule programs : Consider the previous example that a shopping mall becomes short of an item suddenly and requests a partner supplier to provide it. Then the supplier should provide the item quickly within a predefined time period. If, however, the item is out of stock in the supplier's warehouse and not able to be supplied within the time period, it should be notified to the shopping mall promptly so that the shopping mall can try to find an alternate supplier to fill the item.

This example shows that a shopping mall and a supplier should cooperate in an immediate mode. It can be implemented by the following two rules ;

```
Rule Find-Alternate-Service /* rule on a Shopping Mall */
    Event unable-special-supply(string supplier, string item-1,
            integer n);
    Condition true;
    Action Begin find-alternate-service(string item-1, integer n)
             End

Rule Request-Special-Service /* rule on a Supplier */
    Event request-special-supply(string requester, string item-1,
            integer n);
    Condition no. of item-1 > n;
    Action Begin special-order-processing(string requester,
             string item-2) End
    Alternative Action Begin raise-event('notify-unable-special-
                          supply') End
```

5. ACTIVE RULE COMPONENT

5.1 Architecture of Active Rule Component

The intimate cooperation written in active rules is processed by an active rule component. The architecture of the component consists of five modules, Communication-Manager, Event-Manager, Rule-Manager, Event-Rule-Interface, and Actions/Applications. The overall architecture is shown in Fig. 4. In the following we describe each module with its structure as a figure.

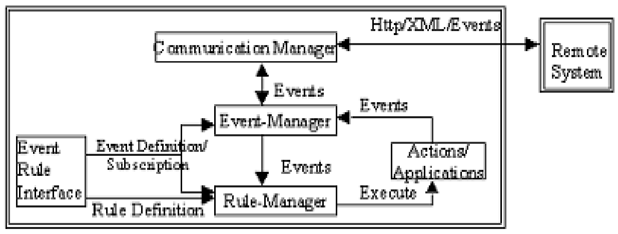

Fig.4. Overall Architecture of Active Rule Component

Communication-Manager: Using HTTP protocol the Communication-Manager sends and receives event messages for intimate cooperation in the form of XML with Communication-Manager of partner's active rule component. It is implemented in Java servlet[9] of a web server and contains two roles. Firstly it receives XML messages through a web server, extracts events, and transfer to local Event-Manager. Secondly it receives event from local Event-Manager, transforms into XML format, and using HTTP post command transmits to the Communication-Manager of partner's system[10]. Send_Event and Raise_Event in Fig. 5 take the two roles respectively.

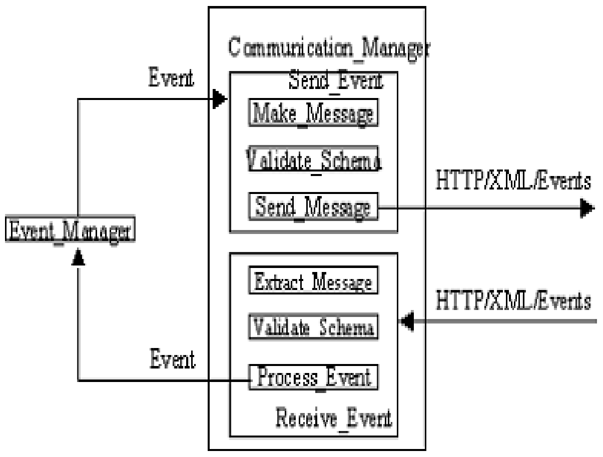

Fig. 5. Communication Manager

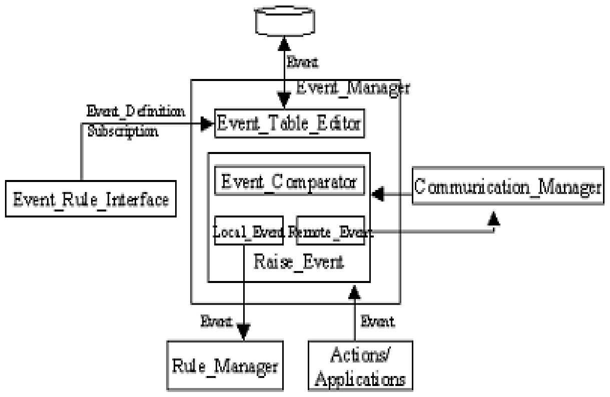

Fig. 6. Event Manager

Event-Manager : It manages schema definition of events and their subscription, identifies whether subscription of an event is local or remote, and transfers the identified event to corresponding Rule-Manager which subscribes it. It is implemented in Java servlet of a web server. Events are defined and registered for subscription by system administrators or programmers via Event-Rule-Interface. Then the definition and subscription are recorded into Event-Schema-Table by Event-Manager. At run time the Event-Schema-Table is referenced by Event-Manager to check whether an event is local or remote. The Event-Schema-Table consists of four tables to store all information related to events. Four tables are;

event-schema=(event-name, no-of-parameters)
publisher-schema=(event-name, publisher)
subscriber-schema=(event-name, subscribers)
parameter-schema=(event-name, para-name, type, position)

Since Event-Schema-Tables are stored in database, Event-Manager uses JDBC to connect database for storing and retrieving events. The structure of Event-Schema-Table is shown in Fig. 6.

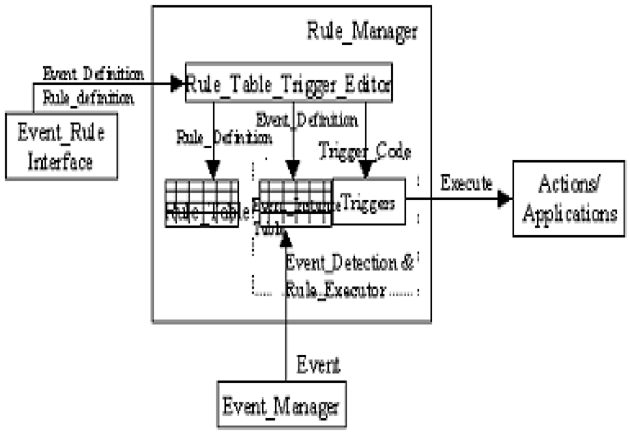

Fig. 7. Rule Manager

Rule-Manager : The Rule-Manager evaluates and executes active rules. It is implemented with basic trigger facilities of the underlined DBMS and contains Event-Instance-Tables, a Rule-Table, and triggers on the Event-Instance-Tables. An active rule for intimate cooperation is defined via Event-Rule-Interface and recorded into the Rule-Table. When an event is defined, its Instance-Table is created together.

A rule for close cooperation is written using triggers on the Instance-Table. For example, the rule 'find-alternate-supplier' can be written as a trigger on the Instance-Table 'unable-special-supply-instance-table' in the syntax of Oracle 9i[11];

```
create trigger find-alternate-supplier after insert on
            unable-special-supply-instance-table
begin
            if condition is true
            then call find-alternate-supplier();
end;
```

The structure of Rule-Manager is shown in Fig. 7.

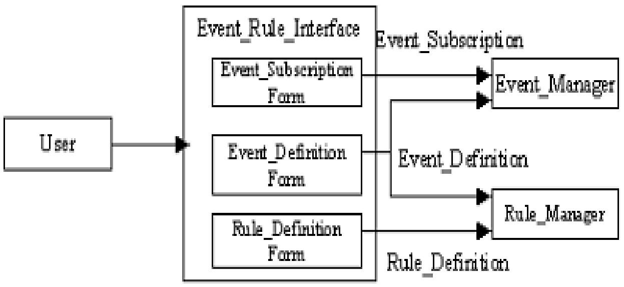

Fig. 8. Event Rule Interface

Event-Rule-Interface : The Event-Rule-Interface is an interface for system administrator or programmers to define events and rules as well as to manage them, *i.e.*, search, delete, and update. It utilizes facilities of Event-Manager and Rule-Manager, that is, Event-Table-Editor and Rule-Table-Trigger-Editor, respectively. It is shown in Fig. 8.

Actions/Applications : Actions are the treatments of requested services. Events are generated by applications. And some actions may generate events again. The actions/applications are internal if written in underlined DBMS's API or external if not. Since events for intimate cooperation are related to applications, wrapper codes to generate their occurrences and function calls to notify to Event-Manager are needed. Since the action part of an active rule is to treat the intimate cooperation, it consists of procedure or call statement for stored procedure, which provides the cooperation. The general structure of Actions/Applications is shown in Fig. 9.

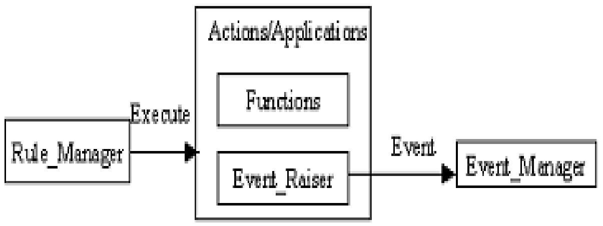

Fig. 9. Actions/Applications

5.2 Interactions among component modules

In order to explain how the active rule component processes the active rules for intimate cooperation, the interaction procedure among modules is described at build time and run time in the following, respectively.

Build Time

1. System administrators or programmers define events for intimate cooperation via Event-Rule-Interface.
2. The definitions of events are recorded into Event-Schema-Table by Event-Manager. System programmers write wrapper codes for occurrences of the events.
3. The definition of an event is transferred to Rule-Manager and its Instance Table is created.
4. Subscription of the event is registered via Event-Rule-Interface by the partner's programmer. The Subscription is stored into Event-Schema-Table by Event-Manager.
5. System administrators or programmers define an active rule on the event via the interface. The definition of a rule is recorded into Rule-Table by Rule-Manager.
6. System programmer writes triggers on the related Event-Instance-Table of the defined active rule and codes for the action part.

Run Time

1. A local event occurrence generated by a local application or action is notified, or a remote event is transmitted from a remote system via Communication- Manager.
2. Event-Manager identifies its subscription from Event-Schema-Table. If its subscription is local, then it is transferred to the local Rule-Manager. Otherwise, that is, remote subscription, it is transferred to Communication-Manager to transmit to a corresponding remote system.
3. Rule-Manager inserts the event instance into its Instance-Table and invokes a trigger to execute corresponding rule.
4. The trigger executes corresponding action for intimate cooperation.

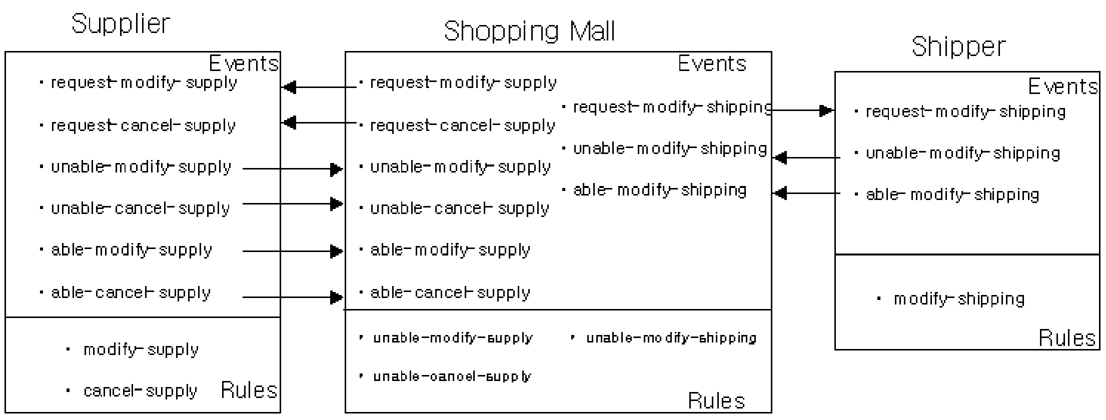

Fig. 10. Events and Rules for Intimate Cooperation

5.3 Implementation and Evaluation of a Pilot System

To validate the intimate cooperation mechanism and applicability of an active rule component, a pilot system has been implemented and applied to a typical B2B E-Commerce scenario. During design and implementation of the system, we considered practicability, interoperability with database, and platform independence. Therefore Java as an implementation language, Commercial DBMS Oracle 9i, and Apache Tomcat Web server were chosen.

Instead of implementing all features of an active rule component from scratch we utilized basic trigger facilities of a commercial DBMS. Major functions of the component are detection of event, evaluation of condition, and execution of action. In this paper, as described in the previous section 5.1 the condition evaluation and the action execution parts are designed and implemented by triggers of a underlined DBMS. That is, Event-Instance-Table for each event is created in DB and trigger codes on this table are written. The condition evaluation and action parts are written in trigger body. Thus, when an event instance is inserted into its Instance-Table, it invokes the related trigger that evaluates the condition and executes action part in the body.

For the application scenario, an internet shopping mall with its suppliers and shippers is used as shown in Fig. 1. Consider previous shopping mall. It decided to provide flexible service that customers are allowed to modify(add or delete items) or cancel their orders, and modify the delivery address within a predefined time period. In order to provide these services the shopping mall should cooperate with suppliers and shippers. Furthermore, since these cooperation affect the normal jobs processed, being processed, and going to be processed, they should be processed in an immediate mode. The events and rules for intimate cooperation among shopping mall, suppliers, and shippers for the scenario are shown in Fig. 10.

The pilot systems are implemented with Windows 2000 server, and Linux server, respectively. The implemented systems have been tested to verify their correctness for all cases. It should be noted that the intimate cooperation mechanism is intended for emergency and asynchronous close cooperation. Thus, it is complementary to workflow management system (WFMS) or business process management system(BPMS) which is intended for regular synchronous job process[12]. Since the proposed active rule component is loosely coupled with other subsystems, it can be easily applied to other systems which need active functionality.

6. CONCLUSION

In this paper an active rule component based on active database abstraction is proposed to support intimate cooperation among distributed EC systems in WWW environment. Since high level active rule programming is supported by the component, the cooperation among business systems and event-based immediate processing can be implemented independently to application logic. Thus, system administrators and programmers can easily program and maintain intimate cooperation among EC systems in WWW.

The active rule component is designed and integrated into business systems using HTTP protocol to be applied through firewalls. The security and autonomy of systems in individual enterprise are assured. Since most of business systems have commercial database systems to store persistent data and commercial DBMSs provide basic trigger functionalities of active capability, the active rule component is implemented to be practical using a commercial DBMS, Oracle. Even though the syntaxes of triggers of commercial DBMSs are different from each other, since the semantics of the triggers are similar, the proposed component can be implemented in other commercial DBMSs such as DB2 and MS SQL server.

In order to extend our research, future work includes support for various phases in business systems' life cycle[13] and extension of cooperation concept to support client oriented active rules. In addition, research on security of event transmission and action execution is needed, since notification of an event leads action execution directly. It may affect applications and database. Thus appropriate security method is required.

REFERENCES

[1] Nobuyuki Kanaya, et. al., "Distributed Workflow Management Systems for Electronic Commerce", *proceedings of 4th International Enterprise Distributed*

Object Computing Conference(EDOC'00), IEEE 2000.

[2] Norman W. Paton and Oscar Diaz, "Active Database Systems", Computing Surveys, ACM, 1999.

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

[4] Brahim Medjahed, et al., "Business-to-business interactions: issues and enabling technologies", *VLDB Journal*, Springer-Verlag, April, 2003. pp.59-85.

[5] Fabio Casati, S. Ceri, S. Paraboschi, and G. Pozzi, "Specification and Implementation of Exceptions in Workflow Management Systems", *ACM Tr. on Database Systems, Vol. 24, No. 3*, September 1999, pp.405-451.

[6] Zongwei Luo, Amit Sheth, Krys Kochut, and Budak Arpinar, *Exception Handling for Conflict Resolution in Cross-Organizational Workflows,* Technical Report, LSDIS Lab, Computer Science, University of Georgia, April 10, 2002.

[7] J. Meng, Stanley Y.W. Su, herman Lam and A. Helal, "Achieving Dynamic Inter-Organizational Workflow management by Integrating Business processes, Events and Rules", *IEEE HICSS-35'02*, 2002.

[8] W3C Recommendation *SOAP:The Simple Object Access Protocol,*http://www.w3c.org/TR/soap/, visited December, 2008.

[9] Danny Coward, *Java Servlet Specification version 2.3,* Sun Java Technology, Release 8/31/01.

[10] R. Fielding, et al., *Hypertext Transfer Protocol -- HTTP/1.1,* W3C, 1999.

[11] Oracle9i SQL Reference Release 2 (9.2) Part Number A96540-02, October 2002.

[12] Umeshwar Dayal, Meichun Hsu, and Rivka Ladin, "Business Process Coordination: State of the Art, Trends, and Open Issues", *Proceedings of the 27th VLDB Conference*, Roma, Italy, 2001.

[13] David Trastour, Claudio Bartolini, Chris Preist, "Semantic Web Support for the Business-to-Business E-Commerce Lifecycle", *proceedings of www2002*, 2002.

Dongwoo Lee
He received the B.S. and M.S in electronic engineering and Ph.D. in computer science from Korea university, Korea. Since 1995, he has been with the department of computer information science, Woosong university, Korea. His research interests include distributed processing and systems, database, and reactive systems.