[Position Paper] WS-ECA: An ECA Rule Description Language for Ubiquitous Services Computing

Jae-Yoon Jung
ASRI
Seoul National University
Seoul, 151-742, Korea
+31-6-1953-8025
jjyjung@gmail.com

Seung-Kyun Han
Digital Interactions Lab.
Seoul National University
Seoul, 151-742, Korea
+82-2-882-0504
jackleg83@gmail.com

Jonghun Park*
Digital Interactions Lab.
Seoul National University
Seoul, 151-742, Korea
+82-2-880-7174
jonghun@snu.ac.kr

Kang Chan Lee
Protocol Eng. Center
ETRI
Dajeon, 305-700, Korea
+82-42-860-6659
chan@etri.re.kr

ABSTRACT
Ubiquitous computing network comprises a variety of distributed service devices. Today Web services technology enables the heterogeneous devices to provide their own services and interact with each other via well-defined Internet protocol. Nevertheless, service devices in ubiquitous environments require more event-driven, autonomous interaction beyond rather passive service-oriented architecture of the present time. This paper presents an ECA (Event-Condition-Action) rule description language in an attempt to support capability for autonomous interactions among service-oriented devices in ubiquitous computing network. Specifically, the proposed WS-ECA is an XML-based ECA rule description language for web service-enabled devices. The rules are embedded in distributed devices which invoke appropriate services in the network if the rules are triggered by some internal or external events. The presented ECA-based device coordination approach is expected to facilitate seamless inter-operation among the web service-enabled devices in the emerging ubiquitous computing environment.

Categories and Subject Descriptors
C.2.4 [**Distributed Systems**]: Distributed applications and Network operating systems; D.3.2 [**Language Classifications**]: Specialized application languages.

General Terms
Management, Design, Standardization, Languages.

Keywords
Web service, ECA rule, ubiquitous devices, action constraints.

1. INTRODUCTION
Ubiquitous computing environments are becoming more heterogeneous and service rich domains [8]. Devices with particular services are interconnected each other via various types of networks. While web services technology has become a defacto standard for business application integration [9, 14], it is also rapidly emerging as an effective means for achieving inter-operability in ubiquitous computing environments [5, 12].

In this paper, ECA (Event-Condition-Action) rule description language is introduced for event-driven coordination of web service-enabled devices. The ECA rule was originally devised to provide traditional database systems with the capability of event-driven, instantaneous response. Due to its advantages of distinct and comprehensible rule description, it has been spread into a variety of domains and applications, including expert system [3], agent collaboration [10], policy-based control and management [4, 11], and middleware [6, 7].

Even in the field of distributed computing, event-based rule management has been actively applied to coordinate and control distributed systems. A representative work is the policy-based control and management presented in [4, 11]. They proposed the policy description languages (PDL) for a centralized server to control the distributed system. In particular, Shankar [13] defined ECA-P with post-conditions of devices and addressed weak points of previous ECA rules. Nevertheless, the centralized rule execution and control has made it difficult to apply such a result to ubiquitous computing environment, mainly due to the issues of communication overhead among devices, service encapsulation, and private information management.

Furthermore, there are several previous research results on event-based service computing. The SCXML (State Chart XML) [1] is an XML-based language that can define a control mechanism for distributed finite state machines by specifying their state transitions via external events. Yet, it focused on an individual device instead of a system of devices. On the other hand, WS-Eventing [2] defines a framework of effective event exchanges between web services. Especially, the event messages are delivered by use of a publish/subscribe mechanism, and the event content in a message can be described without restrictions for a specific application. Accordingly, this proposal can be used as a protocol to implement event-driven architectures based on web services, and it is adopted as a base protocol for our research.

This position paper proposes an Event-Condition-Action rule description language for Web Services (named WS-ECA) in order to support reactive interactions among service-oriented, heterogeneous devices in ubiquitous environments. The proposed language based on the ECA rules which are embedded in distributed service devices and then are triggered by events from internal and external devices. When the triggered rule satisfies some condition, it will be activated to invoke appropriate services.

In the following, Section 2 propose the proposed framework for ECA rule-based coordination of service devices, and Section 3 presents the structure and elements of the WS-ECA proposed in this paper. Finally, Section 4 concludes the paper.

2. ECA RULE BASED MANAGEMENT IN UBIQUITOUS SERVICE COMPUTING
ECA rules introduced in this paper enable the service devices to activate each other via event-based interaction. The proposed

* Corresponding author

WWW 2006, May 22–26, 2006, Edinburgh, UK.

ECA rules have been designed so that they can satisfy the following requirements that are necessary for effective coordination of systems of ubiquitous service devices.

- Conditional response & event filtering: The rules can be activated by appointed events and designated conditions according to user preference.
- Event passing: Events satisfied by specific conditions can be forwarded to another specific device, broadcast to multiple devices, or multicast to predefined devices.
- Temporal reaction: For the same event type, different actions can be performed according to their occurrence times.
- Logical expression in rules: Rule schema can support logical expression in rule definition such as conjunction, disjunctions, and negation.
- Rule chaining: Complex rules can be decomposed and expressed by several simple rules.

In ubiquitous computing environments considered in this paper, service devices are assumed to be interacting via the events registered through a publish/subscribe mechanism. Specifically, the events, conditions, and actions that constitute the ECA rules are defined as follows: Events are notification messages from internal or external service device. Conditions are the boolean expression that must be satisfied in a device for some actions to occur. Finally, actions are the instructions that provide active functionality for service devices, which includes service invocation and event generation.

Figure 1 shows the components of the proposed WS-ECA framework. An WS-ECA has three basic components, namely event, condition, and action. In addition, the schema of the rule description documents defines variables to facilitate condition evaluation based on event messages and system states.

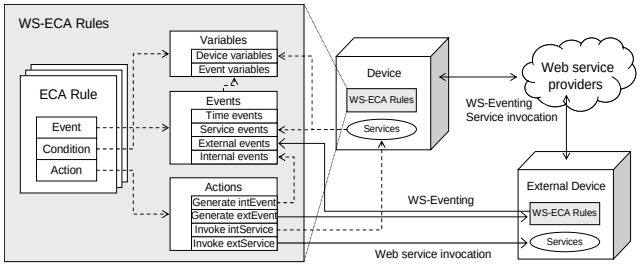

Figure 1. The basic structure of WS-ECA rules.

3. WS-ECA: ECA RULE DESCRIPTION LANGUAGE

This section describes the schema of WS-ECA in detail. The proposed schema supports the primitive events and actions as well as the composite ones for distributed ECA rule processing. First, Figure 2 shows the basic structure of WS-ECA in terms of XML.

```
<ECARule name="xs:NCName" targetNampespace="xs:anyURI"
        xmlns="http://di.snu.ac.kr/2005/eca/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" >
        <variables>? <variable ... />+ </variables>
        <events> event+ </events>
        <actions> action+ </actions>
        <rules>
                <rule>+
                        <event name="xs:QName"/>
                        <condition expression="XPath Expression"/>
                        <action name="xs:QName"/>
                </rule>
        </rules>
</ECARule>
```

Figure 2. Basic structure of WS-ECA.

3.1 Event

```
<events>
      <timeEvent type="once" name="xs:NCName">
            xs:dateTime </timeEvent>
      <timeEvent type="periodic" name="xs:NCName"
            unit="xs:duration" issued="xs:dateTime"?
            expired="xs:dateTime"?>xs:dateTime </timeEvent>
      <timeEvent type="relative" name="xs:NCName"
            baseEvent="xs:NCName" interval="xs:duration"/>
      <intEvent name="xs:NCName"/>
      <extEvent name="xs:NCName" eventID="xs:anyURI"/>
      <svcEvent type="before" name="xs:NCName"
            service="xs:QName"/>
      <svcEvent type="after" name="xs:NCName"
            service="xs:QName"/>
      <compositeEvent type="OR" name="xs:NCName">
  event+ </compositeEvent>
      <compositeEvent type="AND" name="xs:NCName"
            TTL="xs:duration"> event+ </event>
      <compositeEvent type="SEQ" name="xs:NCName"
            TTL="xs:duration"> event+ </event>
      <compositeEvent type="NOT" name="xs:NCName">
  event </compositeEvent>
</events>
```

Figure 3. Event schema of WS-ECA.

An event is the incident that triggers a rule. The events that can trigger a rule are called primitive events, and they include the following four types:

- **time events:** The event is generated by a timer at some specific point of time. Time events have three types of events: *absolute*, *periodic*, and *relative*. The event of *absolute* type is generated once at some time point, the event of *periodic* type occurs periodically, and finally the event of *relative* type is specified relative to some specific event by use of 'before' or 'after' operator.
- **internal events:** The event is generated by the internal system components including the rule engine and the device. It can be used to recognize the state change of a device or to trigger other rules.
- **external events:** The event is generated from a publishing device and is transmitted to subscriber devices through WS-Eventing.
- **service events:** The event can be one of two types: *before* and *after*. Specifically, the *before* (*after*, respectively) type is generated before (after, respectively) the specific service of the device begins (finishes, respectively).

More than one of these four primitive events may constitute composite events with the following logical operators.

- **disjunction (e1|e2|..|*en*):** The composite event of the type "*OR*" has more than one sub-events. One or more of the sub-events must occur within its specific time interval.
- **conjunction (e1&e2&...&*en*):** The composite event of the ty

pe "*AND*" has more than one sub-events. All of the sub-even ts must occur once or more times within its specific time interval.

- **sequence (e1,e2,...,e*n*)**: The composite event of the type "*SE Q*" has more than one sub-events. All of the sub-events must ever occur sequentially within its specific time interval.
- **negation (~e):** The composite event of the type "*NOT*" has only one sub-event. The sub-events must not occur within its specified time interval.

3.2 Condition

The condition part of ECA rules is a boolean statement that must be satisfied to in order to activate a rule. In WS-ECA, the condition statement is described in terms of an XPath expression (Clark, 1999). The expressions in the condition can use the values excerpted from the event part of a rule through the use of extension functions of XPath's built-in functions as shown in Table 1, or the variables defined in a rule document.

Table 1. Extension functions to XPath's built-in functions

Functions	return type	return value
eca:getVariable(event QName, path PathExpr)	xs:any	Specific values from event variables
eca:getDateTime(event QName)	xs:dateTime	Date and time information of the system

Variables can be aliases for specific elements of an events in the rule (called event variables) or user-defined states of a system (called device variables). The two types of variables are used to express the conditions conveniently or to assign input data for service invocation in the actions. The syntax for the variables is presented in Figure 4.

```
<variables>?
  <variable name="xs:NCName" systemVar="xs:QName"?
   eventVar="eca:getVariable(event QName, path PathExpr)"? />+
</variables>
```

Figure 4. Variable schema of WS-ECA.

3.3 Action

The action part of the rule is the instruction that must be executed by an internal or external device when a triggered rule is activated. In the proposed framework, the allowed types of actions can be one of the following:

- **invokeService (service)** : The action invokes a n internal or external service.
- **createExtEvent (event)** : The action creates an external eve nt and publishes it to the subscribing devices.
- **createIntEvent (event)** : The action creates an internal event and triggers other rules of the same device.

The action parts of ECA rule may consist of above primitive actions or their composite actions. A composite action is composed of more than one primitive or composite action with two basic operators: conjunctive and disjunctive operators. We do not consider sequential actions since they can be defined by use of series of ECA rules. Figure 5 shows the proposed schema for the action element.

- **conjunction (a1&a2&...&a*n*)**: The action requests to execut e all of its sub-actions. It contains a *transaction* attribute that indicates whether or not the rule engine should guarantee atomicity of all the sub-action. The attribute is a boolean variables of which the default value is 'false'.
- **disjunction (a1|a2|...|a*n*):** The action requests to execute one of its sub-actions. It contains a *sequence* attribute that indica tes whether the rule engine should execute the sub-actions in order. The attribute is of boolean type and its default value is 'true'.

```
<actions>
    <invokeService name="xs:NCName" service="QName">
        xs:any </invokeService>
    <createIntEvent name="xs:NCName" intEvent="xs:NCName">
        xs:any </createIntEvent>
    <createExtEvent name="xs:NCName" extEvent="xs:anyURI">
        xs:any </createExtEvent>
    <compositeAction name="xs:NCName" operator="AND"
        transaction="xs:anyURI"> action+ </compositeAction>
    <compositeAction name="xs:NCName" operator="OR">
action+ </compositeAction>
</actions>
```

Figure 5. Action schema of WS-ECA.

4. CONCLUSIONS AND DISCUSSION

This paper proposed the schema of event-based rule description language for the purpose of effective coordination of web service-enabled devices in ubiquitous computing environment. The proposed WS-ECA rules enable the service devices to interact with each other via WS-Eventing. WS-ECA is a stateless and light service description language that can support instantaneous activation upon a WS-Eenting message. It has advantages that users can describe required interaction among the devices in ubiquitous computing environment where multiple devices exchange their events based on publish/subscribe mechanism.

WS-ECA rules for individual devices may have service discrepancies with each other or cause undesirable situations when they are executed concurrently, since they can be created and processed independently. Motivated by this, we are now developing conflict detection and resolution algorithms for distributed ECA rule processing in order to prevent the situation. A rule conflict may occur when several rules triggered by an event are ready to execute simultaneously possibly conflicting service actions. The conflicts of ECA rules in distributed devices can be categorized into static conflicts and dynamic conflicts depending on whether the conflict is resolved in design time or run time. When a new rule is evaluated to contain any conflict with other rules in design time, it is said to be in a static conflict, and it cannot be registered and must be modified by users. On the other hand, if it is judged to contain any potential dynamic conflict with others in run time, additional resolution rules must be supplemented in order to register the rule in the system. Afterwards, if a dynamic conflict among the registered rules actually happens in run time, its resolution rules will handle the conflict and instruct some prescribed actions to the corresponding devices.

The presented framework on event-driven coordination of distributed web-service-enabled devices is expected to contribute

to the efficient implementation of emerging ubiquitous service-based systems.

5. REFERENCES

[1] R.J. Auburn, J. Barnett, M. Bodell, and T.V. Raman. State Chart XML (SCXML): State Machine Notation for Control Abstraction 1.0. W3C Working Draft, 2005.

[2] D. Bank et al. Web Services Eventing. 2004. http://ftpna2.bea.com/pub/downloads/WS-Eventing.pdf

[3] N. Bassiliades, and I. Vlahavas. DEVICE: Compiling production rules into event-driven rules using complex events. Information and Software Technology, 39:331-342, 1997.

[4] S. Calo, and M. Sloman, Policy-Based Management of Networks and Services. Journal of Network and Systems Management. 11(3):249-252, 2003.

[5] A. Carter, and M. Vukovic. A Framework For Ubiquitous Web Service Discovery. In Proc. of the 6th UbiComp, 2004.

[6] M. Cilia, and A. Buchmann. An Active Functionality Service for E-Business Applications. ACM SIGMOD Record, 31(1): 24-30, 2002.

[7] K. Dube, B. Wu, and J. Grimson. Framework and Architecture for the Management of Event-Condition-Action (ECA) Rule-Based Clinical Protocols. In Proc. of the 15th IEEE Symp. on Comp.-Based Med. Sys., pages 288-294, 2002.

[8] A. Friday, N. Davies, N. Wallbank, E. Catterall, and S. Pink. Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments. Wireless Networks 10:631–641, 2004.

[9] Y. Huang, and H. Garcia-Molina. Publish/Subscribe in a Mobile Environment. Wireless Networks 10:643–652, 2004.

[10] K. Liu, L. Sun, A. Dix, and M. Narasipuram. Norm Based Agency for Designing Collaborative Systems. Information Systems Journal, 11(3): 229-247, 2001.

[11] J. Lobo, R. Bhatia, and S. Nagvi. A Policy Description Language. In Proc. of National Conference of the American Association for Artificial Intelligence, Orlando, FL, 1999.

[12] A. Sashima, N. Izumi, and K. Kurumatani. Location-Mediated Coordination of Web Services in Ubiquitous Computing, in Proc. of IEEE Int'l Conf. Web Services (ICWS'04), pages 109-114, 2004.

[13] C.S. Shankar, A. Ranganathan, and R. Campbell. An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments. In Proc. of the 2nd Int'l Conf. on Mobile and Ubiq. Sys., 2005.

[14] S. Vinoski. Integration with Web Services. IEEE internet computing, 7(6): 75-77, 2003.