SOEML: A Smart Object Event Markup Language using Temporal Intervals

Takuro Yonezawa  
Jin Nakazawa  
Hiroshi Sakakibara  
Graduate School of Media and Governance  
Keio University, Tokyo, Japan  
{takuro, jin, skk}@ht.sfc.keio.ac.jp

Goro Kunito  
Tomohiro Nagata  
Research Laboratories  
NTT DoCoMo, Inc, Kanagawa, Japan  
goro@ieee.org, nagatato@nttdocomo.co.jp

Hideyuki Tokuda  
Faculty of Environment and Information Studies  
Keio University, Tokyo, Japan  
hxt@ht.sfc.keio.ac.jp

ABSTRACT
This paper proposes a smart object event markup language. By attaching sensor nodes to everyday objects, users can augment the objects digitally and apply the objects into various services. When creating such smart object services, users should define events, such as beverage of a cup turns cold or someone sits down on a chair, using physical values from sensors. The most common event definition for end-users is simply describing threshold of sensor values and boolean operation. When they want to define more complex events, such as multiple people sit down on chairs or a user starts to study using a pen and notebook, they need to use a programming language. To define such complex event easily without complex programming language, we present a new event description language called SOEMEL based on temporal intervals among simple events. We also provide users a visual interface to support users defining events intuitively.

Author Keywords
Event Definition, Toolkit, Smart Object, Deployment

INTRODUCTION
To realize ubiquitous computing environment, technologies such as computer, sensor, and network has been improving. Especially, small sensor nodes equipped with various types of sensors such as thermometer, accelerator, or illuminometer have enormous potential to create context-aware services that assist a variety of human activities. Our life is filled with everyday objects, and we often have trouble with them (e.g. lost property). It is important to achieve the ubiquitous computing environment to apply everyday objects into ubiquitous services. Sensor nodes, when attached to everyday objects, enable us to gather real-world information as context. Recently, many researchers have been focusing on the ubiquitous services with these "smart objects" [5, 11]. With smart objects, users will be able to enjoy the privilege of ubiquitous technology anytime anywhere in their lives.

To realize the smart object services in the home environment, the following two questions must be answered. The first is how to make our belongings smart. We already have a number of everyday objects. Therefore, providing an easy way to make them smart is important. We have addressed this with uAssociator [13], an easy association method between sensor nodes and objects. The second question, that this paper focuses on, is how to create smart object services which reflect users' requirements. It is not practical to build all the services that users may want to use. Therefore an environment for users to create services themselves is necessary.

The most common smart object applications are described naturally as a collection of rule-based conditions. MediaCup [5] is a common example: "when a MediaCup recognizes that the beverage in the cup is getting cold, MediaCup notifies a user to drink it quickly." To create such services, users must define an event by using physical sensor values and an action such as sending an e-mail or beeping a sound. Because these definitions are unfamiliar tasks for users, there has been an increasing effort and interest in developing infrastructures or toolkits which enables users to create context-aware application by themselves. From the point of view of context definition, these tools can be classified by the following two: low-level toolkits [3, 6] and high-level toolkits [7]. While low-level toolkits provide the needed support for acquiring context, a large amount of codes must still be written to develop simple sensor-rich applications. On the contrary, while high-level toolkits enables end-users to create simple context-aware applications for their instrumented environment, it is impossible to define more complex or flexible context. This is because only setting thresholds of sensor values or using boolean operation is not a sufficient way for defining complex events. For example, "beverage turned cold" cannot be defined simply as "if the cup's temperature <= 40 degrees." This is because "beverage turned cold" implicitly means "the beverage which was hot turned cold." Therefore, the event should be defined as "if the cup's temperature > 40 degrees and then <= 40." This means that it is important for context definitions to describe temporal relation between simple events. Let us present another example. An event "Multiple people...
sat down at the same time” can be useful for recognizing a “meeting” context. In this case, the event can be defined such as “if more than 3 chairs detect a person’s weight at least once in 30 sec.” This type of event is so complex that users need to write highly complex programs.

Our purpose is to create a mid-level toolkit which enables users who have basic information skills to create both simple and complex smart object services. To realize the purpose, this paper focuses on a way to define events which are trigger of smart object services. As shown above examples, it is important to make consideration of temporal relation between simple events for defining more complex events. However, it is difficult task for users to define complex events which concerns temporal concept. For solving this problem, we presents a smart object event markup language called SOEML. SOEML is an XML-format language, and it enables users to define events by using interval-based temporal operation. Allen argues that all events have duration and considers intervals to be the basic time concept [2]. We extended the interval-based temporal operation to define smart objects events flexibly. SOEML has following three features; 1) it enables users to define complex event by combining simple events considering temporal relationship, 2) it enhances reusability of events by separating logic of events and target smart objects, and 3) it supports defining flexible events which are composed of multiple smart objects as “if more than 3 chairs detect a person’s weight at least once in 30 sec”.

The rest of this paper is organized as follows. In next Section, we present details of SOEML and its implementation. In Section 3, we discuss features of SOEML. Then, we survey on related work in Section 4. Finally, we conclude the paper describing foresight in Section 5.

SOEML: SMART OBJECT EVENT MARKUP LANGUAGE

In this Section, we present details of a smart object event description method using interval-based temporal operation, called SOEML. With SOEML, users can define both simple event and complex event easily.

Structure

SOEML is composed of 5 main elements: <event_template>, <temporal_relation>, <time_duration>, <smart_object> and <event>. Figure 1 shows an overview of these 5 elements. Details of each element is shown below in order.

---

**Event Template and Temporal Relation**

Event Template is a basic event model in SOEML. Smart object events can be detected by adapting the Event Template to target smart objects. There are two types of the Event Template: Atomic and Composite.

- **Atomic Event Template**

Atomic Event Template is used for defining primitive events, using a threshold of a sensor value. Figure 2 illustrates an example of Atomic Event Template which defines “temperature is less than 30 degrees.” For describing both Atomic and Composite Event Template, <event_template> element is used. The “type” in <event_template> is used for specifying whether the event template is Atomic or Composite. In element <event_template type="atomic">, users describe concrete sensor information. An element <condition> contains 3 elements: <type>, <sensor> and <value>. <type> is used for specifying sensor node types such as uPart [4] or Mote. <sensor> specifies which sensor on the node should be monitored. In case of using uPart sensor node, there are 3 types of sensors; temperature, movement and illuminance. <value> is used for setting a certain threshold of sensor values. An attribute “exp” in <value> governs types of thresholds. We prepared the following 7 types of thresholds: greater, greater-equal, equal, less, less-equal, between and except.

- **Composite Event Template**

Composite Event Template is a complex event which is defined by the correction of two event templates. For combining multiple event templates, we use temporal interval logic (see Figure 3). In reference [2], a set of 13 relations between intervals are defined, and rules governing the composition of such relations controls temporal reasoning. Additionally, we defined a new relation “any” to the set to improve SOEML’s flexibility. We present detail of “any” relation later in this Section.

Figure 4 shows an example of the Composite Event Template. This template relates two Atomic Event Templates by “meets” relation. Element <temporal_relation>, which has two or more <event_template> elements, is used for defining temporal relationship between event templates. It has a “type” attribute which defines a temporal relation from 14 types of relation shown in Figure 3. In this case, “type” is set to “meets” relation. Atomic Event Template "Some-
contains two kinds of information; sensor information such as a smart object. Figure 6 shows an example of an event named "CupTurnsCold." The element \(<\text{smart_object}>\) contains two kinds of information; sensor information such as node types and node ID, and object information such as object names and owner of the object. Applications, which provide smart objects services, need to know what object each sensor node is monitoring. This, in turn, requires association, or making a semantic relationship between the sensor node ID and its object information. uAssociator which we developed in [13] provides an easy association method and the associating information is stored into a JPEG file in an XML format (see Figure 7). The element \(<\text{smart_object}>\) contains the associating information. By copying this information to SOEML or selecting the JPEG file through a user interface shown in latter in this Section, users can define and use \(<\text{smart_object}>\) easily.

Alike Event Template, an Event has both atomic and composite types. Atomic Event is used for defining events of a
smart object. On the contrary, Composite Event is used for defining events which involves multiple smart objects. Figure 8 is a Composite Event which defines "If both a pen and a notebook are moved at least once within certain 10 second." A Composite Event named "Studying" has a nested construction. First, it relates a Composite Event named "PenAndNotebookMoved" and time duration "10 seconds" with a "during" relation. This means that when the event "PenAndNotebookMoved" occurs within 10 seconds, the event "Studying" fires. The event "PenAndNotebookMoved" is also a Composite Event: it relates the event "PenMoved" and "NotebookMoved" by a relation "any". The relation "any" is a special temporal relation that fires an event when all inner events fires in any relation. Moreover, when "any" is used, additional attributes "firedCondition" and "firedNumber" can be set to the element <temporal_relation> for defining flexible conditions. For example, the code
<temporal_relation type="any" firedCondition="more-equal" firedNumbers="1"/> means that when one or more inner events fire, the condition is fulfilled. There are 5 types of "firedCondition" - more, more-equal, equal, less-equal and less. By using "firedCondition" and "firedNumber", logical addition or exclusive disjunction can be defined.

**Implementation**
In this Section, we describe the implementation of the SOEML system. First, we will present the event detection mechanism. Then, we will show the user interface for creating the SOEML. We implemented these system with Java and JAXB [1] for XML parser. For defining schema for SOEML, we used XML Schema.

**Event detection**
We assume that the users will use smart object services in the home environment where there is one or more computers which operates applications, cooperating with sensor nodes mounted on objects. Events which can be detected are highly influenced by temporal intervals which sensor nodes send packets to a computer. If the intervals of the packets being sent are different, it is impossible to evaluate whether the composite event occurred or not. Therefore, we use sensor nodes which send a packet at an equal interval (500 milliseconds). The current detection mechanism for temporal evaluation is based on an event-driven algorithm. When an event constructing a Composite Event is recognized, a temporal evaluation is executed in every interval (500 milliseconds). Each event stores both the start time and the end time, and the system computes every past events or time duration which users have defined. Note that the start time and the end time of Composite Event or Atomic Event which contain Composite Event Templates are dependent to inner events which construct the Composite Event/Event Template. Figure 9 shows the time chart of the Composite Event Template "SomethingTurnsCold" shown in Section 2. The Event Template "SomethingHot" started from T1 until T2, while "SomethingCold" started from T2 until T3. This temporal order matches "meets" relation, and so the Composite EventTemplate "SomethingTurnsCold" fires. In this case, while the Atomic Event Template "SomethingHot" and "SomethingCold" fires every time when the sensor data matches the event, "SomethingTurnsCold" fires at T2 when "SomethingHot" and "SomethingCold" matches "meets" relation. If "SomethingTurnsCold" is a part of upper Composite Event Template, "SomethingTurnsCold" is treated as an event which has time interval from T1 to T2.

**User Interface**
We also implemented a prototype of user interface for describing SOEML. By using the interface, users can load SOEML, visualize its structure and save as XML code. The interface provides a structure of the elements visually with animation (see Figure 10). Users only need to define thresholds, or select applicable elements from comboBox for defining SOEML. This enables users to configure SOEML intuitively. For deciding thresholds, users can refer concrete sensor data illustrated in chart. The interface also supports associating sensor nodes and objects by uAssocator [13]. This means the interface supports users to install smart objects and define its event.

**DISCUSSION**
We discuss features of SOEML in terms of descriptive capability and reusability.

**Descriptive Capability**
SOEML enables users to define complex events by correlating simple events which is based on thresholds of sensor nodes. Because the correlation of events can be defined by 14 temporal interval logics including Allen’s 13 relations.
The event of SOEML which defines a Composite Event simply as Figure 11. Figure 12 shows a visualized image programming language, users can model the events by SOEML simply as Figure 11. Figure 12 shows a visualized image of SOEML which defines a Composite Event "Meeting". The event "Meeting" correlates "MoreThan3ChairsUsed" event and time duration "30 seconds" by a "during" relation. "MoreThan3ChairsUsed" event is also a Composite Event which uses the temporal relation "any". In this case, "firedCondition" and "firedNumber" is set for defining the condition of "more than 3 chairs being used." With "any" relation, users can define flexible events. However, there are limitations in SOEML compared by programming language. For example, it is impossible to define events which requires an average of sensor values. Alternatively, it is burden task to define composite events which includes repetition of an event. As future work, we will focus to improve flexibility of SOEML for defining these events.

Reusability
In SOEML, an event is expressed as a pair of Event Template and Smart Object. In other words, when users want to adapt the same event recognition logic to an object, the only thing users need to do is to change the Smart Object paired to the Event Template. This feature boosts up the reusability of SOEML. Additionally, all events are based on simple thresholds of sensor values. This reduces difficulty for users to modify SOEML defined by other people. We basically do not assume that the users can use the same event that they have modelled in a different environment. This is because the sensor values that the object detects differs when the environment of user changes. Therefore, the feature that a user can use or easily change an event which another person has made is very important.

RELATED WORK
Dey et. al. interviewed 20 people without programming skills about favorable application for ubiquitous computing environment [7]. As a result, 80% out of 371 proposed applications can be described with if-then rules. This indicates the efficiency of the if-then rule as an end user programming in the ubiquitous computing environment. Following these results, Dey et.al. has built a visual programming environment which enables users to write the if-then rule. However, the definable rules are limited to events possessing a simple operation such as boolean logic. Similarly, there have been many approaches to define services using the ECA(Event-Condition-Action) rule. Shankar et. el. quoted that the ECA rule could not assign conditions before and after firing which was a fault for ECA [10]. Shankar adopted the ECPAP rule which adds the before and after rule, and built petri net for plural rules, increasing the number of rules able to change. However, their target users who defines the ECPAP rule is an owner of the room, not the non-expert users. Jung focused on the users’ mental model, proposing an event definition method using 5W1H [8]. However, to write in a 5W1H method, there must be an environment where the sensor data is able to be understood as context, therefore it is insufficient to use at home or non-instrumented environment.

Composite event detection research has been done in active database research. Yoneki et. al. proposed to introduce interval-based temporal operation to describe events in the sensor network system [12]. They defined 10 types of com-

```xml
<event type="composite" name="Meeting">
  <temporal_relation type="during">
    <event type="composite" name="MoreThan3ChairsUsed">
      <temporal_relation type="any">
        firedCondition = "more" firedNumber = "3">
        <event ref="Chair1Used"/>
        <event ref="Chair2Used"/>
        <event ref="Chair3Used"/>
        <event ref="Chair4Used"/>
        <event ref="Chair5Used"/>
      </temporal_relation>
    </temporal_relation>
  </temporal_relation>
</event>
```

Figure 11. Composite Event of "InMeeting"

```xml
<event type="composite" name="Meeting">
  <temporal_relation type="during">
    <event type="composite" name="MoreThan3ChairsUsed">
      <temporal_relation type="any">
        firedCondition = "more" firedNumber = "3">
        <event ref="Chair1Used"/>
        <event ref="Chair2Used"/>
        <event ref="Chair3Used"/>
        <event ref="Chair4Used"/>
        <event ref="Chair5Used"/>
      </temporal_relation>
    </temporal_relation>
  </temporal_relation>
</event>
```

Figure 12. Visualized SOEML "InMeeting"
Complex event operators such as conjunction, disjunction or concatenation. Though their semantics covers various types of composite events (e.g., including location information), high level language for defining the events is unconsidered. On the contrary, the Tag and Think [9] is a research using temporal relationship to detect events alike ours. In Tag and Think, in order to estimate what the object is from the values from the sensor nodes attached to the object, developer defines the relation of the object’s status and the possible status considering the temporal relationship, and evaluates from the state transition diagram and the status obtained by the experimented data. Tag and Think defines the object’s status with the amount of change of a sensor at a certain time, therefore it enables estimation with high accuracy to various environment. On the contrary, we have focused on not just the same object, but on an environment where users can define events containing multiple objects using the threshold of sensor nodes. Additionally, the statement of a “any” relation enables users to define events flexibly.

CONCLUSION
To enable context-aware services to fit user’s life or respond to users’ request, the importance of toolkit will be increased in the decade ahead. To realize toolkit for smart object services, this paper presents a smart object event markup language called SOEML. Defining events is an unfamiliar task for users, so easy description way without complex programming is necessary. SOEML enables users to define both simple and complex event based on thresholds of sensor values by using XML format. To define complex event, users can use various temporal logic. Additionally, we presented new relation “any” to improve flexibility of events. SOEML also provides a reusability of event because it splits off logic and its target by using the Event Template.

Finally we describe future work. First is an evaluation of our system. We have plan to execute users study. Second is cooperating with other sensors such as location sensors or RFID. By introducing various sensors to SOEML, more complex event can be defined. Third is connecting events to various actuators. With cooperating information appliances, a toolkit which supports to bootstrap and create smart object services can be realized.

REFERENCES