Contact-based Notation for Describing Rules on Sensor Nodes

Takeshi Kanda[†] Yutaka Yanagisawa[‡] Michita Imai[†]
Yasue Kishino[‡] Takuya Maekawa[‡] Takeshi Okadome[‡]

[†]Graduate School of Science and Technology, Keio University
{takeshi,michita}@ayu.ics.keio.ac.jp

[‡]NTT Communication Science Laboratories
{yasue,yutaka,maekawa}@kecl.cslab.ntt.co.jp, houmi @idea.brl.ntt.co.jp

Abstract

In a ubiquitous computing environment, small computers attached to physical objects are required to send sensor data to many other nodes on a narrowband wireless network to provide various types of context-aware services.

To reduce the cost of sending sensor data, this paper proposes a notation to indicate a pair of objects to share data with each other, based on the 'contact' relation between objects. We call this notation, contact-based notation (CbN). In our method, because each node can send data only to nodes contacting the node physically based on the notation, we can reduce both the distance between nodes and the amount of nodes to which to send data.

Moreover, this paper describes an experiment conducted to examine the communication costs on each sensor node described in our proposed notation. The results show that the cost of our method is lower than the cost of the simple broadcasting method.

1. INTRODUCTION

In an indoor ubiquitous computing environment, small sensor nodes will be embedded into a large number of objects in the real world. Each sensor node obtains events related to the object by inference from the sensory data. In this inference process, a node often shares sensor data with other nodes to extract complicated events related to several objects. To share data, in general, a node communicates with another node using a wireless network. Although each node has only a small battery, the process of communication expends much battery power. An increase in the communication cost reduces the battery life of each sensor node.

In existing sensor network systems for indoor environments, each node searches for nodes with which to share sensor data based on broadcasting. For example, when a 'chest' node must check the situation of items included only in the chest, the node broadcasts a message to all nodes around the chest even if the nodes are not inside of the chest. If most nodes are outside of the chest, the broadcasted message incurs much useless communication cost.

To reduce such redundant costs of communication in a sensor network, we introduce a method to limit pairs of objects that communicate with each other based on the 'contact' relations between objects. Our method has the following two advantages:

1. Reducing the distance between a pair of nodes to communicate with.

   In our method, node A can share its data with only nodes $A_0, \ldots, A_n$, which are physically contacting node A. Our limitation can reduce the communication cost because the distance between contacting nodes is quite short. When node A should send data to a distant node B, node A can send data recursively through nodes $C_0, \ldots, C_n$ between A and B, where A is contacting C_0, C_0 is contacting C_1, and C_n is contacting B.

2. Pointing the nodes to send data without ambiguity.

   Because contacting is a strictly physical relation between objects, we can indicate a pair of nodes to share data without ambiguity. For example, in a description "If a book is put near a fire, the system makes an alert," the relation 'near' has ambiguity in each situation. So the user must define the 'near' relation for each service. On the other hand, the physical 'contacting' relation can be strictly defined in any situation as long as each object is not transformable, i.e., it is not a liquid but a solid object. We consider that this strictness guarantees the quality of the services.

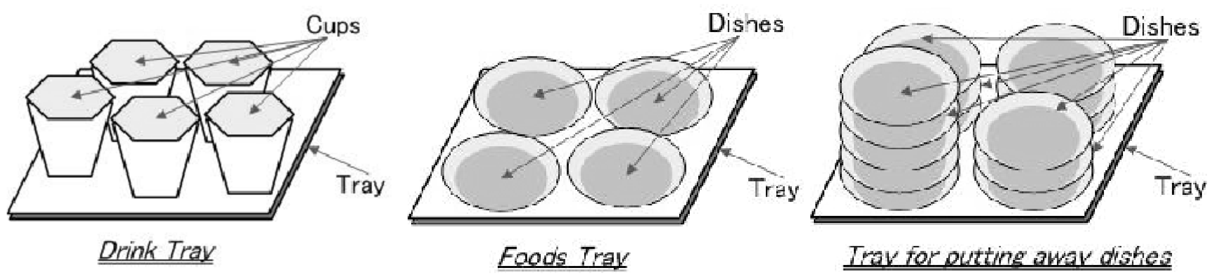

Figure 1. Group Estimation

In this paper, we explain logic-based 'contact-based notation' to indicate a pair of nodes to share sensor data using the physical contact relation. Since the logic languages are one of the most available types of programming languages to describe event-driven application services with short description, this paper describes how to introduce our notation into a logic language. We consider, however, our proposed contact-based notation can be introduced into another language, for example, a functional language and an object-oriented language.

After definition of our notation, we show an experiment to evaluate the reduction of communication cost in our implemented simulator. In this experiment, we compare the cost of our method with the cost of the broadcasting method using several typical arrangement patterns of many nodes. As a result, our method can reduce the communication cost in all cases.

2. APPROACH

This work focused on sensor nodes embedded in a number of objects in the physical world. An application developer must describe rules to extract events using a sensor node as a smart object. By sharing data with nearby smart objects, each smart object infers its surrounding situation.

Our goal is reduction of the communication cost for sharing sensor data among sensor nodes to infer the situation and events occurring in the real world. To achieve this goal, we introduce the contact-based notation to indicate a pair of objects to share sensor data with each other. The contact-based method can reduce the cost of communicating data between a pair of sensor nodes, however, we must consider how to describe the contact-based notation in practical application systems.Before considering the reduction technique, we must discuss the requirements of application systems and how to describe rules using CbN to satisfy the requirements.

The rest of this section illustrates particular applications that utilize smart objects before enumerating the detailed requirements in describing rules.

2.1. Studied Applications

1. **Group Estimation**

   In group estimation, smart agents detect a pre-defined combination of things based on their spatial relations. For example, at a banquet, a waiter puts several dishes on a tray to serve guests. Each tray has a different role that corresponds to the kind of dishes stacked on it. Figure 1 illustrates the different roles of a tray. A tray must be able to handle the information about the dishes on it to decide its role.

2. **Sorting Support**

   In a retail shop, a lack of commodities typically decreases the amount of sales [5]. In sorting support, smart objects detect the lack of commodities or a mistake in their ordering. Sorting support can be used in a large bookstore where books are generally sorted based on such categories as magazine, novel, and non-fiction, as well as by author and edition.

3. **Monitoring**

   In monitoring, smart objects check the physical situations and spatial relations of objects to inform us of pre-defined situations. We can adopt monitoring applications for security guard services in living spaces that can decrease the cost of the guard checking all situations. For example, when breakable objects are laid on a slanted desk, the application informs the guard that the situation is dangerous.

4. **Automated Electric Appliances**

   By attaching a sensor node (and an actuator device) to an electric appliance, it can automatically function corresponding to defined situations. An automated appliance decreases the cost of operating it by hand. We assume an intelligent lighting application that performs

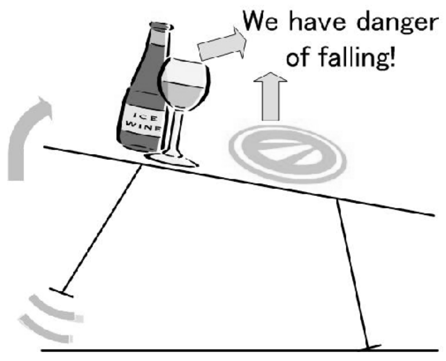

Figure 2. Monitoring

different operations corresponding to objects present. If a user puts a magazine or a notebook on the table, the table lamp becomes well lit. If a user puts a beer on the table, the table lamp is dimly lit.

2.2. Requirements

The above applications require the following abilities.

1. **Recursive Notation based on Contact Relations** Group estimation needs to extract the situation where dishes are stacked. Sorting support needs to extract the order of books sequenced. We can express such situations with recursive notation based on contact relations between objects (Figure 3).

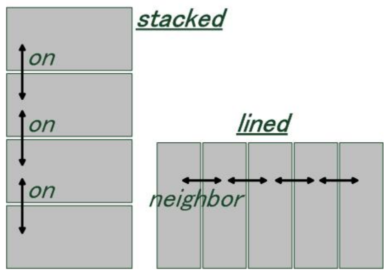

Figure 3. Recursive notation based on contact relations

2. **Path Specification based on Contact Relations**

   In monitoring applications, for example, a smart object often needs to extract situations from the data obtained from objects that do not have direct contact with it. For example, when a desk monitors an object in a box that is placed on it, sensor nodes attached to the desk must receive data from the object that is in the box and not directly contacting the desk. To achieve such communication, the application must specify the path between distant objects based on contact relations (Figure 4).

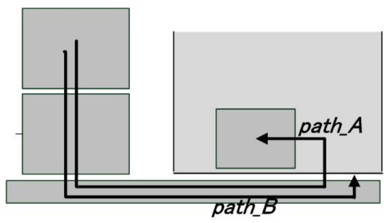

Figure 4. Path specification based on contact relations

2.3. Solution

To fulfill the above goals and requirements, CbN adopts contact relations between objects in the following way.

- **Sharing data between objects in contact,** which achieves efficient communication between sensor nodes while the application is running.
- **Propagation of data through the path of objects in contact,** which achieves recursive notation and relative path specification by sensor nodes.

3. FRAMEWORK

Figure 5 shows the framework of application development on smart objects. In our framework, an application developer uniformly writes rules on several objects with contact-based notation (CbN). The distributor generates sub-rules on each object from CbN rules and sends them to each sensor node. The distributor can automatically generate sub-rules because each CbN rule clearly specifies categories of smart objects. Each sensor node executes a lightweight inference engine to follow the sub-rules. Sensor nodes communicate with contacting sensor nodes and share data with them.

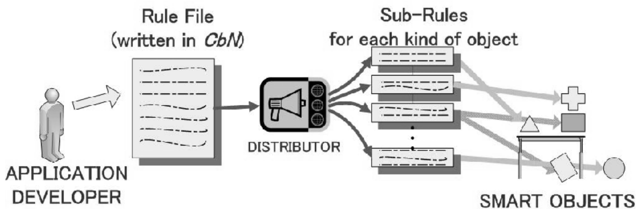

Figure 5. Entire Framework

4. NOTATION

4.1. Overview

This section explains our proposed contact-based rule notation (CbN), which enables smart objects to extract an event cooperatively at low communication costs. Many researchers have suggested several types of Agent Communication Languages (ACLs) that notate the communication between intelligent agents.

Existing ACLs do not, however, provide the kind of notation that can express the spatial relations between agents, since most ACLs are used to describe communications between agents existing on the Internet. Therefore, existing ACLs cannot describe a communication between smart objects in the real world.

We introduce CbN to describe inference rules and communication rules on smart objects in the following way.

- CbN provides special predicates that express spatial relations between contacting objects.
- CbN provides the transmission rules to send data obtained by a smart object to a specified group of smart objects.

4.2. Design

Basic Representation CbN's basic logical model follows first-order logic.

Each smart object is notated as $o_1, o_2, \ldots, o_n$. Variables are notated as $x_1, x_2, \ldots$, while constants are notated as $A_1, A_2, \ldots$. A variable can be substituted by a character string, a numerical number, or a smart object. Variables are categorized into user variables, which a user can freely define, and reserved variables, which a system defines. We explain reserved variables later. Variables and constants are termed arguments and notated as $X_1, X_2 \ldots$. A unit where two arguments are combined with an arithmetic operator $+, -$ is also an argument. For example, $A_1 + A_2$ is an argument if both A_1 and A_2 are arguments.

A predicate is notated as $p(X_1, X_2, \ldots, X_n)$, which is termed an n-argument predicate. A predicate has a truth value, $true$ or $false$, which is determined corresponding to the values of arguments. In this paper, we often abbreviate the predicate $p(X_1, X_2, \ldots, X_n)$ to p. CbN offers two types of predicates: predicates that a user can freely define and reserved predicates that the system automatically gives as the truth value. We explain the reserved predicates later.

A predicate can be combined with another predicate with a logical operator, $\wedge$ or $\vee$. Such a combination of predicates is also a predicate. An inference rule R is defined as $p_1 \rightarrow p_2$, which means "if p_1 is true, p_2 is also true."

The truth value of a predicate is completed in each smart object. For example, a true predicate p in a smart object o_1 does not affect the truth value of the same predicate p in another object o_2. In order to notate a predicate for each smart object, we introduce a special notation $p@o$, which indicates the truth value of the predicate p in the smart object o. This notation is used for transmission rules.

Reserved Predicate A reserved predicate is managed by the system. The form of a reserved predicate is $p(o_1, o_2)$ or $p(o_1, X_1, \ldots)$. $p(o_1, X_1, \ldots)$, which includes one or more system variables. Table 1 shows CbN's reserved predicates.

Table 1. Reserved Predicates

spatial relation	*sr_on, sr_in, sr_neighbor*
sensor value	*sn_temperature, sn_luminance sn_acceleration, sn_humidity*
property	*pr_owner, pr_edition, pr_author pr_weight*
category	*category*

$sr\_on(o_1, o_2)$, $sr\_in(o_1, o_2)$, and $sr\_neighbor(o_1, o_2)$ represent the spatial relations between smart objects (Figure 6). (a) sr_on represents a support-supported relation between smart objects. (b) sr_in represents an inclusion relation between smart objects. (c) $sr\_neighbor$ represents a neighbor relation between smart objects.

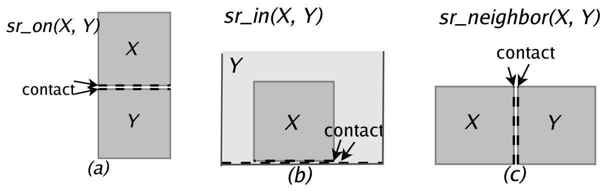

Figure 6. Spatial relations: (a)sr_on (b)sr_in (c)sr_neighbor

Transmission Rule A transmission rule provides the notation needed to copy a predicate from one smrt object to another. Using the transmission rule, each smart object can extract an event and select an action with the data of another smart object.

A transmission rule is notated in the following form.

$$(p_1@x_1 \wedge p_2@x_1 \wedge \ldots \wedge p_n@x_1 \rightarrow p_0@x_0)$$

Note that $p_1, p_2, \ldots, p_n$ must include a spatial relation denoting that x_1 contacts x_0.

Example of Recursive Notation CbN can describe recursive notation based on contact relations. Below is an example of recursive notation that represents a situation where several smart objects are stacked.

- Inference Rule
  $sr\_on(o_1, o_2) \rightarrow rec\_on(o_1, o_2, 1)$
  $rec\_on(o_1, o_2, x_2) \wedge rec\_on(o_2, o_3, x_3) \wedge x_1 = x_2 + x_3$
  $\rightarrow rec\_on(o_1, o_3, x_1)$

- Transmission Rule
  $sr\_on(o_0, o_1)@o_1 \wedge rec\_on(o_1, o_3, x_1)@o_1$
  $\rightarrow rec\_on(o_1, o_3, x_1)@o_0$
  $sr\_on(o_3, o_4)@o_3 \wedge rec\_on(o_1, o_3, x_1)@o_3$
  $\rightarrow rec\_on(o_1, o_3, x_1)@o_4$

Example of Path Specification to a Distant Object CbN can describe relative path specification based on contact relations. Below is an example of relative path specification in the situation illustrated in Figure 4.

- Inference Rule
  $rec\_on(o_1, o_3, 2) \wedge sr\_on(o_4, o_3) \rightarrow path\_B(o_1, o_4)$
  $path\_B(o_1, o_4) \wedge sr\_in(o_5, o_4) \rightarrow path\_A(o_1, o_5)$

- Transmission Rule
  $sr\_on(o_4, o_3)@o_3 \wedge path\_B(o_1, o_4)@o_3$
  $\rightarrow path\_B(o_1, o_4)@o_4$
  $sr\_in(o_5, o_4)@o_4 \wedge path\_A(o_1, o_5)@o_4$
  $\rightarrow path\_A(o_1, o_5)@o_5$

5. PROTOTYPE

Here, we mention our implemented prototype sensor node. Our nodes (Figure 7) can detect the contact relations of other sensor nodes. Each sensor node also runs a logical inference engine and can communicate with other sensor nodes. Their specifications have been given in previous papers [3]. Accordingly, this section simply provides a summary of the specifications.

In our prototype, we have a device to detect a pair of contacting objects. Our device is similar to RFID. Each node has a thin coil to generate lines of magnetic force, another coil can detect the lines when a coil moves near the other coil. The coil can send an ID signal to the other coil by changing the strength of the magnetic force. In this way, each node can know which node moves near the node. Our current implemented coil can detect only one coil, but in the future, we will improve this device to detect multiple coils. Figure 8 illustrates our developed coils (wrapped by a thin black package) to detect contacting objects.

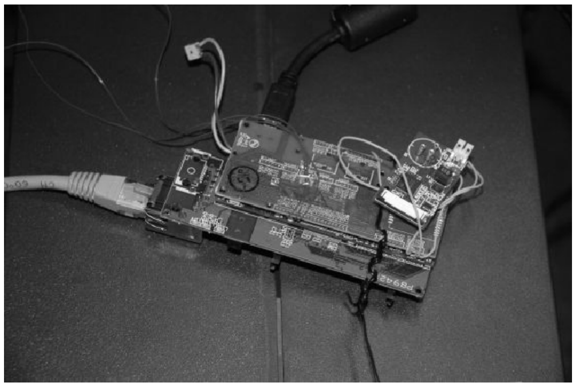
Figure 7. Sensor node core

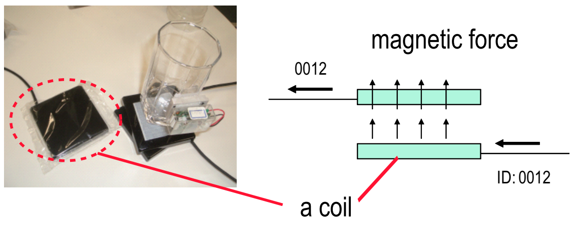

Figure 8. Sensor device to detect contacting objects

Our sensor nodes can share data only with the other contacting sensor nodes in the following ways. 1) When two sensor nodes make contact, they send and receive their IDs through their detection sensors. 2) After they recognize each other, they start to share context data by unicast. 3) Once the contact fails to be detected, the communication is broken.

Figure 9 shows the composition of our developed inference engine that works on the sensor node.

The inference engine performs forward reasoning by using sensory data obtained from several sensor nodes. As a result of the inference, it extracts a local situation from shared sensory data.

6. EVALUATION

6.1. Read Counts

In order to confirm that our framework decreased communication costs between sensor nodes, we compared *read* operations in the conventional communication method — where each sensor node broadcasts data to other sensor nodes within a constant distance — with those described in CbN.

The experiment was conducted by computer simulation, because we have only several prototype devices currently.

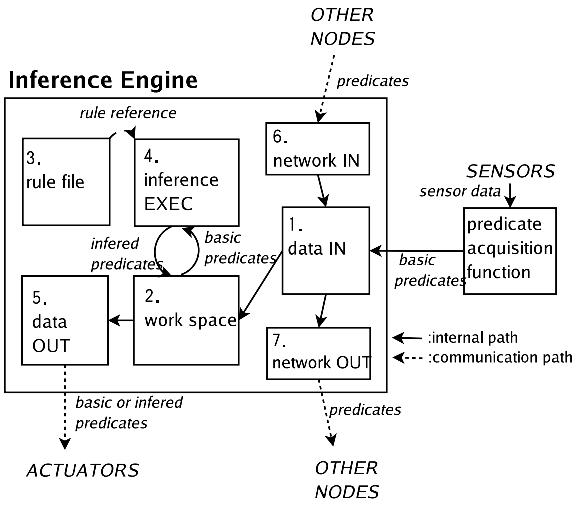

Figure 9. Inference engine on sensor node

We set up three indoor environments, each of which differs from the others in its density of smart objects (Figure 10): (a) illustrates an environment containing 5 smart objects within 1 square meter, (b) has 10 smart objects, and (c) has 20 smart objects. In each environment, we simulated the existing communication method (simple broadcasts) and the proposed communication method (CbN's transmission rules). In the broadcast, the simulation program assumed that each sensor node shares data only with the sensor nodes that are within a constant distance, not with any sensor node outside of this distance. The simulation program counted the *read* operations of each sensor node at three distances: 10 cm, 20 cm, and 50 cm. In the proposed communication method, the simulation program assumed that each sensor node follows CbN's inference rules and transmission rules. The simulation program counted the *read* operations of each sensor node under the assumption that they follow the rec_on and rec_in rules explained in section 4.

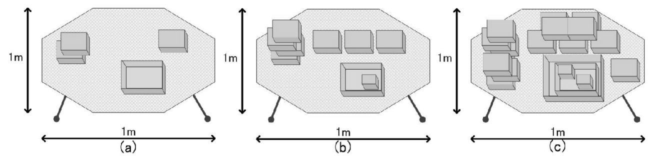

Figure 10. Simulation environments: (a) 5 smart objects (b) 10 smart objects (c) 20 smart objects

Table 2 shows the average *read* counts of smart objects in the three environments.

The results of the simulations reveal that the broadcast-based communication proportionately increases the communication costs as the number of smart objects increases.

Table 2. Average read counts

	5 /m^2 (a)	10 /m^2 (b)	20 /m^2 (c)
contact base	2.0	3.2	5.0
broadcast (10cm)	2.0	3.8	6.5
broadcast (20cm)	2.0	4.7	8.0
broadcast (50cm)	3.2	6.7	13.2

In the environment containing 5 smart objects (environment 1), for example, the average *read* count in the broadcast (50 cm) was 3.2, while in the environment containing 20 smart objects (environment 3), the count increased to 13.2. Similarly, in environment 1, the average *read* count in the broadcast (10 cm) was 2.0, while in environment 3, this count increased to 6.5.

On the other hand, the results show that the increase in the contact-based communication was smaller than that of broadcast-based communication. For example, in environment 1, the average *read* count was 2.0, while in environment 3, the count was 5.0.

We conclude that CbN enables each sensor node to communicate at low communication cost even in an environment dense with smart objects. This is because the number of communication targets does not depend on the density of smart objects but on the number of smart objects in direct contact.

7. RELATED WORKS

This section describes related works, which are categorized into two groups: 1) Smart objects, where sensor nodes embedded in objects cooperatively work to extract an event; and 2) Rule description techniques in sensor networks.

First, we explain smart objects. MediaCup [1], Cooperative Artefact [7], and u-Texture [4] are representative studies on smart objects. Beigl et al. proposed the MediaCup project to create an intelligent cup with a small sensor node that has sensing, data processing, and communication capability. Each cup can share data with neighboring cups and comprehend the physical situation of the cups such as "the cup is stationary," "the cup is moving," and "someone is drinking out of the cup." Strohbach et al. suggested Cooperative Artefact, where several objects share knowledge and cooperatively infer surrounding contexts. Each object detects neighboring objects by using an ultrasonic transmitter and an ultrasonic receiver and shares sensory data with the neighboring objects to infer surrounding contexts. Cooperative Artefact can be applied to chemical containers, which allows cargo systems to detect hazardous situations. Kohtake et al. designed and implemented u-Texture, a block that can detect its situations. Each block can detect its vertical direction and the existence of neighboring blocks. Each block displays various information depending on the combination of blocks. Consequently, u-Texture can support assembly line workers.

Second, we explain the rule description techniques in sensor networks. SICL [6] and RuleCaster [2] are representative applications of this technology. Siegemund et al. proposed a communication platform for smart objects that takes an object's current context into account and adapts networking structures accordingly. They developed a high-level description language, called SICL, and a compiler that generates corresponding code for smart objects. The platform provides mechanisms for implementing context-aware communication services and a communication abstraction for inter-object collaboration. Bischoff et al. proposed a state-based programming framework in indoor sensor networks, named RuleCaster. RuleCaster provides a high-level specification to individual tasks and assigns them to the nodes. The RuleCaster compiler provides two strategies of rule distribution: (1) centralized distribution, and (2) decentralized distribution. In the centralized distribution strategy, one central node is responsible for all rule processing, while the other nodes only provide their services to the central node for condition evaluation. In the decentralized distribution strategy, an overlay network consisting of several subnetworks is generated. Processing is distributed over several nodes that are responsible for their own spatial region.

8. CONCLUSIONS AND FUTURE WORK

This paper proposed a framework to describe rules for extracting an event in smart objects. To reduce the number of other sensor nodes with which each sensor node must communicate, we introduced CbN to provide an application developer with the notation needed for specifying the communication pairs of sensor nodes. Our framework can be used for building such applications as group estimation, sorting support, and situation monitoring.

One experiment was conducted to evaluate the communication costs of our framework. The results revealed that our framework reduced the number of *read* operations on each sensor node.

Finally, we concluded that CbN enables smart objects to extract an event at low communication cost when sensor nodes are embedded in a large number of objects.

References

[1] M. Beigl, H. W. Gellersen, and A. Schmidt. Mediacups: Experience with design and use of computer-augumented everyday artefacts. *Computer Networks*, pages 401–409, March 2001.

[2] U. Bischoff and G. Kortuem. A state-based programming model and system for wireless sensor networks. *PerSens 2007*, March 2007.

[3] T. Kanda, Y. Yanagisawa, T. Maekawa, M. Imai, H. Kawashima, and T. Okadome. A distributed inference system on sensor nodes using neighbors' context data. *SWOD 2007*, April 2007.

[4] N. Kohtake, R. Ohsawa, T. Yonezawa, Y. Matsukura, M. Iwai, K. Takashio, and H. Tokuda. u-texture: Self-organizable universal panels for creating smart surroundings. *UbiComp 2005*, pages 19–36, September 2005.

[5] C. Metzger, J. Meyer, E. Fleisch, and G. Troester. Weight-sensitive form to monitor product availability on retail shelves. *Pervasive 2007*, May 2007.

[6] F. Siegemund. A context-aware communication platform for smart objects. *Pervasive 2004*, pages 69–86, April 2004.

[7] M. Strohbach, H. W. Gellersen, G. Krtuem, and C. Kray. Cooperative artefacts: Assessing real world situations with embedded technology. *UbiComp 2004*, pages 250–267, September 2004.