A Framework for Interacting Smart Objects

Arnab Sinha, Paul Couderc

A Framework for Interacting Smart Objects

Arnab Sinha and Paul Couderc

INRIA, Rennes-Bretagne Atlantique,
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
{arnab.sinha,paul.couderc}@inria.fr
http://www.inria.fr/en/en/teams/aces

Abstract. In this paper, we propose a framework enabling physical objects used everyday to participate in smart interactions. These objects are RFID tagged containing self description of their properties. The paper describes how smart context aware services can be supported directly by a collection of smart objects. One of the main advantage of our approach is the smart objects being piggybacked with necessary information, optimized enough to make inferences locally, without dependence on external system support.

Keywords: internet of smart objects, collection, object property, RFID tag, NFC, context representation, pervasive, ubiquitous computing

1 Introduction

Smart objects and smart environments are core concepts in pervasive computing. A common understanding of the internet of things is the ability of more and more usual objects to be connected or referenced in the internet (or in cloud services). Since their services are dependent on the network infrastructure, occurances of failures are probable due to unavailability. Protecting privacy is an increasing concern. This motivated us to think of ways that would enable pervasive applications to take these collective decisions without external support. The objects must be piggybacked with the necessary information enabling them to take part in spontaneous decision making processes locally and avoiding remote communication. We call them as "self-described" objects due to semantic properties being attached to them. The intention is to primarily support limited but focussed decisions for pervasive applications. This information can be put into the RFID tag of the objects. RFID tagging is a developing trend, which mostly consists of storing a reference to remote information. Our approach proposes a step forward, to put some extra bits of information that could help in taking basic spontaneous decisions. This also provides a utilization of its limited memory space, a constraint that would decrease in the forthcoming years. The availability of information locally, would make the system more scalable while reducing on the network usage. Lastly, there are concerns of security with centralized databases containing huge amount of personal information. The access trends and patterns could have possibilities of being stored, profiled and misused resulting in breach of privacy.

Let's consider the ready to assemble furnitures like IKEA^{TM}. The stores sell the disassembled pieces of a furniture packaged together. It needs to be assembled by the users taking the help of an instruction manual provided along. We propose that the important pieces be self described with NFC tags. Using a reader, an user can scan tags individually for information about them. Additionally, they are notified with information about their adjoining pieces from the remaining unscanned set. A NFC-enabled mobile could serve the purpose as a portable reader with an application that can be reused for providing this service independently of the puchased set of furniture.

Waste management is another domain for the application of collective decisions. Self describing the waste items with their properties could make the management simpler. For example, information like its composition would help to perform better sorting. A plastic bottle dropped in a glass bin can contaminate its entire contents and should be detected.

Analysing the situations closely, we can make few observations. Every item contains the necessary information for its self description. Based on the collective information about all the items present locally, the system suggests or makes some decision depending on the domain in consideration. This would be referred as **inference** in the rest of the paper. Inferences made could be pro or against by nature. In the first example, the application suggested to look for adjoining pieces while a furniture is in the process of assembling. The domain requires inferences, pro in nature recommending for adjoining pieces. While for the waste domain, inferences are made for incompatibilities as the application alerts when a problematic item is being added to a bin.

2 Approach Principles

Centralized approach

If this classical approach is used, the domain knowledge would be stored in a server or centralized in an external database. The objects of the ubiquitous application would require to refer this central knowledgebase each time for making inferences. Hence, there should a communication infrastructure in place to transmit the data back and forth. Present RFID tags containing reference is an example of such approach.

Distributed approach

This approach stores the domain knowledge in pieces spread across devices in physical space. One of such devices to hold these pieces are passive proximity communication tags like RFID/NFC which have limited memory space. Their handheld readers might also have limited computation capabilities. So, efficient context representation is required for optimum performance to make collective inferences.

Our approach

In this paper, we have proposed a framework to encode the knowledge for making inferences in such a resource-limited distributed environment. Sometimes it may need to have a trade-off between the partial knowledge possessed by each item and keep the rest available locally. Its most important aspect is that the inferences can be done locally without references to any centralized knowledge-base. In the next section, we would explain our framework to encode information for item interactions.

Our framework proposes an encoding method to self describe items of a domain in an efficient way. Inferences are made among a collection of such items present locally. If the information in the tags are stored in the proposed format, our generic algorithm (described later) can be reused without any modifications and irrespective of the domain. The entire implementation can be classified as intelligent product as described in [1–3], with an exceptional advantage that items don't possess or require unique identity for our purpose.

The functioning of our framework is also comparable to the Internet of Smart Objects [4]. The physical objects are RFID tagged containing information to transform into smart objects. These smart objects collectively form an Internet of Things. The aggregation of the information contained in the local IoT provides interesting inferences and services. Our situation can be better called as Intranet of Things (InoT) as inferences could be made with the smart objects located locally without using any network for communication [5].

3 Inference Model

The examples described in section 1 are not exhaustive. There are many other real world domains which could work on similar principles have been discussed later. This section, discusses the general model about how a domain knowledge be interpreted in terms of its properties. This knowledge is also encoded into the smart objects so that inferences could be made locally.

3.1 Defining domain properties for inferring

Let's consider a sample domain D to explain our general model to infer the interactions. We will refer to this example throughout the paper. Suppose D consists of *10* marked important properties $\{p_1, p_2, ... \, p_9, p_{10}\}$ and some of these properties are incompatible to each other. Graph G represents the knowledge graphically as in figure 1. Each of the properties in G is represented as a vertex and an edge is drawn between two properties if there exists an incompatibility between them. G represents the global knowledge of all the properties of our sample domain. Our objective would be to distribute it in a way such that the entire domain knowledge is not required for making inferences. Subsequently, we have proposed a format for encoding the distributed knowledge for items having flexibility inferring simple to complex interactions. Making such inferences would be the objective for a collection of items.

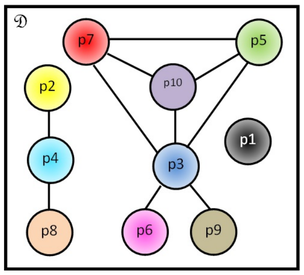


Fig. 1. Incompatible properties of the sample Domain

Table 1. Incompatibility of properties

Property	Incompatible with
p_1	-
p_2	p_4
p_3	p_7, p_5, p_6, p_9, p_{10}
p_4	p_2, p_8
p_5	p_7, p_3, p_{10}
p_6	p_3
p_7	p_5, p_3, p_{10}
p_8	p_4
p_9	p_3
p_{10}	p_3, p_5, p_7

From figure 1, we tabulate each property separately along with its incompatible ones as in table 1. The idea is similar to creating an adjacency matrix from a graph. But their finer difference lies in the format for representing the information.

Initially we require to have a knowledgebase containing the interaction of the properties of the domain. Using it, the properties of the items are self described using RFID tags. A tag would contain the description string in the format $\{properties : incompatible\_properties\}$. It consists of two fields with a colon as separator between them. For example, if an item I of D possesses property p_3, its tag would contain the information in the format:

$$p_3 : p_5,\ p_6,\ p_7,\ p_9,\ p_{10}$$

It is basically the third row of table 1. An item can have multiple properties. For such a case we perform an union of the individual fields. An item having the properties p_4 and p_6 would be self described as:

$$p_4,\ p_6 : p_2,\ p_8,\ p_3$$

An alternative solution for the above would be affixing multiple tags i.e. one tag each for the properties p_4 and p_6 written in the same format. If an item has too many properties to hold in the memory of one RFID tag, this is the easiest and cost effective method.

Until now we have suggested an encoding method to describe the items with their properties in the tag. Given a set of such items locally, our objective is making inferences, pro or against depending on the domain. Algorithm 1 provides an outline to perform this task. It is executed once each time an item I_i is detected by RFID reader. It is assumed they are self described in the same format as above. It could be used for inferring incompatibilities between the newly introduced item and the items present locally on pairwise basis. The reader reads the two fields of the tag in two different arrays as $Prop_i[]$ and $Iprop_i[]$. Additionally, it contains a set $S\{\}$ which is initialized to $\varnothing$ when the algorithm

Algorithm 1: Inferring incompatibility of items

```
Input: item $I_i$ detected by the RFID reader
Output: infer if item $I_i$ is compatible with existing set of items
Initialize: Set S ← ∅
while item $I_i$ detected by the RFID reader do
    set flag to TRUE i.e. no incompatibilities inferred for item $I_i$
    Read the tag and store it's two fields in $Prop_i[]$ and $Iprop_i[]$
    for each element Ip in $Iprop_i[]$ do
        if $Ip \in S$ then
            set flag to FALSE i.e. Ip an incompatible property to item $I_i$, is
            incompatible with one or more existing set of items
    if flag is TRUE then
        for each element p in $Prop_i[]$ do
            $S \leftarrow S \bigcup p$ i.e. adding the properties of item $I_i$ to set S
    return flag
```

is executed for the first time. It caches the local context by incrementally storing the list of compatible item's properties, when added.

Referring back to the example in the beginning of this section, suppose we have four items I_1, I_2, I_3 and I_4 having properties p_2, p_1, p_3 and p_9 respectively. The interesting observation is that inferences would depend on the order of items added and the existing collection present locally. From the graph theoretic point of view, it can be stated that the algorithm prevents forming any subgraph of the graph G in figure 1.

3.2 Inferring incompatibility in groups

Until now, we have discussed inferences among pairs of properties. There may be some application domains where inferences are based on groups of properties that are present together locally. For example in figure 1, properties p_3, p_5, p_7 and p_{10} can be considered incompatible as a group. The important aspect of being in group is that incompatibility is only ensured with the presence of every property of the group. Such groups are represented as cliques in a graph. A clique of a graph G is a complete subgraph of G. In the graph G of our example, the group of properties p_3, p_5, p_7 and p_{10} forms a clique. This information has to be distributed in a way such that it could be efficiently encoded in the RFID tag to self describe the item. For the purpose we construct a string in the following format:

$$f_0\text{:}f_1\text{:}f_2\text{:}f_3\text{: ... :}f_n$$

The string consists of n fields with colon as separators with each containing a set of properties. The first field f_0 describes the properties of the item. This is similar to the previous example. The second field f_1 contains all individual pairs of incompatible properties. The groups of incompatible properties are encoded in the fields from f_2 to f_n. From the example, an item can self describe itself as having property p_3 with:

$$p_3 : p_6, p_9 : p_5, p_7, p_{10}$$

Summarizing the information for all the properties as it would be encoded is presented in table 2.

Property f_0	Incompatible with f_1:f_2:f_3: ... :f_n
p_1	-
p_2	p_4
p_3	p_6,p_9:p_5,p_7,p_{10}
p_4	p_2,p_8
p_5	-:p_3, p_7,p_{10}
p_6	p_3
p_7	-:p_3,p_5,p_{10}
p_8	p_4:-
p_9	p_3:-
p_{10}	-:p_3,p_5,p_7

Table 2. Incompatibility of properties

To accomodate this additional feature about groups of incompatible properties, algorithm 2 outlines a procedure to make such inferences. It has some additions to algorithm 1. It ensures that none of the incompatible pairs of properties as well as groups doesn't occur locally. Taking example from the graph in figure 1, the algorithm infers incompatibility when some of the properties among p_2, p_3, p_4, p_6, p_8 and p_9 are added locally and tries to form a subgraph. These are the properties that form incompatible pairs in the graph. Additionally, we have the properties p_3, p_5, p_7 and p_{10} forming a clique as described earlier. So collectively the situation would inferred safe until all but one of these properties exist locally. The algorithm combines both of the above to make inferences. An exception would be property p_1 which is disconnected from the rest of the graph. So it is always inferred safe regardless of other properties present locally.

4 A Smart Tool Box Application

The **Smart Tool Box** is an example of ubiquitous application which helps to maintain safety standards at sensitive sites like aircrafts [6]. Let's consider a workshop has a depot for tools which issues them to its workers in a toolbox. Each type of tool in the depot is assigned an unique identification. Every tool is NFC tagged which stores its type. When a worker requests for some tools, they are lent out in a tool box. Henceforth the box should always contain all

Algorithm 2: Inferring incompatibility for group of items

```
Input: item $I_i$ detected by the RFID reader
Output: infer if item $I_i$ is compatible with existing set of items locally
Initialize: Set S ← ∅
while item $I_i$ detected by the RFID reader do
    set Flag to TRUE i.e. no incompatibilities inferred for item $I_i$
    Read field $f_1$ from the tag
    for each property p in field $f_1$ do
        if $p \in S$ then
            set Flag to FALSE i.e. the property p is present locally hence $I_i$ is
            incompatible
    for each field $f_i$ from $f_2$ to $f_n$ do
        set GroupFlag to FALSE i.e. assuming the all the incompatible
        properties in $f_i$ are present locally
        for each property Ip in $f_i$ do
            if $Ip \notin S$ then
                set GroupFlag to TRUE i.e. Ip an incompatible property in field
                $f_i$ is not present locally
        if GroupFlag is FALSE then
            set Flag to FALSE i.e. all the properties in field $f_i$ are present
            locally hence $I_i$ is incompatible
    if Flag is TRUE then
        for each element p in $Prop_i[]$ do
            $S \leftarrow S \bigcup p$ i.e. adding the properties of item $I_i$ to set S
    return Flag
```

the tools grouped together and warn the user if one or more tools are missing until returned back to the depot. This group information is written onto every tag before handing out the kit to the worker.

In our framework in section 3, an approach is proposed for self describing items with properties to make inferences. In the present context, the domain consists of tools used for maintainance. In the following subsections, we have illustrated how the domain properties could be represented efficiently and the framework used to perform grouping of tools at the depot, on request. In 4.2, we describe how our objective is achieved using a simple android application. Its effectiveness is discussed in 4.3.

4.1 Numeric Representation of Tools Domain

In our implementation, we have used numeric representation of properties written on their NFC tag in the format described in section 3.2. So each type of tool performing distinct function in the depot is assigned a unique natural number. Suppose the depot lends out specific type of toolbox is assigned number 2 along with tool types numbered 14, 5, 6, 26 and 9 as in figure 4.2. The self description string for the group written onto the toolbox in our proposed format would be

12:0:21405062609

Every field of the string description begins with a number specifying the fixed length of the properties. Numbers are padded with '0' on the left to make them all of equal length. According to the proposed format, field f_1 should contain the types having pairwise association with the toolbox. Hence the field f_1 contains 0 which represents properties of 0 length i.e. another way of saying that no property exists for the field.

4.2 The Application

We have developed a small android application to validate and verify the working of our current example. There is a subtle difference between the examples discussed presently with that in section 3. The group of tools in the toolbox strive to remain together throughout and our application tries to verify this situation which we referred to as an inference pro by nature in section 1. Our framework could be utilized for the current scenario but with a minor change. In our framework described in section 3, items are self described with it's own properties and the other's to which it would be incompatible. In this context, the idea is reversed for groups and algorithm 2 is modified accordingly. Figure 6 demonstrates the various phases of the application.

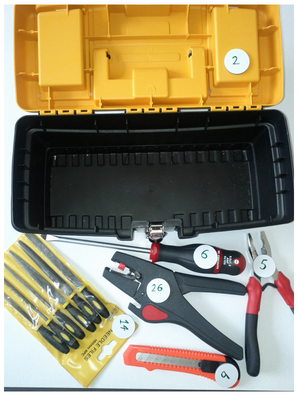


Fig. 2. Initial grouping of Tool Box and contents

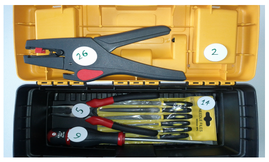
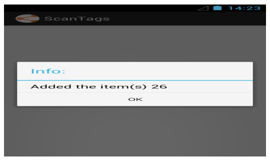
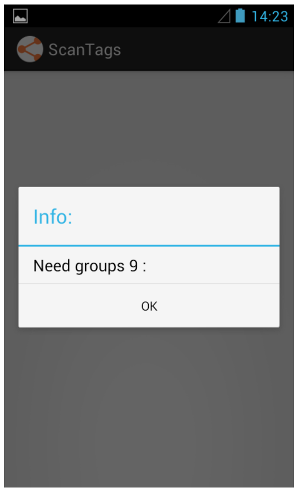


Fig. 3. Integrity Verification of the Tool Box



Added the item(s) 26

Fig. 4. Acknowledgement on adding tool numbered 26



Need groups 9 :

Fig. 5. Notification for tool(s) yet missing

Fig. 6. Smart Tool Box Application

4.3 Validation

Given the nature of our application, we have a validation approach with an use case implementation of working prototype. In this section, we have discussed the performance of the approach.

Coding efficiency The length of each field in the self description would depend on two factors. The maximum length among the group of numbers in the particular field are chosen as fixed length. So each number having smaller lengths are padded on their left adding to the overall length of the field. However if the maximum length among the numbers is less, then the encoded string automatically gets reduced.

Suppose we have the following sets:
A domain $D = \{x \epsilon N\}\ and\ |D| = n$, where n is the number of properties represented.
Any field in the self-description of an item could be represented as set $F \subset D$ ...(i)
Let's define the following functions for some operations:
$|X|$ gives the cardinality or size of $set\ X$.
$max(X)$ gives the largest element of $set\ X$.
$len(element\ e)$ gives the character count of element e.
We can say that,
$max(F) \leq max(D)$
or, $len(max(F)) \leq len(max(D))$ [This gives the character count of highest elements of both sets] ...(ii)
Now, if we represent the field in (i) using the fixed length, we can write
$|F|\ *\ len(max(F)) \leq |F|\ *\ len(max(D))$
Hence we can make three observations for the encoded length of the field which is the L.H.S of the equation:
1. It depends on the number of properties grouped together represented as $|F|$.
2. The length of the largest number also contributes to the total length.
3. The maximum length is bounded upto the R.H.S of the equation.

Figure 7, provides an estimate on the upper bound of the characters required for encoding groups of properties in a field depending on the domain size. The maximum length of the properties for the domain are considered in the graph and hence the lines outline the upper bound on the total length of the field which can never be exceeded. However there may be circumstances when the total length is reduced if the group of numbers in the field comprises of smaller numbers.

For domains having average number of properties, they can be represented by the natural numbers until 99 having 2 characters as fixed length in the fields. For bigger domains with more properties, larger numbers can be used. But using natural numbers upto 999 would probably be sufficient for most domains as it has large interval of numbers. Also the encoding length is guaranteed to remain within the upper bound for the entire interval given the number of properties for a group.

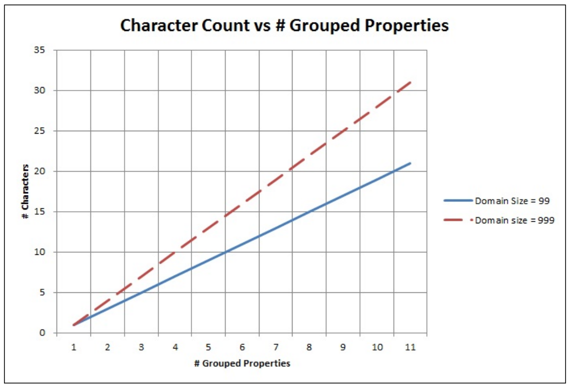


Fig. 7. Encoding performance

The entire encoded string data to self describe an item contains numbers and the special character ':'. When the data is written onto RFID tags, we replace the colon with character 'a' and use 'b' as the data delimiter. Hence the string is transformed to hexadecimal and each of it's character consumes four bits of memory. Hence using numeric properties gives us an advantage whereas in some cases it could even take upto two bytes for a character [7].

Advantages : Deployment, Scalability and Privacy The entire setup is very easy to build. It is effective in terms of cost and time. Initially, it would require identification of the properties of the domain i.e. different type of tools and assigning them unique identifier using cheap NFC tags. The grouping could be done later on-demand, based on the requirement of a worker. Reading and writing of the NFC tags are performed with applications using an android phone. When a large number of tools are grouped together, scalability factor comes into play in two different ways. Firstly, the limited memory space of tags on each tool could pose a problem. This is resolved by sticking multiple tags instead of one. Secondly, the information is available locally for the entire setup. So, the absence of a network reduces the communication time and the only time taken is for the read/write of tags, performed one at a time. This also addresses the concern for privacy. Last but not the least, this approach is simple, energy efficient due to use of passive tags and have moderate user interaction compared to the approach in [8].

5 Related Work

We have proposed a framework in the context of ubiquitous computing research supporting different application domains where smart physical objects are adaptive to decisions based on their collection of objects present locally. There are some similarly work that makes inferences dependent on collectivity [6, 4, 9–11].

[12] demonstrates sorting by a smart bin that accepts or rejects self described waste items containing its percentage composition. Our framework can be used to perform sorting that infers on the compatibility of a self described waste item with the ones already present in the bin.

Antifakos et al. describes in [8], how user manual is avoided by most people and proposed an proactive approach providing guidance to assemble the pieces. Later, Michahelles explains their experience about how this approach failed to attain the objective, as it was a radically new seeking too close cooperation with the users [13]. We have proposed a middle path between these approaches.

Context representation is the language to express collective situation and knowledge. The authors in [14] mentions semantic based ontologies among the many in their extensive survey whereas [15] have demonstrated efficient encoding of semantic data using prime numbers for resource constraint devices which are commonly used in ubicomp solutions.

6 Conclusion

In this paper, we have suggested a framework to represent context for a collection of physical objects. The required knowledge is optimally distributed to contain with the objects which make them smart. We have also discussed how it could be utilized for various ubiquitous applications. Finally, we would like to highlight some of the advantages of our proposed framework:

Privacy and security are important concerns in ubiquitous computing especially in relation to cloud services. The self describing objects do not involve any sensitive information about the items or personal information about the users. Additionally, using numeric properties appear as obscure data to an intruder reading the tag. So, encryption is not necessary.

As stated earlier, our objective for the framework is context representation for inferring activities involving a collection of items. This has been clearly achieved in a decentralized manner by self describing the domain items. In this way we achieve to make inferences locally without any references to external knowledge-base or requiring communication infrastructure. An alternative could be storing the domain knowledge as a key in the local embedded system. But in case of changes, updating the tag contents at source is easier and it would be effected automatically.

Perspectives to this work would include supporting more applications such as smart drugs [6, 16]. We also intend to seek for alternative to using numeric properties and better encoding approaches for compact representation.

References

1. D. Mcfarlane, S. Sarma, J. L. Chirn, C. Y. Wong, and K. Ashton, "The intelligent product in manufacturing control."
2. C. Y. Wong, D. Mcfarlane, A. A. Zaharudin, and V. Agarwal, "The intelligent product driven supply chain," in *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, pp. 4–6, 2002.

3. T. Lopez, D. Ranasinghe, B. Patkai, and D. McFarlane, “Taxonomy, technology and applications of smart objects,” *Information Systems Frontiers*, vol. 13, pp. 281–300, 2011. 10.1007/s10796-009-9218-4.
4. G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, “Smart objects as building blocks for the internet of things,” *IEEE Internet Computing*, vol. 14, pp. 44–51, Jan. 2010.
5. A. Sinha and P. Couderc, “Smart bin for incompatible waste items,” in *Proceedings of the the 9th International Conference on Autonomic and Autonomous Systems (ICAS 2013)*, 2013.
6. K. Römer, T. Schoch, F. Mattern, and T. Dübendorfer, “Smart identification frameworks for ubiquitous computing applications,” in *Wireless Networks*, pp. 689–700, 2003.
7. Y. Glouche and P. Couderc, “An autonomous traceability mechanism for a group of rfid tags,” in *Proceedings of the 6th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2012)*, 2012.
8. S. A. Florian, F. Michahelles, and B. Schiele, “Proactive instructions for furniture assembly,” in *Proc. Ubicomp 2002, Gothenburg*, pp. 351–360, Springer, 2002.
9. L. Cavallaro, E. Di Nitto, C. Furia, and M. Pradella, “A tile-based approach for self-assembling service compositions,” in *Proceedings of the 15th IEEE International Conference on Engineering of Complex Computer Systems*, Citeseer, 2010.
10. M. Strohbach, G. Kortuem, and H. Gellersen, “Cooperative artefacts - a framework for embedding knowledge in real world objects,” in *In Smart Object Systems Workshop at UbiComp*, 2005.
11. M. Strohbach, H. werner Gellersen, G. Kortuem, and C. Kray, “Cooperative artefacts: Assessing real world situations with embedded technology,” in *In Ubicomp*, pp. 250–267, Springer, 2004.
12. A. Sinha and P. Couderc, “Using owl ontologies for selective waste sorting and recycling,” in *OWLED*, 2012.
13. F. Michahelles, *Innovative Application Development for Ubiquitous and Wearable Computing*. Phd (dr. sc. techn.) dissertation, ETH Zurich, Perceptual Computing & Computer Vision Group, Institute for Scientific Computing, ETH Zurich, Switzerland, Dec. 2004.
14. M. Perttunen, J. Riekki, and O. Lassila, “Context representation and reasoning in pervasive computing: a review,” *International Journal of Multimedia and Ubiquitous Engineering*, pp. 1–28, Oct. 2009.
15. D. Preuveneers and Y. Berbers, “Encoding semantic awareness in resource-constrained devices,” *Intelligent Systems, IEEE*, vol. 23, pp. 26 –33, march-april 2008.
16. F. Siegemund and C. Flörkemeier, “Interaction in pervasive computing settings using bluetooth-enabled active tags and passive rfid technology together with mobile phones,” in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, PERCOM ’03, (Washington, DC, USA), pp. 378–, IEEE Computer Society, 2003.