International Journal of Multimedia and Ubiquitous Engineering
Vol. 4, No. 4, October, 2009

Context Representation and Reasoning in Pervasive Computing: a Review

Mikko Perttunen, Jukka Riekki
Department of Electrical and Information Engineering and Infotech Oulu
P.O.BOX 4500, 90014 University of Oulu, Finland
{mikko.perttunen, jukka.riekki}@ee.oulu.fi

Ora Lassila
Nokia Services
5 Wayside Rd, Burlington
MA 01803, USA
ora.lassila@nokia.com

Abstract

Context-awareness has been recognized as an important enabler for pervasive computing. Our aim is to provide a thorough survey of the state of the art in context representation and reasoning in the field of pervasive and context-aware computing. The review is organized in sections according to the knowledge representation techniques applied in the work included in the survey and an overview of the requirements for context representation is provided. We conclude the paper with a discussion of a number of identified limitations in the current solutions and point out opportunities for further research.

Keywords: *knowledge representation, ubiquitous computing, context model*

1. Introduction and scope

Pervasive computing is a vision of personal computing where future living environments are permeated with unintrusive, seamlessly operating services readily available for the user [1]. To fully realize this vision these services need to adapt to the current situation of the environment, including the social situation of the user. Systems that utilize information about the situation of either its users, the environment, or the state of the system itself to adapt their behavior are called context-aware systems [2, 3].

To scope the paper, we note that realizing context-awareness has proven to be a difficult problem on many levels: First, defining what constitutes context information has been studied extensively, and several definitions have been suggested [3-9]. Some definitions approach defining context as an abstract concept while others are rooted in the desire to be able to technically represent context information. This difficulty of definition overarches all research in context-awareness. Second, what and how should be adapted when the context changes and where do the context definitions and adaptation rules come from? These human-computer interaction problems are at least as challenging as the purely technical problems related to realizing context-awareness. Third, recognizing different contexts from sensor measurements and other information constitutes a large part of the research. *Fourth, how to represent and process contexts, the related other knowledge, and adaptation rules*? This technical problem has received considerable research attention and is what this review focuses on.

Before focusing onto the fourth problem, we need to give an overview of work on the first problem; defining the meaning of context and constituents of context information have been recognized as difficult problems and context (sometimes "situation") has been given a

number of definitions; for examples of conceptual definitions see [3, 6]. Exemplars of more technical definitions include [4, 5, 7-9]. By technical we mean that the definition can be easily applied to a practical context representation (encoding) and provides means to distinguish contexts from each other. By conceptual we mean defining what constitutes context, that is, what needs to be encoded [10]. In the scope of this paper we apply perhaps the most cited definition by Dey [3]:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

The fourth problem, and this literature review, focuses on how context information and the related knowledge are *represented* and *processed* in state-of-the-art context-aware systems. Figure 1 illustrates three levels of context information processing with typically used methods and techniques in each layer. The arrows depict the flow of context information. On the *bottom layer* signal processing and machine learning techniques are used to recognize contexts and activities from sensor signals and other information.

The *middle layer* is the core of context modeling, where a multitude of representation and reasoning techniques have been applied. The highly dynamic outputs of the bottom layer puts hard demands on the middle layer.

The *top layer* deals with firing the context-dependent actions or adaptation. Applications use a context query language (CQL) [11] to access context information from context storages or providers. Query languages can be used to describe queries as well as subscriptions. The design, or choice, of the query language depends on the representation techniques used in the middle layer. The meaning of the queries must be well-specified because in the implementation the queries are mapped to the representations used in the middle layer. An important role of the middle layer and the query language is to eliminate direct linking of the context providing components to context consuming components. An early form of this was Dey's context widgets [12]. This abstraction is important due to distribution and mobility (and often the ad hoc nature of networks) which may cause producers to appear and disappear dynamically [13]. Thus, the query language should support querying a context value regardless of its source. However, knowing the source of a value may be useful for the client in the case of finding the cause of an erroneous inference, for example, and can thus be included in the query response. It should be noted that since the CQL acts as a facade for the applications to the underlying context representation, the context information requirements of the applications are imposed as much on the query language as on the context representation and context sources. For detailed reviews of CQLs, see [11, 14]. Procedural programming is typically used on the top layer to create queries and to handle query responses, adapting the application according to context. In contrast, the context representation in the mid layer can be purely declarative.

Considering the query languages, context-aware pervasive computing systems come close to the area of sensor networks, as query processing can be implemented also in a distributed fashion, without an actual database. Henricksen and Robinson [15] state that sophistication of supported queries characterizes the difference between the middlewares for sensor networks and context-aware systems; sensor networks typically provide means for trivial operators

such as *min*, *max*, and *average*, whereas the middleware for context-aware systems allow higher level queries.

In addition to context representation, the fourth problem considers processing context information. As knowledge representation techniques usually have an associated *reasoning* method, some of the context processing may be allocated to these default reasoning mechanisms. This means, for example, that stating a current user context explicitly may entail that the user *is in* other contexts that are determined through reasoning. However, it is well known that representation and reasoning are inherently tied together [16] and thus any expressive representation is computationally inefficient. Inference algorithms can be generally described using concepts *soundness* and *completeness*. As defined in [17], "sound inference algorithms derive only sentences that are entailed", and "complete algorithms derive all sentences that are entailed". If efficiency is our paramount requirement, then requirements on (some of) soundness, completeness, or expressiveness have to be relaxed. This survey focuses on the context representations, addressing computational characteristics of the associated reasoning methods, and the CQLs only in a general fashion.

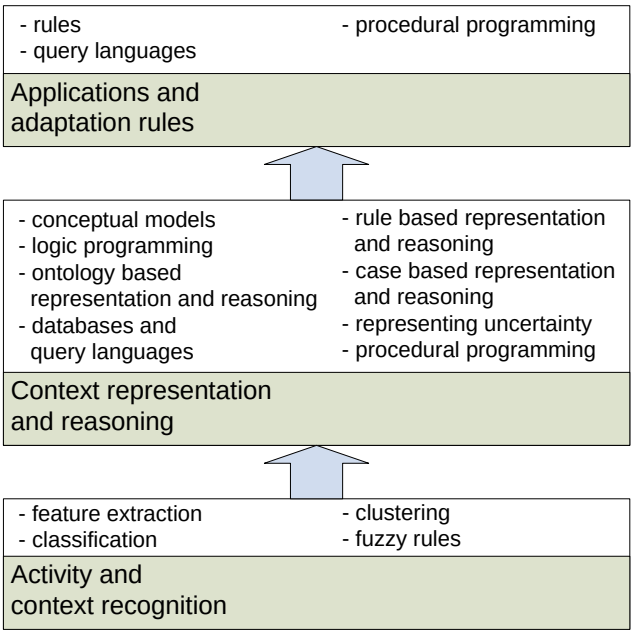

Figure 1. Layers of context information processing with examples of typical methods and techniques for each layer.

Previous reviews of this topic have had somewhat different points of views than ours: Korpipää's thesis contains a general review of information management and related architectures for context-aware systems [18]. Strang and Linnhoff-Popien propose a set of general requirements for representing context information and compare different approaches in the light of those requirements [19]. For a survey on using context as an organizing concept in problem solving in AI and other related disciplines, see, for example [20].

We omit from the review detailed discussion of activity and context recognition (see Figure 1). Instead, the focus of the review is on processing higher level context knowledge

with formal representations deriving from Knowledge Representation and Reasoning (KR&R), a subfield of Artificial Intelligence (AI). Most of the work in context representation falls into this category. Nevertheless, other relevant representation techniques from the literature of context-aware computing are included as well. We consider context primarily from the view point of being an enabler for adaptability in human-computer interaction as in [3]. Thus, we exclude reviewing the more traditional view of dealing with context at the level of a logical theory, as in the work of McCarthy [21], that is, considering the truth of a formula in different contexts. The distinction is not sharp, however, and this will become evident throughout this survey.

We aim to give an overview of the knowledge representation techniques used and to discuss their known limitations. In this paper we contribute a distilled discussion of the evolving field of context representation and reasoning as well as suggest directions for future work.

This paper proceeds as follows. We synthesize a set of requirements for context representation and reasoning from related work in section 2. Section 3 is organized as subsections according to applied knowledge representation techniques. The survey findings are discussed in section 4. We present our conclusions in section 5.

2. Requirements for context representation and reasoning

Because context itself is quite a fluid concept it is hardly the case that a comprehensive set of requirements for context representation has been, or can be put forward. However, in this section we aim to synthesize a set of requirements for context representation and reasoning from the following sources: We combine our base set of requirements from [18] and [19]. This is done to have a common ground to compare the reviewed work against, even when the work at hand did not explicitly specify which requirements it targets.

As expressiveness and efficiency are required in [18], we also incorporate the generic KR&R concepts soundness and completeness [17] to be better able to discuss all of these properties. Since the requirements and features from these different sources have some overlap, we have to some extent unified them to come up with a representative collection. In the following we present only a selection of requirements and features from the above sources, based on our subjective opinion of their relevancy. We suggest the reader turn to the original sources for more complete information.

2.1. Representation

2.1.1. Unique identifiers: Although this requirement seems self-evident, it is so fundamental to distributed systems that it should be explicitly mentioned. That is, for the system to be able to uniquely identify different contexts as well as entities in the real world domains the system deals with, unique identifiers in some scale are necessary. Furthermore, uniqueness enables the reuse of the representations without conflicts in identifiers.

2.1.2. Validation: A context representation should allow validating pieces of data (or knowledge) against it [19]. This enables software components to ensure that data is at least consistent with its schema before performing any processing with it.

2.1.3. Expressiveness: An expressive representation allows representing complex entities and relations, but is in mutual conflict with soundness, completeness, and efficiency of

reasoning [16] (see section 2.2.). Korpipää specifies efficiency and expressiveness as requirements for his context representation [18].

2.1.4. Uncertainty and incomplete information: Since much of context information is measured from the real world through imprecise sensors, the representation should allow encoding uncertainty of the values. If reasoning is involved, uncertainty of conclusions should follow from antecedents. Furthermore, the system should be able to deal with incomplete contextual information. Strang et al. [19] refer to the ability to represent "richness and quality" of context information as well as to deal with "incompleteness and ambiguity".

2.1.5. Simplicity, reuse, and expandability: In contrast to the demand for expressiveness mentioned in 2.1.3, a system should apply only as expressive representation as necessary to encode the domain knowledge. A simple representation promotes reuse and expandability. Simplicity, flexibility and expandability are among the requirements of context representation of Korpipää [18].

2.1.6. Generality: Generality of context representation refers to its ability to support all kinds of context information [18]. We think generality of a context representation is largely defined by its conceptual structure. Nevertheless, the expressiveness of a representation language used to encode context information also affects its ability to encode context information at different levels of complexity.

2.2. Reasoning

2.2.1. Efficiency, soundness, and completeness: Since context information is highly volatile, the ability to deal with dynamic knowledge updates is necessary for the context representation and reasoning system. Using the most expressive system that provides sound, complete, and efficient-enough reasoning under dynamic knowledge base updates is desirable.

2.2.2. Multiple reasoning methods: Korpipää considers it important that multiple reasoning methods can be used with a representation [18]. This almost always implies that some of the other requirements for reasoning must be loosened.

2.2.3. Interoperability: For comprehensive interoperability, contexts should be represented in a syntactically and semantically interoperable format to allow sharing and reuse of representations. The loosely coupled components of a context-aware pervasive computing system should conform to a common representation for message exchange. Moreover, the reasoning procedures should be standardized to ensure that different implementations of the procedures produce the same results. In other words, evaluated against the same set of axioms, a set of assertions should always produce the same conclusions. This implies that when a set of assertions represents a message, its receiver can derive the exact meaning the sender had encoded in the message. As a related requirement, Strang et al. [19] point out the need of "level of formality" for the representation.

2.3. Analyzing the requirements

The aim of this section is to give an insight to the dependencies between the requirements in order to promote understanding of the problems in representing context knowledge.

Deriving the requirements for generic context-aware systems' knowledge representation and reasoning is difficult. Since the system should support applications which are not even known at system design-time, it is scarcely the case that the requirements for KR&R can be precisely derived. Due to this inherent problem the design of the KR&R system necessarily relies on vague and general requirements. A straightforward way to approach the above-described situation is to design for the "average", that is, derive requirements from a typical application.

Due to the issue of generality of the requirements described above, it is difficult to know in which sense we want to improve the expressiveness. For example, in one application it may be important to be able to say "any of my friends" (existential quantification), while another application might benefit more from being able to deal with uncertainty.

Moreover, special purpose procedural attachments [16] (pp. 138) can sometimes largely reduce inference costs when compared to a generic inference mechanism. An important sub-requirement for context representation, related to both expressiveness and efficiency, is the support for retraction. Since contexts are usually recognized based on sensor measurements and new measurements may replace previous values, the previous measurements and the contexts derived based on the previous measurements should be retracted, as well as the context derivation repeated. This generally requires computing every conclusion again. A way to prevent some of the computation in rule-based systems is truth maintenance [22]. The cost is incurred as larger memory consumption.

Simplicity refers to the ease-of-use and is somewhat conflicting with expressiveness. Especially, from the system design point of view it intuitively seems easier to encode the knowledge needed by a simple application in a less expressive, but simple representation than in an overly expressive representation, not providing real benefit for the particular application being designed. This is a trade-off that has to be made in favor of more complex applications utilizing the framework.

A system may incorporate multiple associated inference methods operating on its context representation. For example, the system described in [18] supports this. The fact that in such systems the same representation has multiple interpretations and that the different semantics are necessarily encoded in the individual reasoners hinders interoperability. Of course, often there are many implementations of reasoners for a representation, all sound and complete. In this case only the computational requirements (space, time) vary, while the resulting conclusion set is identical.

The congruency of inference conclusions is a basic requirement for interoperability. Nonetheless, this also implies a disadvantage: it strengthens "ontological commitment" [23], i.e., the more consequences are encoded as axioms in the representation, the more its clients are tied to dealing with the represented entities in the exact same manner. This is undesirable when only a few of the entities of the representation are of interest to the client. An example of how this problem can be tackled is the reuse of modules (instead of the whole ontology) of a Web Ontology Language (OWL) [24] ontology described in [25].

Having introduced context information and synthesized a set of requirements, in the next section we move onto reviewing the literature on context representation.

3. Context models and reasoning

Winograd [10] notes that the context representation problem has two sides:

> *The hard part of this design will be the conceptual structure, not the encoding. Once we understand what needs to be encoded, it is relatively straightforward to put it into data structures, data bases, etc.. The hard part will be coming up with conceptual structures that are broad enough to handle all of the different kinds of context, sophisticated enough to make the needed distinctions, and simple enough to provide a practical base for programming.*

As implied by the above quote, the context representation problem has two sides. We separate these as we go through the work in the following subsections. Since the focus of the recent research has been on the "encoding" (representation) of knowledge, we cover more of that work. To make the distinction clear, from here on we use the term *context model* analogously to a conceptual model. Respectively, we use the term *context representation* to refer to a context model represented using a particular knowledge representation technique.

This section consists of subsections corresponding to the different knowledge representation techniques applied in the reviewed literature. Section 3.1 describes conceptual models that deal with understanding what constitutes context information and the features and categories of contexts. In section 3.2 we review work that applies logic programming. Section 3.3 analyzes work applying ontologies and rules to representing context. Section 3.4 deals with systems using case-based reasoning. In section 3.5 we collect together work dealing with uncertainty of context information from all the above categories.

3.1. Conceptual models and qualities of context information

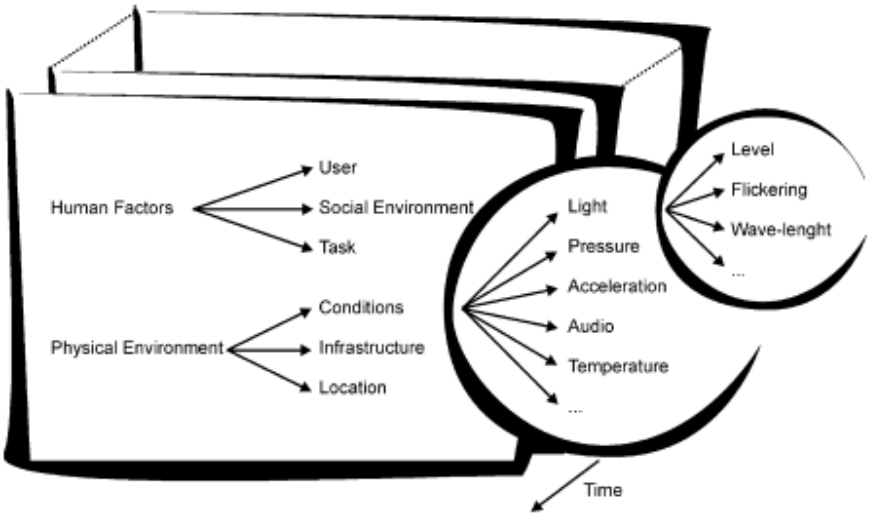

Figure 2. Context feature space[1].

[1] Reprinted from Computers & Graphics, Vol. 23, 6, A. Schmidt, M. Beigl & H. Gellersen, "There is more to context than location", pp. 893-901, Copyright (1999), with permission from Elsevier.

By conceptual models we refer to models that deal with what constitutes context and conceptual structure of context. In addition, in this section we consider work that identifies qualitative features of context information. Some of the reviewed work does not provide encodings of the proposed models using any knowledge representation technique.

One of the first attempts to explicitly model context information in context-aware computing is by Schmidt et al. [4]. Their approach is to model context using features, so that for each context there is a set of relevant features, where for each feature a value range is defined. Figure 2 (from [4]) illustrates this conceptual model. For example, physical environment, conditions, light, and level constitutes a path where level is the leaf node and has a concrete value.

Korpipää et al. [26] present a context model based on that of Schmidt et al. [4]. The properties of their context structure are context type, context value, confidence, source, timestamp, and attributes. We come back to the encoding of this work in section 3.3.

Gray and Salber [6] focus on dealing with the aspects of sensed context, as opposed to context information in general, which constitutes also of static information like user identity. Most of their work concerns what they call the meta-attributes of sensed context information. These are *forms of representation, information quality, sensory source, interpretation*, and *actuation*. The dominant attribute is information quality, which constitutes of *coverage, resolution, accuracy, repeatability, frequency,* and *timeliness*. For detailed descriptions of the attributes the reader is directed to the original paper. The authors do not propose a solution to a particular problem, but rather discuss how the features of sensed context should be taken into account in a design process.

Perhaps the single most influential piece of work in the area of context-aware computing has been the Context Toolkit by Dey et al. [12]. The toolkit is based on a conceptual framework consisting of context widgets, interpreters, aggregators, services, and discoverers. In the framework widgets collect context information and interpreters process the information to provide more abstract context information. Aggregators assemble information that concerns a certain entity. Utilizing the context information, services can perform actions on the environment. Discoverers are used by applications to find the other components in the environment. Context representation in this conceptual framework is embedded into its components, mostly widgets. The authors introduce four categories of context information related to entities: *identity, location, status* (or *activity*), and *time*.

In [27] the authors present a context model for an agent based middleware architecture. The context model is based on earlier work of one of the authors [28]. In the model, environmental state descriptions based on entities and their properties are called situations. In the situation model, states are connected with transitions. The transitions can be caused by changes in observed entities' properties. To generalize the situations, they are described using generic roles and the entities are mapped to the roles according to their current property values. Moreover, situation descriptions contain relations that can be, for instance, Boolean comparison operator between values of a specific property of two or more entities. The authors note that this context model may seem not to be scalable, because the situation states will hardly capture all possible contexts. They further claim, however, that the situation model can be dynamically extended.

Jang and Woo [29] propose a conceptual format of context messages. This format includes user identity (who), objects identity (what), location (where), time (when), user

intention/emotion (why), and user gesture (how). This message format is encoded as a text string.

Gastelli et al. [30] propose a model similar to Jang and Woo [29] above, called "W4" (who, what, where, and when). Their system stores context as tuples of the W4 format in tuple spaces. The W4 imposes a common base structure to the data, but the tuple contents are simple encoded as strings. The first part of tuples denotes its type and the seconds the value, for example, "person:Gabriella".

Riva [31] describes a context representation based on key-value pairs and an associated SQL-like query language. An example is the triplet <noise=medium, light=natural, activity=walking> which defines the context type "walking outside". Each such context item contain values for the type, timestamp, and the optional validity duration, source identifier, correctness, precision, accuracy, completeness, and levels of privacy and trust may be included.

3.2. Logic programming

The commonality in the works reviewed in this section is that the proposed context representations are based on a logic programming language. Some of them utilize other methods as well, but the above mentioned was considered an adequate reason for a paper to be classified under this category.

In a project called Gaia, Ranganathan and Campbell have developed a predicate logic representation of context information [32] based on logic programming using XSB [33]. For each context in their model, there is a first order predicate the name of which describes the context type. A typical example they use is "Location (chris, entering, room 3231)". In addition, logic operators, such as quantification, conjunction, negation, and disjunction can be used to combine the context predicates into more complex context descriptions. Quantification is always done over finite sets, for instance, the list of users of the system.

In addition to the above-described work, Ranganathan and Campbell have applied AI planning techniques to the Gaia system [34]. Namely, they propose the usage of STRIPS [16] (pp.312) planning. In a more recent work, in his thesis [35], Ranganathan points out that they considered planning computationally too expensive for their system and that they recognized most of the produced plans contained the same actions.

Henricksen and Indulska [36] propose a situation abstraction, which is based on predicate logic. In the expressions of their representation they allow and, or, and not logical connectives as well as quantification to be used. Quantification (similarly to Ranganathan and Campbell above) is allowed only over finite sets. The basic expressions equality, inequality, and "assertion" are supported. Assertion is used in the expressions of the situation abstraction to define the sets over which the quantification is done. The representation also incorporates ambiguity in the form of fact alternatives, and unknowns represented by null values in the database. The interpretation of the model is based on three-valued logic. An assertion evaluates to "possibly true" when replacing some constants in a partially matching database tuple with the null value causes it to match with the assertion. Similarly, assertions matching tuples containing alternative facts are given the third truth-value "possibly true". The model is interpreted under the closed-world assumption. A situation S can be defined as $S(v_1, \ldots, v_n) : \varphi$, where φ is a logical expression with free variables from $v_1, \ldots, v_n$. An example situation is shown in Figure 3 (from [36]).

Loke [8] proposes representing contextual information as Prolog-style logic programs [37]. He uses the term of sensors fall

```
if in_meeting_now(E) then
   with_someone_now(E),
   has_entry_for_meeting_in_diary(E).
if with_someone_now(E) then
   location*(E,L),people_in_room*(L,N), N > 1.
if has_entry_for_meeting_in_diary(E) then
   current_time*(T1),
   diary*(E,'meeting',entry(StartTime,Duration)),
   within_interval(T1, StartTime, Duration).
```

Figure 3. Definition of situation CanUseChannel[2].

inside specified ranges. In Loke's model a *situation program* defines the Prolog rules that relate a situation to the sensor readings. Sensor readings are represented as Prolog predicates. Since all situations are defined using rules, the representation also allows reasoning about the relations of the situation definitions, for example, whether a situation subsumes another situation. The example from [8] in Figure 4 defines the situation "in meeting now". To conserve space, we only note that in the example the predicate *location*(E,L)* binds the location of entity E in variable L.

The reasoning in [7] is also based on logic programming, but as the focus of the work is on ontologies, we review it more closely in the next section. Whereas the results of executing rule-based programs depend on various rule execution strategies, the ontological models reviewed in the following section are purely declarative.

$$
\begin{aligned}
&CanUseChannel(person, channel):\\
&\quad \forall device \bullet requires\_device[channel, device]\\
&\quad \bullet\ located\_near[person, device]\\
&\quad \wedge\ permitted\_to\_use[person, device]
\end{aligned}
$$

Figure 4. Definition of "in meeting now" context[3].

3.3. Ontology-based representation

By ontology one can mean a multitude of things from a taxonomy of terms to a logically sound representation and reasoning method [38]. In this section we review work on rule-based and ontology-based representation and reasoning about context. Thus, we don't restrict the reviewed work based on the type of ontologies. As will become clear, most of the reviewed ontology-based work involves rule-based reasoning only, or in addition to the inferences encoded in the ontology axioms. Production rules are the essence of expert systems [39].

Most of the ontology based work in this category applies description logics [40]. Description logic (DL) knowledge bases consist of two components, the TBox and the ABox. The TBox contains the terminology of the application domain and the ABox assertions about

[2] Reprinted from Pervasive and Mobile Computing, Vol. 2, 1, K. Henricksen & J. Indulska, "Developing context-aware pervasive computing applications: Models and approach", pp. 37-64 Copyright (2006), with permission from Elsevier.

[3] Reprinted from S.W. Loke, "Logic Programming for Context-Aware Pervasive Computing: Language Support, Characterizing Situations, and Integration with the Web", in proc: IEEE/WIC/ACM International Conference on Web Intelligence, pp. 44-50, 2004., © 2004 IEEE.

individuals, using the terminology of the TBox. DL languages consist of three key constructs: concepts (classes) that represent classes of objects, roles that describe binary relations between concepts, and individuals, which represent instances of classes. An important feature in description logics is defining a concept in terms of necessary and sufficient conditions based on other concepts (and roles) [40] (pp. 13).

The basic operation of description logic reasoners is subsumption determination, i.e. checking whether a concept is more general than the other. Key inference operations with respect to ABox are *realization*, which means "determining the concepts instantiated by a given individual" and *retrieval*, which, in turn, means "determining the set of individuals that instantiate a given concept" [40] (pp. 310).

Gruber defines ontology as "an explicit specification of a conceptualization" [41]. Studer et al. define ontology more precisely [23]:

> "*An ontology is a formal, explicit specification of a shared conceptualisation. A 'conceptualisation' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. For example, in medical domains, the concepts are diseases and symptoms, the relations between them are causal and a constraint is that a disease cannot cause itself. 'Formal' refers to the fact that the ontology should be machine readable, which excludes natural language. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group.*"

The Cobra project [42] proposes OWL [24] to represent context information. Their work composes of designing OWL ontologies [43] and using those in a smart meeting room scenario. Chen and others note that their system does reasoning both based on the axioms in the ontologies and utilizing additional rule-based reasoning with arbitrary RDF [44] triples. The former is done using Jena's [45] rule-based OWL reasoning engine and the latter by applying Java Expert System Shell (Jess) [46]. They give no details of which OWL reasoning services they use, but say that the system uses Jess rules only when required, that is, when the OWL reasoning is not able to produce the answer to a query. The mechanism for detecting when OWL reasoning is not enough is not described. However, in this case, the system is said to be able to query the ontology reasoner to find all relevant supporting facts, and to convert the resulting RDF graph(s) into a Jess representation. A forward-chaining procedure is executed in Jess and any new facts are converted back to RDF and asserted to the ontology reasoner.

Gu et al. [47] have also modeled context by using OWL ontology language. They designed a context ontology as two layers: the upper ontology and the domain ontology. The upper ontology contains ContextEntity as the root element and its children are CompEntity, Activity, Location, and Person. Their approach is to represent contexts in first-order predicate logic, using the terminology defined in the ontology in the first-order expressions. However, they state that their system supports reasoning tasks based on RDF Schema and OWL Lite axioms, as well as rule-based reasoning with arbitrary RDF triples. That is, the reasoning is not based on first-order logic. The system is implemented using Jena [45], which indeed provides these capabilities.

The Semantic Space project [48] uses OWL to represent context information as well. They argue that the advantages of using ontological approach in representing context are, for

instance, shared understanding of the semantics of context representations, ontology-based context reasoning, knowledge reuse by using existing ontologies. The listed techniques are analogous to standard ontological engineering methods [38] and as such not specific to representing context. The ontology used in Semantic Space is almost identical to that of Gu et al. above, the works sharing one common author. In Semantic Space, the applications can pose queries in RDF Data Query Language (RDQL) [49] to the knowledge base. Their system incorporates a simple way of avoiding conflicts, when application-specific rule-based reasoning is used; the applications send the forward-chaining rules to the reasoner and the reasoner returns the results of the execution to the application, without storing them to the knowledge base. Implementation is done using Jena [45].

A similar approach is implemented in Nicklas et al. [50]: an application specific context ontology is crafted in OWL and rule-based reasoning is implemented using Jena generic rule engine [45]. Thus, no DL reasoning is used.

A commonality between the above research efforts is that the developed ontologies, to the extent presented, are generally based on taxonomic top-down hierarchies of the domain elements. A different approach is used in [51] and in [7].

In the former, Khedr and Karmouch propose dividing the model into "levels of expressiveness". In addition, they divide their model into two ontologies that they call relational ontology and dependency ontology. The relational ontology is conventional domain ontology, whereas the dependency ontology is an attempt to represent the parameters of inference rules as ontology classes and their properties. Inference in the system is based on rules, that is, not on the inferences licensed by the ontologies. The ontologies are implemented in OWL and reasoning is done using Jena [45]. No examples of inference rules are given.

In the latter, Strang and others describe an ontology that they call an aspect-scale-context model, according to its main concepts. There is a bi-directional relation between aspect and scale, as well as between scale and context information. As an example of these relations, the authors give the aspect "GeographicCoordinateAspect", which may have two related scales: "WGS84Scale" and "GaussKruegerScale". In this example, valid context information could be an instance of "GaussKruegerScale" with the appropriate instance attributes. As said before, the ontology of Strang et al. is similar in style to that of Khedr and Karmouch. Moreover, reasoning in the ontology is also based on dedicated logic programming rules. The ontology of Strang et al. and rules can be used to, for example, check that a context instance is consistent with its scale. These ontologies do not facilitate interoperability between systems if the associated rule systems are not also used.

In [9] Lassila and Khushraj consider representing context using description logics (DL) [3], where OWL DL is used as a concrete example. One of the ideas behind their context model is that entities in a context-aware system can "be" in any number of contexts at a time. In their ontology, contexts are represented as classes and the current contextual information about an entity is represented as an individual. Their system uses both retrieval and realization. They also consider specifying new contexts in terms of existing context classes using generic DL class constructors. As an example, a context could be defined as an intersection of existing contexts.

An example (adapted from [9]) shown in Figure 5 presents a walkthrough of ontology-based reasoning. The order of reasoning does not necessarily follow that of a practical

reasoner; instead the purpose is to illustrate how the context definitions represented in an OWL DL ontology are used in the inference.

Lassila and Khushraj point out that a problem in representing context with DLs is the known lack of composition constructor for properties [52, 53]. For example, the composition shown in Figure 6 (expressed in a rule format where variables are denoted by question marks) cannot be expressed.

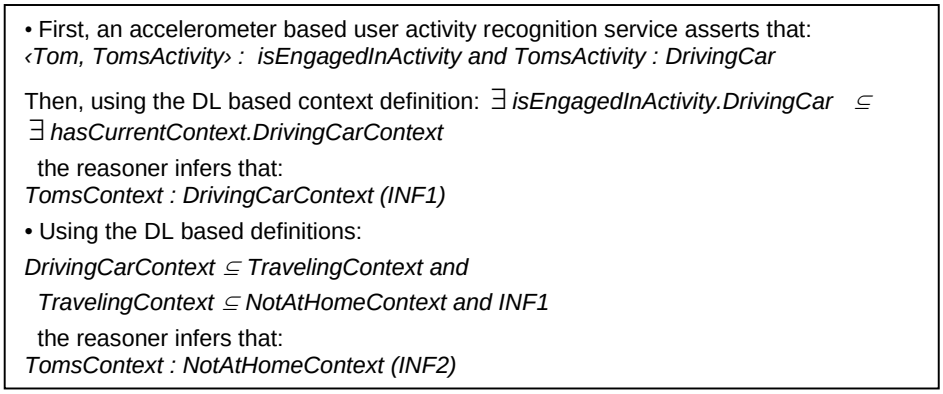

Figure 5. An example of ontology-based reasoning[4].

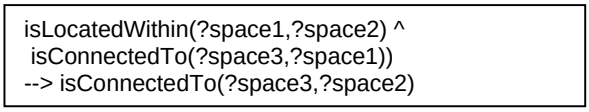

Figure 6. An example of composition construct[4].

As another weakness, [9] remarks the lack of procedural attachments. To solve these problems, the authors use a hybrid reasoning process, where they add rule-based reasoning on top of DL reasoning. This architecture is depicted in Figure 7 (from [9]). They present an algorithm that specifies the integration of a DL reasoner and a rule-based reasoner. While Lassila and Khushraj describe their algorithm from DL point of view, by considering the inverse, their system is close to the idea of using DL system as the working memory of a rule-based system as described in [16] (pp. 181).

In a similar approach to that of [9], Agostini et al. suggest combining rule-based reasoning with DL reasoning [54]. They propose OWL DL ontology for context representation, but represent user profiles separately using Composite Capability/Preference Profiles (CC/PP) [55]. Certain attributes in the profiles are linked to concepts in the ontology for more formal specification. Referring to performance issues, the authors suggest that most DL reasoning should be performed offline, not when services are provisioned to user. The DL reasoning task they mainly consider is querying the instances of a particular class. They note that a key feature of their reasoning approach is that results of executing the rule-system are not stored to the ABox. This way the data flow is unidirectional; from the ABox to the rule-based system.

[4] Reprinted from O. Lassila & D. Khushraj, "Contextualizing Applications via Semantic Middleware", in proc. of The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, San Diego, USA, pp. 183-189, © 2005 IEEE.

Khushraj et al. [56] propose an extension of tuple spaces to integrate ontology-based reasoning to query processing. Tuples describe services and generic data items; the work does not propose a specific context model. SUN's object oriented tuple space called JavaSpace [57] is used to implement the system. Integration of tuple spaces and ontology-based representation is done through enforcing an object field in every tuple to contain a DAML+OIL individual [58]. Furthermore, the consistency of the DAML+OIL individuals from newly written tuples is checked by a reasoner before they are committed to the knowledge base. Querying is done using a special query format (query template) which essentially controls what queries are send to the reasoner. A special matcher agent performs the queries and combines the query results according to a tailored algorithm.

Yet another work that is based on OWL DL reasoning is reported in [59]. The proactive service recommendation system for internet services retrieves a set of relevant tasks for the mobile user based on his current context. The task model and related tools are first described in [60]. The OWL based context representation uses classification (realization) to compute the current context of the user. The model includes qualitative descriptions of time, location, as well as relations among people. The tasks in the system are described using the OWL-S process model [61]. The system retrieves a set of predefined high-level tasks (e.g. "go to destination") by matching context tags of the tasks to the current context of the user, and lets the user browse them through a mobile device. The high-level tasks have a hierarchy of alternative subtasks (e.g. move-by-taxi, move-by-train) and finally the leaves of the task tree represent mobile services the user can utilize through his mobile's web browser.

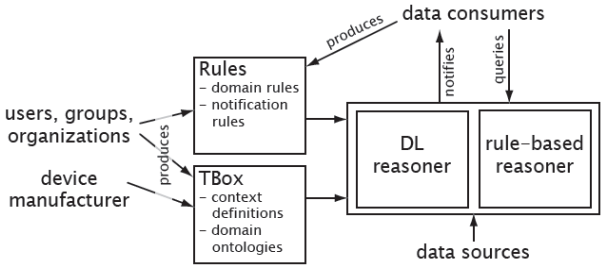

Figure 7. Context reasoning architecture[4].

W3C has begun to develop an ontology for context-based adaptation of web content [62]. The ontology focuses on describing the device, for example its display and audio support. The specification as of now does not give any reasoning examples, and it is thus unclear to what extent these capabilities of OWL DL are going to be used.

Bobillo et al. [63] divides knowledge related to the problem at hand into two ontologies: domain ontology and context ontology. As usual, the domain ontology contains the entities, relations and individuals of the domain being modeled. Context ontology is used to describe the setting where the domain ontology is used and consists of sensor data and user preferences, for example. In addition to these base ontologies, they define a Context-Domain Relevance (CDR) ontology. The CDR can be used to derive context-dependent, that is, relevant knowledge from the domain ontology.

Chaari et al. [64] utilize ontologies to represent context and using rules to trigger adaptation. Their approach aims to use the expressive combination of OWL and Semantic Web Rule Language (SWRL) [53]. However, described implementation is based on Jena generic rule engine [45], while it is not possible to fully translate SWRL rules into a rule based reasoner [53].

Ranganathan and Campbell [32] utilize ontologies in the logic programming system to define allowed types for the arguments of the predicates. The ontology is implemented in DAML+OIL [58] ontology language. The authors claim that they use FACT reasoning engine to ensure the validity of context expressions. However, they give no further detail of the reasoning tasks through which this is accomplished.

Korpipää et al. [26] present a work on developing a *lightweight ontology* for mobile device context-awareness. A lightweight ontology is an ontology that contains concepts and properties, as well as concept taxonomies, but no constraints [38]. The structure of the context model is based on that of Schmidt et al. [4] discussed in section 3.1. What makes this work to stand out form the rest is that the structure is represented using RDF, and a set of properties for characterizing the context expressions is defined. These properties are context type, context value, confidence, source, timestamp, and attributes. Since the ontology serves here mostly as a common vocabulary, any inference methods can be used, as long as the results are provided in the common format.

The system of Van Kleek [65] and Shrobe uses RDF(S) to represent sensor information and derived user activities. As opposed to most reviewed systems, their system is geared towards long-term collection and storage of this information.

3.4. Case-based representation

In case-based reasoning (CBR) data (cases) is represented as points in Euclidean space or as symbolic descriptions [66]. The symbolic descriptions may consist of complex graphs, for instance. Therefore, a key issue is finding relevant (often approximate) similarity measures for the cases. Aamodt [67] describes a typical CBR process; first, a case or a set of most similar cases is retrieved. Second, use the retrieved case to solve the current problem. Third, revise the solution. Fourth, retain the experience of this case to the extent it is likely to be applicable for solving future problems. Generally, CBR is seen as a process of finding problem solutions attached to previous similar cases and reusing those to solve the current problem [68]. The papers in this section are reviewed according to the above characterizations of CBR.

Turner's [5] context modeling approach is called context-mediated behavior, where context information is represented using contextual schemas (c-schemas). The c-schemas are similar to frames [69], containing slots, among others, for the features that *must* be present in the current situation for it to be recognized as an instance of context the c-schema represents. Features that are *likely* to be present in the context the schema represents are also included. These allow the agent to predict the state of the environment beyond its sensory inputs. Analogously to CBR cases' problem solutions, the c-schemas also contain information about how the agent should behave in the context. An exemplar of this is that the current context may affect the interpretation of sensor inputs. C-schema indices relate the schema to other schemas, typically its specializations [70]. Overall, the schema memory structure is similar to a conventional case memory [67]. Turner's system implements C-schemas using common lisp object system.

In a process similar to CBR's situation assessment [68], which Turner names context assessment, the features of the current situation are used to retrieve a set of c-schemas from schema memory. Subsequently, this set is reduced by comparing the degree of match between the current situation and each c-schema, ranking them accordingly. Each of the final c-schemas represents a context, which, in turn, partly describes the current situation. After this, these c-schemas are merged into what is called a *context object* [5]. The author remarks that these processes have not been fully implemented.

Zimmermann [71] proposes CBR for context representation as well and discusses it by considering a case with four contextual dimensions: location, identity, time, and environment/activity. The case is attached to a recommendation, analogous to the problem solution in CBR. The recommendation can be, for instance, delivered to a person that is in a museum near a painting that is from the artistic period he likes, as defined in his user profile. Zimmermann also discusses generalizing cases as well as modeling episodes of user activity with CBR. No system implementation is described.

Kofod-Petersen and Aamodt [72] discusses early work on a similar CBR-based context modeling system. The basic idea behind the CBR model is to associate a probable user task or goal with the user's current situation, that is, each case contains a task or a goal. They model high-level context as a taxonomy and extend the taxonomy with a domain-specific multi-relational semantic network. Generalizing cases is used as well, and an example case with a matching generalized case is presented. However, implementation and evaluation of the system is yet to be done.

3.5. Representing uncertainty and vagueness

As the contextual information in context-aware computing often originates from sensors in the environment, uncertainty in context-aware systems is unavoidable. Despite this, there is only a handful of work that deals with representing and reasoning under uncertainty in these systems.

Ranganathan and others describe reasoning in the Gaia system with vague and uncertain information [73]. Their model is based on assigning a confidence value for each context predicate (c.f. section 3.2) and interpreting the confidence as either a membership value in fuzzy logic or a probability in probabilistic logic. The authors do not discuss whether they somehow define the intended interpretation for predicates from these alternatives.

For probabilistic reasoning, Ranganathan and others apply a form of probabilistic predicate logic for which reasoning is known to be sound and complete. As an example of the source for probabilities in their environment they describe person localization with RFID badges; a probability that a detected badge is in a room is a result of dividing the badge's detection circle that is inside a room by the area of the whole circle. They also discuss other sources of uncertainty, such as whether the person is actually wearing the tag or not.

The authors also describe applying Bayesian networks for inferring activities from uncertain context information. In their network, leaves represent sensed context information and nodes information to be inferred. All the nodes are random variables of which possible values are context predicates. An example network for inferring the activity in a meeting room is given, where the values are "meeting", "presentation", and "idle". However, no probability tables in the network are given.

Gaia system utilizes confidence values also for access control. Here, the confidence values are simply compared against certain thresholds, such as in the rule shown in Figure 8 (example from [73]).

Loosely based on earlier work of Ding and Peng [74], Gu [75] presents a model where web ontology language OWL [24] is extended to allow representing dependency relations between properties. This is implemented by linking the properties with a rdfs:dependsOn property.

```
canAccess(P, display) :-
  confidenceLevel(authenticated(P), C), C > 0.7,
  Prob(activity(2401,cs 101 presentation), Y), Y > 0.8,
  possessRole(P, presenter)
```

Figure 8. Access control rule[5].

Their model also associates prior and conditional probabilities to RDF [44] triples, such as status(John, Sleeping)). The original work by Ding and Peng, considers extending OWL so that typical description logic (DL) reasoning tasks, for example, concept subsumption [40] are supported. However, algorithms for the reasoning tasks are yet to be developed. By using a distinctively simpler, syntactic approach, Gu et al. only consider a structural transformation from their extended OWL/RDF graphs into a Bayesian network, mapping the rdfs:dependsOn relations to arcs in the Bayesian network. They give no complete examples of the probability-extended OWL ontologies.

Similar work in modeling uncertain context information with Bayesian networks has been done by Truong and et al. [76]. They propose representing Bayesian networks in a relational model, where certain classes, called p-classes are used to store probabilistic information. Namely, p-classes' attributes have associated constraints: parents-constraint and conditional probability table (CPT) constraint. The former lists all the attributes in the parent classes that the attribute depends on. The latter is used to record the conditional probability distribution, which depends on the parents. The authors consider integrating their approach with representing domain knowledge in OWL, but no details are given.

Schmidt proposes time-based decay of validity of context information [77]. The decay function is context type specific, ranging for static (birthday) to quick decay (location).

Mäntyjärvi and Seppänen [78] apply fuzzy logic to represent contexts information. Fuzzy logic is manifested in vague predicates [16]. Through these vague predicates Mäntyjärvi and Seppänen represent concepts such as *user activity*, which can take values "movements", "walking", and "running", with soft borders. This work does not propose a specific encoding for communicating the fuzzy contexts and fuzzy rules, thus it is at the border of being a low level processing method (see Fig.1) or a context representation and reasoning scheme.

Similar to Mäntyjärvi and Seppänen, Padowitz et al. [79] represent context information as a simple multi-dimensional vector of sensor measurements. A context is defined as a range of values in this multi-dimensional space. Based on the context definitions and the current sensor measurements, they derive a confidence value to represent the uncertainty in the

[5] Reprinted from A. Ranganathan, J. Al-Muhtadi & R.H. Campbell, "Reasoning about uncertain contexts in pervasive computing environments", Pervasive Computing, Vol. 3, 2, pp. 62-70, © 2004 IEEE.

occurrence of a context (or a situation). The related reasoning engine replies to queries by encoding contexts and their confidences in XML.

Other examples of efforts in this category are the one discussed in section 3.1 by Gray and Salber [6] and the work by Henricksen and Indulska [36] referred to in section 3.2. Gray and Salber discuss the issue of quality of information in general, whereas Henricksen and Indulska describe their closed world interpretation of three-valued logics, where, the "possibly true" value is used to represent ambiguous or uncertain information.

3.6. Summary

The reviewed papers from section 3.1 through 3.5 are summarized in Table 1. Note that since some work contains aspects of more than one category, the columns of the table overlap. In the following we give an overview of the requirements each of the reviewed papers touches on. We do not to analyze the respective merits of the papers in achieving the goals, but instead aim to give the reader an overview of the extent to which each of the requirements have received research attention. It is interesting to find that some work focus

Table 1. A summary of reviewed papers. Columns list the categories and each category contains a list of papers included in the review.

Conceptual models	Logic programming	Ontology-based representation	Case based representation	Representing uncertainty and vagueness
[4], [26], [6], [12], [27], [28], [29], [30], [31]	[32], [34], [35], [36], [8], [7]	[42], [47], [48], [51], [7], [9], [54], [59], [63], [64], [32], [26]	[70], [5], [71], [72]	[73], [74], [75], [76], [6], [36], [78], [79], [77]

directly on one of the requirements, while other do not specify any particular goals in terms of requirements.

All work applying RDF or OWL naturally supports expressing *unique identifiers*. This work includes [7, 9, 26, 32, 42, 47, 48, 51, 54, 59, 63, 64]. Other work does not deal with unique identifiers. Generating unique identifiers and ensuring their uniqueness is an issue we do not consider here.

Ranganathan et al. designed a custom *validation* mechanism to validate rule parameters [73]. XML schema provides validation support in [27, 28]. OWL provides little support for validation.

Agents are used in several projects [27, 42, 80-82] on context-awareness, but there is no direct work on *interoperability* of context sources and consumers. The work using web ontology languages indirectly supports a form of interoperability, but much of the work rely on custom reasoning systems, thus compromising interoperability.

The interplay between *efficiency, expressiveness, soundness,* and *completeness* has not been studied with respect to finding the most suitable tradeoffs for context representation. We return to some possible future work in section 5.

Work dealing with *uncertainty* and *vagueness* was reviewed in section 3.5. The work in [6, 36, 73-76] deals with representing uncertainty. The work in [78, 79] proposes schemes to manage vagueness of context information.

Korpipää et al. proposes to use RDF to represent context information in order to support *simplicity* [26]. All ontology-based work support *expandability* and *reuse* somewhat better than, for example, rule-based systems reported in [7, 8, 32, 36].

Among the reviewed work [4, 7, 26] aim to support *generality* through using an overarching conceptual structure for context information as a basis of their model.

4. Discussion

This review has surveyed context modeling and representation describing several, often fundamentally different, approaches. We begin this section by listing some findings from the survey and bringing into focus a set of current challenges. At the end of the section we suggest some topics for further research.

4.1. Context representations

A generic finding about the knowledge representation in context-aware systems is that while utilizing context is considered the key factor in those systems, representing context is rarely distinguished from representing other knowledge. Rather, the problem of relevance is left for the applications that utilize the context information, for instance, the applications query only those elements of information they require. For example, in the Gaia system [32] a context may be "Location (chris, entering, room 3231)". The representation does not define that this fact is a context, or about a context, rather than a general fact about the domain. In the following we discuss a bit whether that would be useful or not.

Counter examples are the work in [9, 59] where contexts are represented explicitly. Thus, it can be asked whether it is beneficial to model contexts as "first-class objects", that is, distinguish contexts from other knowledge in context-aware computing. The key idea in [9, 59] is to represent contexts as (mostly named) concepts in a description logic. This corresponds to enumerating all contexts that are required by the system and defining them in terms of other domain knowledge or in terms of other contexts. Interestingly, this relates to some of the ideas in propositional logic of context as described by Serafini et al. in [83]. That is, contexts are modeled as "first-class objects", context objects can have relations to other domain objects, and context can have relations to other contexts (e.g. through generalization). Due to enabling the representation of these relations, it seems that "first-class" representation of contexts would be beneficial. However, a focused study would be needed to find and analyze the possible downsides.

An interesting feature of Turner's context model (c.f. section 4.4) is that the current context may affect the interpretation of sensor inputs. This causes a feedback loop in the context recognition process and it should be studied whether this is beneficial compared to conceptualizing inputs as evidence from which the contexts are determined. His suggestion that some concepts may have context-dependent meaning is analogous to the notion of context dependent adaptation of context-aware services. These aspects of Turner's model are close to the intuitions behind the efforts towards developing a formal notion of context in AI (for a concise review on formalizing context, see e.g. [83]).

Summarizing the reviewed work on modeling uncertainty and vagueness, it can be noted that there is no work presenting a model satisfying all the requirements listed in section 2. Moreover, it seems that the benefit of modeling uncertainty and vagueness has not been evaluated beyond the capability of representing it; that is, the work doesn't make it clear how easy it is to utilize such models in applications, what is the computational expense, and in what kind of applications does it benefit the users.

4.2. Reasoning about context

In this review, we included work on rule-based, logic programming, ontology-based (description logic), and case-based reasoning. As shown, majority of the recent work is applying an ontology-based approach. Indeed, context ontologies seem to provide some intuitive benefit for context-aware application development. In particular, as the context representation and reasoning of the system should be divided between *generic* and *application-specific*, the *generic* representation and reasoning can be encoded in the common ontologies, and the *application-specific*, in turn, in ontologies extending the common ontology and as rules. This is roughly the split shown in Figure 1, and analogous to the common way of designing an upper ontology and domain-specific ontologies [38]. Nevertheless, as we note in section 4.3 this, and other intuitive benefits of ontologies for context representation are hard to evaluate.

We observed that most of the recent work describing usage of OWL to represent context merely refer to using OWL inference, but the focus is on making inferences using an external rule-based system. There is an important distinction between inferences licensed by the ontology axioms and inferences based on arbitrary rules. In the former, any reasoner for that ontology language produces the same results, whereas in the latter both the ontology and the rules in the specific rule language are needed, possibly also an identical rule engine. For this reason, much of the benefit of using standard ontology languages is lost when inference is based on ad hoc rules, merely using the ontology terms as a vocabulary. Nevertheless, extending the reasoning beyond the inferences licensed by the ontology axioms is often necessary due to the fact that the expressive power of the ontology language is often insufficient for the task at hand. Current research tries to seek solutions to this by extending web ontology languages with rules [84].

In addition to the above considerations, there are many challenges associated with the specific representation methods. As ontology-based representations are the focus of much recent research, we note next some issues related to them that would deserve a more detailed analysis than given in the work described in section 3.3.

As an example, the OWL DL language is designed for monotonic inference. This means assertions cannot cause the truth of previous assertions or the previous conclusions to change. This does not fit modelling context, because usually the knowledge base (ABox) evolves with time. This is also true of many other applications that aim at a faithful representation of "the real world".

Related to the first example, context-awareness requires retraction of assertions. Consider, for example, keeping a proposition about the current room of a person up to date. Although some DL systems support retraction, typical description logics systems are optimized for query answering in domains with relatively static information rather than in domains with rapid changes in ABox information. Traditionally, this has meant that retractions (and additions) lead to expensive reclassification. Parsia et al. [85] report interesting ongoing work

towards reducing the cost of incremental additions and retractions by exploiting the monotonicity of DLs.

As a final ontology-related issue, any assertion can possibly make the DL knowledge base inconsistent, and an inconsistent knowledge base satisfies any query [86]. On the one hand, it is expensive to check consistency after each assertion, but on the other hand, it is better than not being able to detect inconsistency in the first place.

An intuition about the usage of predicate logic for context representation is that its expressiveness may cause it to be inefficient, because context knowledge is very dynamic. To the best of our knowledge, there are no results on comparing the efficiency of rule-based systems (e.g. Jess [46]), logic programming systems (e.g. XSB [33]), and more generic theorem provers (e.g. SNARK [87]) with respect to their capability to deal with dynamic context knowledge.

None of the (description) logic-based approaches discussed above are capable of dealing with uncertainty and vagueness. Some work (e.g. [11, 77]) approaches this by aiming to combine ontological modeling with modeling of uncertainty, but fall short in providing a principled approach that preserves the benefits of formal ontologies. Other work, such as the extension of Gaia described in [73], suggest a possibly feasible approach, but evaluation is lacking. As Ranganathan [73] notes, often the designer doesn't know the probabilities or confidence values to be associated with types of information. In these cases, learning them by collecting a large amount of data may be possible, but is a difficult task.

The only reviewed system geared towards handling large amounts of context information collected over time is presented in [65]. To give an approximate scale, the work reports about 300MB of RDF-represented data collected in 3 weeks for a person. This implies that complex reasoning over even a short history of contextual information may be infeasible.

5. Future work

We see improving of the evaluation methodology and facilities as by far the most important direction of future work for context representation. The lack of a common evaluation framework and results disabled us from performing any useful comparison between the reviewed systems. Devising a *generic evaluation framework* and metrics might help to focus research more on the core issues related to context representation and reasoning. At the same time, a way to improve the quality of research and to advance the state of the art would be to collaboratively specify a set of test contexts to be recognized and represented from a test data set. This would facilitate comparison of representation and reasoning systems. Hypothetically, given such test data, it should be possible to quantitatively compare the kind of work reviewed in section 3. The test data set should reflect the characteristics of context information processing, for example, the data flow from position and acceleration sensors and the query rate from clients. It is a major challenge to devise a representative test data set.

Many currently immature techniques may provide some aid for context-awareness in the future. As an example, fuzzy description logic systems [40] (pp. 246) could be used for representing partial appearance of a context. Any study on a new technique for context representation should investigate how the technique performs, using the common evaluation framework. This may help to understand the overall advantage of the new technique.

Sometimes it is possible to improve the efficiency of inference by using domain specific heuristics. In general this involves knowing, for instance, which queries are the most frequent and perhaps the most expensive. Using domain specific heuristics for designing optimizations may be achieved without altering the query answer. Moreover, incorporating special purpose algorithms or functions should be supported by the KR&R system. The user or the designer should be provided the means to input these special purpose functions. For example, in pervasive computing a location model is often important and many queries relate to locations of persons and devices. The classical example, "print to nearest printer", for instance, could sometimes (depending on the actual location of the nearest printer) be answered more efficiently if the search for answer could use the "query location" as an index [88]. That is, the inference engine should be capable of efficiently using relations that express the locations a certain location is connected to – in this case, propagating the search from the closest to the farthest location.

Brachman and Levesque call systems incorporating the previous kind of special purpose algorithms hybrid reasoning systems [16]. The idea of hybrid reasoning systems is to use special purpose reasoning always when it makes reasoning more efficient, making sure the reasoning stays sound and complete. Only when there is no special purpose algorithm for a predicate symbol does the system use generic reasoning, often a resolution algorithm. An example of a theorem prover providing a possibility for using special purpose procedures is SNARK [87]. Using a hybrid reasoning system for representing and reasoning about context seems interesting and has not been reported yet. Of special interest would be to investigate what are the most plausible aspects of context knowledge for special purpose reasoning.

6. Conclusions

This review shows that in the field of context-aware computing, many knowledge representation techniques have been experimented with. The field still lacks a major breakthrough – although using expressive (e.g. OWL DL) ontologies is suggested in a number of papers, the evidence does not yet show that these systems would meet all requirements. This lack of evidence comes up as the small number of work reporting quantitative evaluation and as the non-existence of work reporting large scale deployment.

While the computational issues of context representation have not been systematically treated in the field of pervasive computing, the reviewed work implies that formal context representation puts hard requirements on the KR&R systems in terms of dynamicity and expressiveness. Due to this, formal context representation runs across the fundamental trade-off between the expressiveness of representation and the complexity of reasoning. The future research has to investigate this interplay more closely to find the most suitable solutions for data-intensive pervasive computing systems. This work could be also geared towards understanding whether the required level of interoperability and uncertainty modeling could be achieved through other representation techniques. In order to measure future achievements the community should establish a common evaluation framework.

Acknowledgements

The authors would like to thank the Nokia Research Center and Infotech Oulu. The first author would like to thank the Fulbright-Technology Industries of Finland and the Finnish Graduate School in Electronics, Telecommunication, and Automation (GETA).

References

[1] M. Weiser, "The computer for the 21st century", Scientific American, Vol. 265, 3, pp. 94-104, 1991.
[2] B. Schilit, N. Adams & R. Want, "Context-aware computing applications", Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, pp. 85-90, 1994.
[3] A. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing, Vol. 5, 1, pp. 4-7, 2001.
[4] A. Schmidt, M. Beigl & H. Gellersen, "There is more to context than location", Computers and Graphics (Pergamon), Vol. 23, 6, pp. 893-901, 1999.
[5] R.M. Turner, "A Model of Explicit Context Representation and Use for Intelligent Agents", Proceedings of Modeling and Using Context: Second International and Interdisciplinary Conference, Trento, Italy, Trento, Italy.Vol. 1688, pp. 375-388, 1999.
[6] P. Gray & D. Salber, "Modelling and Using Sensed Context Information in the Design of Interactive Applications", Proceedings of Engineering for Human-Computer Interaction: 8th IFIP International Conference, Toronto, Canada, Vol. 2254, pp. 317-335, 2001.
[7] T. Strang, C. Linnhoff-Popien & K. Frank, "CoOL: A Context Ontology Language to Enable Contextual Interoperability", Proceeding of Distributed Applications and Interoperable Systems: 4th IFIP WG6.1 International Conference, Paris, France, Paris, France.Vol. 2893, pp. 236-247, 2003.
[8] S.W. Loke, "Logic Programming for Context-Aware Pervasive Computing: Language Support, Characterizing Situations, and Integration with the Web", in proc: IEEE/WIC/ACM International Conference on Web Intelligence, pp. 44-50, 2004.
[9] O. Lassila & D. Khushraj, "Contextualizing Applications via Semantic Middleware", in proc. of The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, San Diego, USA, pp. 183-189, 2005.
[10] T. Winograd, "Architectures for Context", Hum.-Comput.Interact., Vol. 16, 2, 3 & 4, pp. 401-419, 2001.
[11] R. Reichle, M. Wagner, M.U. Khan, K. Geihs, M. Valla, C. Fra, N. Paspallis & G.A. Papadopoulos, "A Context Query Language for Pervasive Computing Environments", Pervasive Computing and Communications, Sixth Annual IEEE International Conference on, pp. 434-440, 2008.
[12] A. Dey, G.D. Abowd & D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", Human-Computer Interaction, Vol. 16, pp. 97-166, 2001.
[13] F. Perich, A. Joshi, T. Finin & Y. Yesha, "On data management in pervasive computing environments", Knowledge and Data Engineering, IEEE Transactions on, Vol. 16, pp. 621-634, 2004.
[14] P.D. Haghighi, A. Zaslavsky & S. Krishnaswamy, "An Evaluation of Query Languages for Context-Aware Computing", Database and Expert Systems Applications, 17th International Conference on, pp. 455-462 , 2006.
[15] K. Henricksen & R. Robinson, "A survey of middleware for sensor networks: state-of-the-art and future directions", Proceedings of the international workshop on Middleware for sensor networks, Melbourne, Australia, pp. 60-65, 2006.
[16] R.J. Brachman and H.J. Levesque, Knowledge representation and reasoning, Morgan Kaufmann, Amsterdam, 2004.
[17] S. Russell & P. Norvig, Artificial intelligence – a modern approach, Pearson Education, Upper Saddle River, New Jersey, USA, 2003.
[18] Korpipää P. Blackboard-based software framework and tool for mobile device context awareness, Ph.D. University of Oulu, 2005.
[19] T. Strang & C. Linnhoff-Popien, "A Context Modeling Survey", Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham, England, 2004.
[20] P. Brezillon, "Context in problem solving: A survey", Knowledge Engrg. Rev., Vol. 14, 1, pp. 1-34, 1999.
[21] J. McCarthy, "Notes on formalizing context", Proceedings of IJCAI-93, Chambéry, France, pp. 555-560, 1993.
[22] K.D. Forbus and J.d. Kleer, Building problem solvers, MIT Press, Cambridge, MA, USA, 1993.
[23] R. Studer, V.R. Benjamins & D. Fensel, "Knowledge engineering: Principles and methods", Data Knowl.Eng., Vol. 25, 1-2, pp. 161-197, 1998.
[24] S. Bechhofer, f. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinnes, P.F. Patel-Schneider & L.N. Stein, "OWL Web Ontology Language Reference", W3C, 2004.
[25] B.C. Grau, B. Parsia, E. Sirin & A. Kalyanpur, "Modularity and Web Ontologies", in proc. KR, pp. 198-209, 2006.

[26] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen & E.J. Malm, "Managing context information in mobile devices", Pervasive Computing, IEEE, Vol. 2, 3, pp. 42-51, 2003.
[27] J. Soldatos, I. Pandis, K. Stamatis, L. Polymenakos & J.L. Crowley, "Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services", Computer Communications, Vol. 30, 3, 2007.
[28] J.L. Crowley, "Context Driven Observation of Human Activity", Ambient Intelligence, First European Symposium, EUSAI 2003, Veldhoven, The Netherlands, pp. 101-118, 2003.
[29] S. Jang & W. Woo, "Ubi-UCAM: A Unified Context-Aware Application Model", Modeling and Using Context, pp. 1026-1027, 2003.
[30] G. Castelli, M. Mamei & F. Zambonelli, "Engineering contextual knowledge for autonomic pervasive services", Information and Software Technology, Vol. 50, 1-2, pp. 36-50, 2008.
[31] O. Riva, "Contory: A Middleware for the Provisioning of Context Information on Smart Phones", in proc. Middleware, pp. 219-239, 2006.
[32] A. Ranganathan & R.H. Campbell, "An infrastructure for context-awareness based on first order logic", Personal Ubiquitous Comput., Vol. 7, 6, pp. 353-364, 2003.
[33] K. Sagonas, T. Swift & D.S. Warren, "XSB as an efficient deductive database engine", Proceedings of the ACM SIGMOD international conference on Management of data, Minneapolis, Minnesota, United States, New York, NY, USA.pp. 442-453, 1994.
[34] A. Ranganathan & R.H. Campbell, "Autonomic pervasive computing based on planning", Proceedings of International Conference on Autonomic Computing, May 17-18, New York, NY, USA, pp. 80-87, 2004.
[35] A. Ranganathan., A Task Execution Framework for Autonomic Ubiquitous Computing, Ph.D. diss., University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 2005.
[36] K. Henricksen & J. Indulska, "Developing context-aware pervasive computing applications: Models and approach", Pervasive and Mobile Computing, Vol. 2, 1, pp. 37-64, 2006.
[37] L. Shapiro and E.Y. Sterling, The Art of PROLOG: Advanced Programming Techniques, The MIT Press, 1994.
[38] A. Gómez-Pérez, M. Fernández-López and O. Corcho, Ontological Engineering, Springer-Verlag, London, 2003.
[39] P. Jackson, Introduction to Expert Systems, Addison Wesley Longman, Boston, MA, USA, 1998.
[40] F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P.F. Patel-Schneider, The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, New York, 2003.
[41] T.R. Gruber, "A translation approach to portable ontology specifications", Knowledge Acquisition, Vol. 5, 2, pp. 199-221, 1993.
[42] H. Chen, T. Finin, J. Anupam, L. Kagal, F. Perich & C. Dipanjan, "Intelligent agents meet the semantic Web in smart spaces", Internet Computing, Vol. 8, 6, pp. 69-79, 2004.
[43] H. Chen, F. Perich, T. Finin & A. Joshi, "SOUPA: standard ontology for ubiquitous and pervasive applications", Proceedings of The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 258-267, 2004.
[44] Resource Description Framework (RDF), URL: http://www.w3.org/RDF/.
[45] Jena - A Semantic Web Framework for Java, URL: http://jena.sourceforge.net.
[46] E. Friedman-Hill, Jess, the Rule Engine for the Java™ Platform, URL: http://herzberg.ca.sandia.gov/jess/index.shtml.
[47] T. Gu, H.K. Pung & D.Q. Zhang, "Toward an OSGi-Based Infrastructure for Context-Aware Applications", Pervasive Computing, Vol. 3, 4, pp. 66-74, 2004.
[48] X. Wang, J.S. Dong, C.Y. Chin, S.R. Hettiarachchi & D. Zhang, "Semantic Space: an infrastructure for smart spaces", Pervasive Computing, Vol. 3, 3, pp. 32-39, 2004.
[49] A. Seaborne., RDQL - A Query Language for RDF, URL: http://www.w3.org/Submission/RDQL/.
[50] D. Nicklas, M. Grossmann, J. Mínguez & M. Wieland, "Adding High-level Reasoning to Efficient Low-level Context Management: A Hybrid Approach", Pervasive Computing and Communications, Sixth Annual IEEE International Conference on, pp. 447-452, 2008.
[51] M. Khedr & A. Karmouch, "ACAI: agent-based context-aware infrastructure for spontaneous applications", Journal of Network and Computer Applications, Vol. 28, 1, pp. 19-44, 2005.
[52] B.N. Grosof, I. Horrocks, R. Volz & S. Decker, "Description logic programs: combining logic programs with description logic", WWW '03: Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary, pp. 48-57, 2003.
[53] I. Horrocks, P.F. Patel-Schneider, S. Bechhofer & D. Tsarkov, "OWL rules: A proposal and prototype implementation", Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 3, 1, pp. 23-40, 2005.

[54] A. Agostini, C. Bettini & D. Riboni, "Loosely coupling ontological reasoning with an efficient middleware for context-awareness", Proceedings of The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 175-182, 2005.
[55] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M.H. Butler & L. Tran, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0", 2004.
[56] D. Khushraj, O. Lassila & T. Finin, "sTuples: semantic tuple spaces", Mobile and Ubiquitous Systems: Networking and Services, The First Annual International Conference on, pp. 268-277, 2004.
[57] E. Freeman, K. Arnold and S. Hupfer, JavaSpaces Principles, Patterns, and Practice, Addison-Wesley Longman Ltd, Essex, UK, UK, 1999.
[58] I. Horrocks, DAML+OIL: "A Reason-able Web Ontology Language", Advances in Database Technology - 8th International Conference on Extending Database Technology, March 25-27, Prague, Czech Republic, Vol. 2287, pp. 2-13, 2002.
[59] M. Luther, Y. Fukazawa, B. Souville, K. Fujii, T. Naganuma, M. Wagner & S. Kurakake, "Classification-based Situational Reasoning for Task-oriented Mobile Service Recommendation", Proceedings of the 2nd International Workshop on Contexts and Ontologies: Theory, Practice and Applications, Riva del Garda, Italy, 2006.
[60] T. Naganuma & S. Kurakake, "Task Knowledge Based Retrieval for Service Relevant to Mobile User's Activity", in proc. The Semantic Web – ISWC 2005, Galway, Ireland, Vol. LNCS 3729, pp. 959-973, 2005.
[61] Martin,D., Burstein M., Hobbs J., et al, (accessed Sep 1 2007) "OWL-S: Semantic Markup for Web Services", URL: http://www.w3.org/Submission/OWL-S/.
[62] R. Lewis & J.M.Cantera Fonseca, (accessed May 28 2008) "Delivery context ontology", URL: http://www.w3.org/TR/2008/WD-dcontology-20080415/.
[63] F. Bobillo, M. Delgado & J. Gómez-Romero, "Representation of context-dependant knowledge in ontologies: A model and an application", Expert Systems with Applications, Vol. 35, 4, 2008.
[64] T. Chaari, D. Ejigu, F. Laforest & V. Scuturici, "A comprehensive approach to model and use context for adapting applications in pervasive environments", Journal of Systems and Software, Vol. 80, 12, pp. 1973-1992, 2007.
[65] M. Van Kleek & H. Shrobe, "A Practical Activity Capture Framework for Personal, Lifetime User Modeling", in proc. User Modeling, pp. 298-302, 2007.
[66] T..Mitchell, Machine Learning, McGraw-Hill, 1997.
[67] A. Aamodt & E. Plaza, "Case-Based Reasoning - Foundational Issues, Methodological Variations, and System Approaches", AI Communications, Vol. 7, 1, pp. 39-59, 1994.
[68] D.B.Leake, "CBR in Context: The Present and Future", In: Leake,D.B., Editor, Case-Based Reasoning: Experiences, Lessons, and Future Directions, AAAI Press/MIT Press, Menlo Park, CA, USA, 1996.
[69] M. Minsky, "A Framework for Representing Knowledge", MIT AI Laboratory memo 306, 1974.
[70] R.M. Turner, "Context-mediated behavior for intelligent agents", International Journal of Human-Computer Studies, Vol. 48, 3, pp. 307-330, 1998.
[71] A. Zimmermann, "Context-Awareness in User Modelling: Requirements Analysis for a Case-Based Reasoning Application", in proc. Case-Based Reasoning Research and Development: 5th International Conference on Case-Based Reasoning, LNAI 2689, Trondheim, Norway, pp. 718-732, 2003.
[72] A. Kofod-Petersen & A. Aamodt, "Case-Based Situation Assessment in a Mobile Context-Aware System", in proc. Artificial Intelligence in Mobile Systems, pp. 41-49, 2003.
[73] A. Ranganathan, J. Al-Muhtadi & R.H. Campbell, "Reasoning about uncertain contexts in pervasive computing environments", Pervasive Computing, Vol. 3, 2, pp. 62-70, 2004.
[74] Z. Ding & Y. Peng, "A probabilistic extension to ontology language OWL", in proc. 37th Annual Hawaii International Conference on System Sciences, pp. 111-120, 2004.
[75] T. Gu, H.K. Pung & D.Q. Zhang, "A bayesian approach for dealing with uncertain contexts", in proc. Second International Conference on Pervasive Computing, Vienna, Austria, 2004.
[76] B.A. Truong, Y.-. Lee & S.-. Lee, Modeling uncertainty in context-aware computing, in proc. Fourth Annual ACIS International Conference on Computer and Information Science, pp. 676-681, 2005.
[77] A. Schmidt, Ontology-Based User Context Management: "The Challenges of Imperfection and Time-Dependence", On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE, pp. 995-1011, 2006.
[78] J. Mäntyjärvi & T. Seppänen, "Adapting Applications in Mobile Terminals Using Fuzzy Context Information", Human Computer Interaction with Mobile Devices, pp. 383-404, 2002.
[79] A. Padovitz, S.W. Loke & A. Zaslavsky, "The ECORA framework: A hybrid architecture for context-oriented pervasive computing", Pervasive and Mobile Computing, Vol. 4, 2, pp. 182-215, 2008.

[80] J. Riekki, J. Huhtinen, P. Ala-Siuru, P. Alahuhta, J. Kaartinen & J. Roning, "Genie of the net, an agent platform for managing services on behalf of the user", Comput.Commun., Vol. 26, 11, pp. 1188-1198, 2003.
[81] F.L. Gandon & N.M. Sadeh, "Semantic web technologies to reconcile privacy and context awareness", J.Web Sem., Vol. 1, 3, pp. 241-260, 2004.
[82] D.J. Cook, "MavHome: an agent-based smart home", Pervasive Computing and Communications, in proc. First IEEE International Conference on, pp. 521-524, 2003.
[83] L. Serafini & P. Bouquet, "Comparing formal theories of context in AI", Artificial Intelligence, Vol. 155, 1-2, pp. 41-67, 2004.
[84] J. Maluszynski, "Combining Rules and Ontologies. A survey", REWERSE, Tech. Rep. I3-D3, March 3, 2005.
[85] B. Parsia, C. Halaschek-Wiener & E. Sirin, "Towards Incremental Reasoning Through Updates in OWL DL", in proc. workshop on Reasoning on the Web, May 22nd, Edinburgh, UK, 2006.
[86] P. Haase, F. Harmelen, Z. Huang, H. Stuckenschmidt & Y. Sure, "A Framework for Handling Inconsistency in Changing Ontologies", in proc. 4th International Semantic Web Conference, November 6-10, Galway, Ireland, pp. 353-367, 2005.
[87] M.E. Stickel, R.J. Waldinger & V.K. Chaudhri, (accessed: Sep 24, 2009) "A Guide to SNARK", URL: http://www.ai.sri.com/snark/tutorial/tutorial.html.
[88] C. Becker & F. Dürr, "On location models for ubiquitous computing", Personal Ubiquitous Comput., Vol. 9, 1, pp. 20-31, 2005.

Authors

Mikko Perttunen is a Ph.D. student in the Computer Engineering Laboratory of the Department of Electrical and Information Engineering in the University of Oulu. He received a M.Sc. from the Department of Electrical and Information Engineering at the University of Oulu in September 2004. His research interests include data mining, context-awareness, and the Semantic Web.

Jukka Riekki obtained the degree of Doctor of Technology in 1999 at the University of Oulu in Finland. Since 1986, he has been a member of faculty of the University of Oulu, where he is currently Professor in the Department of Electrical and Information Engineering. He leads the Intelligent Systems Group together with Prof. Juha Röning and Prof. Tapio Seppänen. His main research interests are in context-aware systems serving people in their everyday environment. He has over 20 year experience of both basic and applied research projects.

Ora Lassila is a technology strategist at Nokia Services, and a member of Nokia's CEO Technology Council. Prior to joining Nokia Services, Lassila was Research Fellow at the Nokia Research Center Cambridge. He has been an elected member of the Advisory Board of the World Wide Web Consortium (W3C) since 1998, and represented Nokia in the W3C Advisory Committee in 1998-2002. In 1996-1997 he was a Visiting Scientist at MIT Laboratory for Computer Science, working with W3C and launching the Resource Description Framework (RDF) standard; he served as a co-editor of the RDF Model and Syntax specification. Much of his research work at the Nokia Research Center focused on the Semantic Web and particularly its applications to mobile and ubiquitous computing. He collaborated with several US universities, and was an active participant in the DARPA Agent Markup Language (DAML) program. His previous positions include Project Manager at the Robotics Institute of Carnegie Mellon University and Research Scientist at the CS Laboratory of Helsinki University of Technology. He has also worked as a software engineer in several companies (including his own start-up). He is the author of more than 80 conference papers and journal articles. Dr. Lassila holds a Ph.D. in Computer Science from the Helsinki University of Technology.