The Knowledge Engineering Review, Vol. 22:4, 315–347. © 2007, Cambridge University Press
doi:10.1017/S0269888907001208 Printed in the United Kingdom

Ontology-based models in pervasive computing systems

JUAN YE, LORCAN COYLE, SIMON DOBSON and PADDY NIXON

Systems Research Group, School of Computer Science and Informatics, UCD Dublin, Ireland;
e-mail: juan.ye@ucd.ie, lorcan.coyle@ucd.ie, simon.dobson@ucd.ie, paddy.nixon@ucd.ie

Abstract

Pervasive computing is by its nature open and extensible, and must integrate the information from a diverse range of sources. This leads to a problem of information exchange, so sub-systems must agree on shared representations. Ontologies potentially provide a well-founded mechanism for the representation and exchange of such structured information. A number of ontologies have been developed specifically for use in pervasive computing, none of which appears to cover adequately the space of concerns applicable to application designers. We compare and contrast the most popular ontologies, evaluating them against the system challenges generally recognized within the pervasive computing community. We identify a number of deficiencies that must be addressed in order to apply the ontological techniques successfully to next-generation pervasive systems.

1 Introduction

In 1991, Mark Weiser claimed that 'the most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it' (Weiser, 1991). His work pioneered the field of ubiquitous or *pervasive computing*; pervasive computing systems are interactive systems that involve multiple devices, services, and software agents. They provide appropriate behaviours adapting to the user's changing tasks and environments through different interface modalities and devices (Dobson & Nixon, 2004). *Context* informs this process by providing a structured, unified view of the world in which the system operates (Coutaz *et al*., 2005). The best and most adaptive pervasive computing applications are those that are most context-aware.

In April 2004, the European Commission's IST Future and Emerging Technologies group and the US National Science Foundation jointly supported the Disappearing Computer Strategic Research Workshop (Skordas *et al*., 2004). The objectives of this workshop were to consolidate the research experiences in the domain of pervasive computing and to map out the core and fundamental challenges for the next stage of research in the field. The discussions at this workshop came under many areas, including discovering the fundamental primitives of pervasive computing, understanding their semantics, and developing corresponding implementations. In the area of information retrieval and management, they included developing semi-automatic approaches that allow users, devices, and applications to extract from their environment the necessary information to operate. In the area of security, they covered security, privacy, and trust infrastructures that aimed to maximize user confidence in pervasive computing systems. In the area of human–computer interaction, the discussions covered the development of hardware infrastructure for input and output interaction, of software infrastructure for manipulating and controlling interaction devices, and of core enabling middleware services. The workshop also covered a discussion

on the potential impact of influential, new, and developing technologies on the field. These discussions led to a special issue on 'The Disappearing Computer' in the Communications of the ACM (Streitz & Nixon, 2005), which refined these initial areas into five guiding themes:

Sensing and context In order to develop reactive pervasive computing systems it will be necessary to capture, process, and exploit the contextual parameters that inform and guide human behaviour.

Privacy, trust, and security Privacy encompasses reasoning about trust and risk involved in the interactions between users and services. Trust controls the amount of information that can be revealed in an interaction. Risk analysis allows us to evaluate the expected benefit that would motivate users to participate in these interactions. Security describes the cryptographic techniques used to secure the communication channels and required data.

Discovery One of the key requirements for pervasive computing systems is an approach or service capable of assimilating and filtering information from its various inputs (such as sensors, services, applications, and users). Given some infrastructure to communicate this information, an approach to matching will be needed, which correlates relevant input events and facts to a particular contextual service. This is essential to allow the user and the application to discover the necessary information from the environment to achieve a defined goal or complete an activity.

Interaction design As computers disappear from pervasive computing environments, novel human–computer interactions will need to be investigated to deal with the peculiarities of their environments, including invisible devices, implicit interaction, and real, virtual, and hybrid interactions.

Essential infrastructure It will be necessary to develop tools to maintain and upgrade infrastructure over its entire life cycle, as well as to allow the infrastructure to communicate failures effectively to its users.

Besides the challenges in the above five themes, we add an additional challenge for pervasive computing, that of modelling and handling uncertainty. Many data in a pervasive system are inherently uncertain, since they come directly from real-world sources, which often provide data that are incorrect, imprecise, conflicting, or incomplete. For this reason, pervasive computing systems should integrate uncertainty into every decision they make.

Ontologies, as a promising means for knowledge sharing and reuse, have gained recognition in other fields of computer science, including e-commerce (Obrst *et al.*, 2001; Eckstein *et al.*, 2004), information integration (Guidetti, 2002; Varzi & Vieu, 2004; López de Vergara *et al.*, 2003), and the semantic Web (Gil *et al.*, 2005; Sure & Domingue, 2006). Ontologies provide standard and formal semantics, whose strength in conceptualization is exerted by their normalization and formalization. The normalization is reflected in a semantic agreement for the meaning of terms. The formalization is reflected in formal ontology languages that are used to encode ontologies (Bachimont *et al.*, 2002). This survey analyzes the existing use of ontologies in pervasive computing and proposes directions for future work, along the lines of the strategic themes outlined earlier.

The remainder of the paper is arranged as follows. Section 2 introduces the prominent definitions of ontologies with emphasis on the nature of ontologies, and describes the standard languages that are used to build ontologies. It summarizes the advantages of ontologies for computer science applications, and describes the issues relative to formal ontology development. It also outlines a number of criteria under which existing ontologies should be evaluated.

Section 3 introduces several prominent pervasive computing systems that use ontological modelling. We focus our analysis on *CoBrA*, developed in the University of Maryland (Chen *et al.*, 2004b), which uses a pervasive ontology called *SOUPA* to characterize the key concepts of pervasive computing; *Gaia*, developed in the University of Illinois at Urbana-Champaign, which uses ontologies to deal with context awareness, service discovery and matchmaking, and interoperation between entities in a pervasive computing environment (Ranganathan *et al.*, 2004b); *GLOSS*, developed by four European universities, which employs ontologies for the precise understanding of various contexts and services in the smart space (Coutaz *et al.*, 2003); *ASC*, developed in the German Aerospace Center, which uses CoOL (Context Ontology Language) to enable context awareness and interoperability (Strang *et al.*, 2003b); and *CONON*, developed at the National University of Singapore, which constructs the upper ontologies for general concepts in a pervasive

computing environment, and also keeps the domain-specific ontologies extensible (Gu *et al*., 2004). In the end, we discuss a number of pervasive computing systems that use task-specific ontologies.

In Section 4, we analyze the pervasive systems ontologies introduced in Section 3. In Section 4.1, we evaluate the approaches applied by these systems to ontology modelling, and assess the generated structures and conceptualizations. The following sections evaluate these deployments along the strategic themes outlined by Skordas *et al*. (2004) and Streitz & Nixon (2005). Many of these themes have provided natural playgrounds for ontology engineers. Section 4.2 analyzes work in the use of ontologies in *representing context data* and reasoning about them. Section 4.3 analyzes the work done in the areas of *privacy and trust management*. Section 4.4 analyzes the work in the area of *discovery*, which is used to match producers and consumers of contextual information in pervasive environments. Section 4.5 analyzes the use of ontological modelling in improving *interaction design*. The theme of *essential infrastructure* is concerned with the development of infrastructures, and as such it is not an appropriate match for the deployment of ontology engineering techniques. Section 4.6 analyzes the work done in representing uncertain data for pervasive computing. Section 4.7 summarizes our findings.

Finally, Section 5 concludes the survey and generalizes the results of our evaluations. It proposes the suggestions for improving the development of ontologies with respect to the key themes of research identified here.

2 Background on ontologies

Before analyzing the existing applications of ontology-based models in pervasive computing systems, we introduce some background on ontologies. We chronicle the historical definitions of the term *ontology* and expound on the value of ontology use. Later, we introduce the development of ontological technologies, especially the standard ontology languages. Finally, we distill the guidelines for developing ontologies from the ontology engineering community into a set of best practices, and compile a list of design principles for evaluating existing ontologies. In Section 4, we will apply these criteria to evaluate the quality of the most prominent ontologies in use in the field of pervasive computing.

2.1 Ontology definitions

In the Merriam-Webster dictionary, an *ontology* is defined as: 'a branch of metaphysics concerned with the nature and relations of being; or a particular theory about the nature of being or the kinds of existences'. This philosophical definition reflects the essence of ontology: capturing the natural features of realities and relations between realities. The term was borrowed from philosophy and introduced into the knowledge engineering field as a means of abstracting and representing knowledge. Ontologies are used to build consensual terminologies for the domain knowledge in a formal way so that they can be more easily shared and reused.

More recently, ontologies have been applied in many fields of computer science, such as the semantic Web, e-commerce, and information systems. Accordingly, the definitions of ontologies have evolved with different and complementary points of view. Some of these definitions stress the general and intrinsic properties of ontologies, while others are influenced by the mechanical means of developing ontologies. We will illustrate several typical definitions in both of the perspectives.

Gruber (1993) and Borst (1997) introduced a precise definition: 'An ontology is a formal explicit specification of a shared conceptualization'. Fensel (2001) elaborated on this definition by saying that conceptualization 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. *Formal* refers to the fact that the ontology describes what each concept is meant to denote, and specifies formal axioms that constrain the interpretation and well-formed use of these concepts. *Shared* reflects the notion that an ontology captures consensual knowledge; that is, it is not private of some individual, but

accepted by a group. This definition describes how the philosophical nature of ontologies could be incorporated into computer science.

Most of the subsequent definitions (such as Benjamins *et al.*, 1998; de Vergara *et al.*, 2002) are similar to the above one *per se*, while Guarino (1998) refined Gruber's definition by distinguishing ontologies from conceptualizations formally and logically: 'An ontology is a logical theory accounting for the intended meaning of a formal vocabulary'. When this logical theory is used to model a particular aspect of reality—an *intended domain*—an *ontological commitment* is specified to capture the very basic ontological assumptions (such as identities and internal structures) about the intended domain. The ontological commitment gives explicit information about the intended nature of the modelling primitives and their *a priori* relationships for this logical theory. Thus, it constrains a subset of all the possible models of the logical theory by specifying the intended meaning of its vocabulary. These models being constrained are called *intended models*, which only describe those states of affairs that are compatible with the underlying ontological commitment (Guarino *et al.*, 1994). Furthermore, Guarino (1998) focused on the application side to offer a systematic account of the central role that ontologies played in information systems, leading to the ontology-driven information systems; that is, ontologies 'drive' all aspects and all components of an information system.

Another typical style of ontology definition was introduced by Uschold & Jasper (1999): 'An ontology may take a variety of forms, but it will necessarily include a vocabulary of terms and some specification of their meaning. This includes definitions and an indication of how concepts are interrelated which collectively impose a structure on the domain and constrain the possible interpretations of terms'. This definition specified the functions of ontologies and the technical approach for building ontologies.

Gruber and Borst's definition reflect the philosophical foundations of ontologies. Guarino advanced the definition with a logical theory, and demonstrated that ontologies could advantageously drive information systems. The definitions not only help researchers to comprehend the ontology's essence but also work as a mechanical guide for ontology development. Further, Uschold introduced into the definition the semantics and the technical approach for building ontologies. The above historical definitions record how the term *ontology* is more and more concrete and applicable in computer science.

Different types of ontologies are built for different types of applications, and they have varying levels of details. Ontologies are classified in terms of the level of generality (de Bruijn, 2003):

- *generic ontologies*, which describe general concepts, independent of any particular domains;
- *domain ontologies*, which describe concepts for a particular domain (such as, biology or physics);
- *application ontologies*, which describe the concepts necessary for specific applications. The application ontologies might build on generic and domain ontologies (Chandrasekaran *et al.*, 1999);

and in terms of the level of expressiveness:

- *lightweight ontologies*, which aim for a consensual conceptualization, including concepts, taxonomies, and relationships between concepts and their properties (Corcho *et al.*, 2003);
- *heavyweight ontologies*, which apply axioms and constraints to lightweight ontologies. The axioms and constraints are the main building blocks for making the semantic interpretation for the concepts and relationships of the ontologies (Gruber, 1995).

In a general, formal, and explicit way, ontologies capture and specify the domain knowledge with its intrinsic semantics through consensual terminologies and formal axioms and constraints. Technically, ontologies should involve a well-formed vocabulary with clearly defined relations between different terms. There exist different types of ontologies. According to the generality, they can be classified into generic, domain, and application ontologies; according to the expressivity, they can be classified into light- and heavyweight ontologies.

2.2 The applications of ontologies

Ontologies have come into widespread use in many fields, including knowledge representation and integration, information retrieval and extraction, and conceptual model design. They can provide standard terminologies and rich semantics to facilitate knowledge sharing and reuse. Rich semantics requires the ability to understand not only the information but also the environment and reasoning surrounding the use of the information (Bebee *et al.*, 2003). In ontologies, the semantics are embodied by a set of terms, relations between terms, and inference rules for a topic (Gruber, 1993), which are the kernel of ontologies. Terms constitute a controlled vocabulary with explicit definitions. Relations between terms include those between instances, between classes and their instances, and between classes. Inference rules make it possible to define knowledge about a system, which might otherwise be hidden or implicit, and make it possible to derive new knowledge from existing facts.

It can be difficult for a community to agree to a shared conceptualization of a domain. However, it may be realistic to assume that if such generic ontologies gain a wide consensus, they should be shared. Domain and application ontologies may differ to deal with the individual needs of the systems that use them. For example, in a pervasive system, the modelling of location data may be customized to the capabilities of the sensor equipment. Such a location ontology should be generic enough to be independent of the needs of individual sensors and applications, that is, the modelling of location data should be done by the domain ontology. However, application ontologies that use these data are task-dependent, which can be more specific and might not be sharable by other application or specific ontologies.

There exists an assumption that a system may contain multiple non-shared ontologies. For example, Chandrasekaran *et al.* (1999) proposed a practical statement: 'An ontology is unlikely to cover all possible potential uses. In that sense, both an ontology for a domain and a knowledge base written using that ontology are likely to be more appropriate for certain uses than others and unlikely to be sharable across widely divergent tasks'. If systems using different ontologies need to communicate, the relevant concepts from each system's ontology will have to be mapped to each other. Ontology mapping allows semantics to cross the boundaries between systems with heterogeneous ontologies (see Kalfoglou & Schorlemmer, 2003 for a comprehensive review). If two systems use or customize the same generic or domain ontologies, the semantic interoperability is greatly facilitated: they share the same terminologies, the understanding of what classes, properties, and individuals are, and how they relate to each other (Noy, 2005).

In the following, we will classify ontological applications with respect to two perspectives: the scale and the functions.

From an *upper-level* view (i.e. between systems), general sets of libraries of ontologies are abstracted from different systems that describe common situations. The ontologies at the upper level are generic or domain ontologies. Through a standard specification, these ontologies can be shared, reused, and adapted to other systems; they are customized to represent terms in different forms for different users. They should be extensible to allow for the incorporation of new classes and the specialization of concepts and constraints for a particular problem (Uschold & Grüninger, 1996).

From a *lower-level view* (i.e. within a system), ontologies work as a specification of what the system is designed for. The ontologies at the lower level are application (task-dependent) ontologies. They can facilitate the process of identifying the requirements and understanding the relationships among the components of a system. Ontologies can identify the logical connections between elements across the components. Thus, they enable the use of (semi-)automation in checking system consistency and integrity with the specification. In addition, formal ontologies can clarify the assumptions made by different components of the software system, which helps to integrate these components (Uschold & Grüninger, 1996).

In a system, it is often necessary to provide a mechanism to search for a particular piece of information (such as a service or context). Ontologies enhance the searching mechanisms, which may refer to a precise concept rather than a plain definite keyword. Traditional information

retrieval technology is based on keyword processing. The same keywords can have different meanings in different situations, while sometimes different keywords can have similar meanings (Zhang & Li, 2005). This difficulty in grasping semantic meaning from keywords may result in poor performance. A high precision of keywords will tend to result in poor recall—that is, fewer, but more relevant answers will be retrieved. As precision is decreased, more, less relevant documents will tend to be retrieved, resulting in higher recall. The introduction of semantics through ontologies will lead to a new generation of services based on content rather than only on syntax. Thus, searching will be based on topics, and resources will be retrieved related to the semantics of a user's request (Bonino *et al.*, 2004).

2.3 Technologies for developing ontologies

Now we describe the ontology technologies that the W3C recommends for encoding ontologies, as these are widely used in the ontology engineering field and specifically in pervasive computing applications.

The eXtensible Markup Language (XML) is a standard language for describing data in a (semi-)structured manner. In XML, data are labelled (or tagged) with user-defined elements that can contain further elements and data. Document Type Definition (DTD) and XML Schema (XMLS) were introduced to constrain the syntactic structures of XML documents.

The Resource Description Framework (RDF) is an application of XML that imposes needed structural constraints to provide unambiguous methods of expressing semantics. The structural constraints are used to support the consistent encoding and exchange of standardized metadata. They make it possible to interchange the separate packages of metadata defined by different resource communities (Miller, 1998). RDF provides an unambiguous and extensible way to express simple statements in the form $\langle$ subject, predicate, object $\rangle$, where a subject denotes a resource represented as Uniform Resource Identifiers (URI) that can be identified uniquely and globally, an object can be either a literal (such as a string or number) or a URI reference to another resource, and a predicate is a relationship between a subject and an object. RDF Schema (RDFS) provides the facilities to describe the application-specific classes and properties, and to indicate which classes and properties are expected to be used together. In other words, RDFS provides a type system for RDF; it extends RDF with primitive concepts like classes, properties, and instance, and primitive relationships like instance-of and subclass-of.

To support formal semantics and efficient reasoning, the Ontology Interchange Language (OIL) and the DARPA Agent Markup Language (DAML) were designed. These were layered on top of XML(S) and RDF(S). By combining DAML and OIL, DAML+OIL was formed, and was proposed as a W3C standard for ontological and metadata representation (Bechhofer *et al.*, 2001b). DAML+OIL supports primitive and more complex data structures, well-structured semantics, and inference procedures based on description logic.

Based on DAML+OIL, the Web Ontology Language (OWL) was created. OWL facilitates greater interpretability of data by providing additional vocabularies with formal semantics (McGuinness & van Harmelen, 2004). The OWL language provides three increasingly expressive sub-languages designed for use by specific communities of implementers and users. It consists of OWL Lite, OWL DL, and OWL Full in a layered approach; that is, OWL Lite $\subseteq$ OWL DL $\subseteq$ OWL Full. OWL Lite supports classification hierarchies and simple constraints; OWL DL supports maximum expressiveness while retaining computational completeness and decidability; OWL Full allows for maximum expressiveness and the syntactic freedom of RDF without computational guarantees. To date, OWL DL has been the most practical choice for most ontological applications (de Bruijn, 2003). OWL Lite uses only some of the OWL language's features, and has more limitations on the use of the features compared with OWL DL or OWL Full. For example, one of the restrictions on OWL Lite is that it only uses named classes. OWL Full combines the expressivity of OWL with the flexibility and meta-modelling features of RDF; however, use of the OWL Full features means that developers will lose some guarantees that OWL DL and

OWL Lite can provide for reasoning systems. Choosing which sub-language of OWL to use depends on the expressiveness requirement and complexity of the target ontologies.

These formally specified ontology languages make semantics explicit and unambiguous so that ontologies are more amenable to automatic processing and integration (Noy, 2005). Besides ontology languages, ontology technologies include inference engines, annotation tools, ontology-based crawlers, and mining tools (Sure *et al.*, 2004). *Inference engines* (such as Decker *et al.*, 1999; Haarslev & Moeller, 2003; Ha *et al.*, 2005) are used to check the consistency and integrity of ontologies and deduce new knowledge based on the rules or relationships of the concepts specified by ontologies. *Ontology-based annotation tools* (such as Vargas-Vera *et al.*, 2001; Mostowfi *et al.*, 2005) use pre-defined concepts in ontologies to mark up a document. These concepts are a set of instances of classes and relations based on the domain ontology. *Ontology-based crawlers* (such as Ganesh *et al.*, 2004; Erdmann *et al.*, 2001) can retrieve most relevant Web pages or Web caches for users by applying the domain-dependent ontology to prioritize an URL queue. *Ontology mining tools* (such as Pandey & Mishra, 2005; Bernstein *et al.*, 2005) employ relevance functions to the ontologies to unravel the relationships between facts and the existing classes. Additionally, diverse ontology editors have been developed, including Protégé (Noy *et al.*, 2000) and OilEd (Bechhofer *et al.*, 2001a). A detailed analysis of these editors can be found in Denny (2002). The wealth of mature ontology technologies available today is a further incentive for other developers to apply ontologies to problems in other fields, including the pervasive computing domain.

2.4 Ontology engineering

We have shown how ontologies can be applied and which technologies can be used. Now we will detail a few development engineering issues that should be addressed when designing ontologies for a domain: determining the resources, scope, and purpose of required ontologies; capturing the appropriate terms; encoding the ontologies with formal languages; iteratively evaluating the building process and the ontologies in terms of clarity, coherence, ontological commitment, orthogonality, encoding bias, and extensibility; and annotating ontologies with formal documentations.

2.4.1 Ontology development methodologies

de Bruijn (2003) says that, besides complexity in the nature of the domain that ontologies apply to, there exist other complicated issues in both the requirements of ontologies and complexity in ontology languages. Therefore, formal methodologies to guide the development of ontologies are needed just as software engineering needs formal techniques to guide the process of developing software. Currently, a series of formal approaches have been developed in ontology engineering, such as Grüninger & Fox (1995), Uschold & Grüninger (1996), *METHONTOLOGY* (Fernandez *et al.*, 1997), and the *On-To-Knowledge* methodology (Staab *et al.*, 2001; Sure *et al.*, 2002). Jones *et al.* (1998) survey some of these approaches. These methodologies are similar to each other and provide a number of guidelines that should be followed during the ontology engineering process:

Requirement analysis As with software engineering, the most important thing when designing ontologies is to specify the purpose of the ontologies and their necessity. The developers must decide on the kinds of applications that these ontologies will be employed in. They need to determine the scope of the ontologies: whether they are general, domain, or application ontologies. They also need to confirm what resources will be applied in the target system so that they can decide what concepts should be covered in the ontologies.

Building ontologies When building ontologies, the developers should arrange the concepts and terms that need to be captured by these ontologies. There are two traditional approaches to capture and organize terms: bottom-up and top-down. The bottom-up approach starts from the finest-grained terms and generalizes them into different levels of details step by step. This approach leads to a very high level of detail; it is also difficult to spot commonality between related concepts, and increases the risk of inconsistencies. The top-down approach starts from the most coarse-grained terms and

divides up the terms into finer-grained terms. This approach can control the level of detail better; however, it has a risk of less stability. The trade-off solution between the above two is the middle-out approach proposed by Uschold & Grüninger (1996): starting with the most important concepts, and then defining higher-level concepts in terms of these. Thus, the higher-level categories will naturally arise. Furthermore, the most important concepts can be extended by defining the lower-level concepts with finer granularity. This approach relieves the problems existing in a bottom-up approach and those in a top-down approach. After deciding the required concepts, the developers should choose the most appropriate meanings for terms. They should attempt to reuse the most integral and precise definitions for these terms from existing mature ontological definitions. Standard ontology languages should be used to implement ontologies so as to ensure that the conceptualization is formally and explicitly encoded.

Evaluation Developers should perform evaluations of the ontology throughout the whole life cycle of ontology development. It is important to validate whether the ontologies model the system that they are intended to represent. The evaluating approaches include requirement analysis, and informal or formal competency questions (Grüninger & Fox, 1995). The competency questions are defined as an entailment or consistency problem with respect to axioms in ontologies, such as 'given the set of axioms and a set of instances of objects and relations, can we infer some first-order statement that uses only predicates in the language of the given ontologies?' (Uschold & Grüninger, 1996).

Documentation A formal documentation is needed to describe the completed ontologies. Inadequate documentation of ontologies is one of the main barriers for ontology sharing. To address this problem, Uschold & Grüninger (1996) suggested that all important assumptions should be documented, including those about the main concepts defined and the primitives used to express the definitions.

Ontology engineering is more science than art. If ontologies are to realize their full potential, developers should employ the appropriate development methodologies. So far, a number of tutorials for ontology development have been proposed, which are generalized from the experiences of building different types of ontologies. We introduce the above simplified guideline for constructing ontologies, instead of suggesting any specific technique. Similarly, Jones *et al.* (1998) point out that it is important to take into account the variety of experience that is available, rather than basing the methodology too much on the experience of one or two projects.

2.4.2 Ontology design principles

As more and more ontologies have been built, ontology evaluation has become an important issue that must be addressed. The ontology developer needs a way to evaluate the constructed ontologies and possibly to guide the construction process and any refinement step (Brank *et al.*, 2005). A lot of research has been done in evaluating the quality of ontologies. Colomb (2002) proposes three questions correspondingly from three linguistic dimensions: syntactic—whether the model is syntactically correct; semantic—whether the model covers the domain of interest; and pragmatic—whether the model is comprehensible to a user. Similarly, Burton-Jones *et al.* (2005) apply semiotic theory to develop a suite of metrics used to assess ontologies in these linguistic dimensions and in social aspects. Semiotic theory studies the properties of symbols, including how meanings of symbols are constructed and understood. Burton-Jones *et al.* describe the metrics' theoretical basis and collects data to test their feasibility. These metrics try to help ontology developers to capture a more comprehensive and consistent representation of the targeted domain, to check ontologies' syntax, to ensure that ontologies' semantics are meaningful and precise, and to make ontologies relevant for many users/agents. Tartir *et al.* (2005) analyze ontology schema and their populations, and describes them through a well-defined set of metrics. de Bruijn (2003) introduces five design parameters that must be met when developing a good-quality ontology. We combine the work in Uschold & Grüninger (1996), de Bruijn (2003), and Burton-Jones *et al.* (2005), and propose the following coarse-grained criteria to assess ontologies:

Clarity Terms must be defined through necessary and sufficient conditions so that they can be identified unambiguously and communicated effectively. A concept can be called clear if it can be definitely

recognized and distinguished from other concepts through its particular properties and constraints. Developers should minimize the ambiguity, motivate the distinctions between concepts, and provide examples for those concepts that lack necessary and sufficient conditions.

Coherence Definitions must be consistent. Local conflicts occur when the same term is used in two or more ways within one ontology. Incongruity should be detected to avoid reaching incorrect inferences. A general or domain ontology should satisfy both the local coherence requirement and the global coherence requirement, that is, its term definitions must be consistent with those in the typical consensual ontologies.

Ontological commitment Ontologies should make just enough claims about the domain so as to support the intended knowledge sharing and reuse. If too many claims are made on a domain, the extensibility of ontologies is limited; however, if too few are made, the range of applications that can actually use the ontology will be reduced. Developers should restrict themselves to making ontological commitments with respect to the aspects intrinsic to the domain being modelled.

Encoding bias Ontologies should be specified at the knowledge level without depending on a particular symbol-level encoding (Uschold & Grüninger, 1996), for example, the format that dates can be encoded should not be specified in a time ontology. They should be as independent as possible from the applications that will use the ontology, making it easier to facilitate knowledge sharing. Developers should take care not to make representation choices for the convenience of notation or implementation.

Extensibility Ontologies should offer a conceptual foundation for anticipated and potentially anticipated tasks. It should be easy to add new terms to ontologies without affecting existing ones; that is, new items will not cause modification, and will not produce confused or ambiguous meaning on existing ones.

The above design principles are not comprehensive evaluation criteria, and we use them only as a starting point for evaluating ontologies. A new criterion, *orthogonality*, is introduced, which requires that general concepts should be defined as independent and loosely coupled atomic concepts. Each aspect of the knowledge corresponds to an atomic concept, whose meaning is mutually exclusive to others. These atomic concepts can be combined or specialized to suit particular applications or problems. Orthogonality helps to achieve a suitable *ontological commitment* by ensuring that peripheral claims which are external to the concepts in a particular domain are not made. In terms of particular problems or applications, developers are encouraged to customize, combine, or extend these general concepts into more specific ones, which can make more claims. Furthermore, orthogonality, suitable ontological commitment, and minimization of encoding bias will help to achieve extensibility.

2.5 Summary

In this section we illustrated the chronological definitions of ontologies, and concluded with an operational definition: ontologies are used to capture and specify the domain knowledge, whose semantics are expressed through consensual terminologies and formal axioms and constraints. From this definition, we summarized the general applications of ontologies for communication between components or modules within a system, and for the interaction between systems. We then introduced the standard ontology languages that are used to help software agents to process ontologies easily.

The underlying goal of ontology development is to create artefacts that different systems or applications can share and use to interact with one another (Noy, 2005). It is necessary to develop general formal ontologies that can be shared among different pervasive computing systems. We followed with a description of the most developed ontology methodologies in a discussion of ontology engineering. We described the best practices for developing ontologies and a number of design principles to gauge the value of an ontology implementation. The above development techniques were introduced as a starting point for developers. By following these approaches for developing and evaluating ontologies, developers will be able to create ontologies that better facilitate sharing and reusing knowledge.

3 Ontologies in pervasive computing

In this section, we review a number of the most recent pervasive computing systems that use ontology models, including CoBrA (Chen *et al.*, 2004b), Gaia (Ranganathan *et al.*, 2004b), GLOSS (Coutaz *et al.*, 2003), ASC (Strang *et al.*, 2003b), and SOCAM (Gu *et al.*, 2004). All of them try to build heavyweight domain ontologies for pervasive computing. We introduce these systems focusing on a number of perspectives: the infrastructure and its characteristics, the use of ontologies and ontology-related components, and the scenarios that demonstrate their ontology-modelling approaches. These systems and their ontologies will be analyzed and evaluated in Section 4. In Section 3.6, we list some application ontologies in use in the areas of trust, policy management, and location. We also introduce some recent pervasive computing applications that apply ontologies.

3.1 CoBrA

The Context Broker Architecture (*CoBrA*) is a broker-centric, agent-based architecture for supporting context-aware computing in intelligent spaces (Kagal *et al.*, 2001). CoBrA provides a means of acquiring, maintaining, and reasoning about context; sharing knowledge; detecting and resolving inconsistent knowledge; and protecting user privacy (Chen *et al.*, 2005). All of the above capabilities are provided by the context broker, an intelligent agent that is the central component in CoBrA. Inside the context broker, the *Context Acquisition Module* offers a library of procedures that form a middleware abstraction for context acquisition; the *Context Knowledge Base* maintains a shared model of context on behalf of a community of agents and devices in the smart space; the *Context Reasoning Engine* reasons over the context so as to detect and resolve the inconsistent knowledge; and the *Policy Management Module* provides a set of user-defined inference rules that deduce instructions for deciding the right permissions for different computing entities to share a particular piece of contextual information, and for selecting the recipients to receive notifications of context changes.

Contextual information in CoBrA is represented by a set of ontologies called COBRA-ONT that is implemented in OWL. CoBrA-ONT is the key requirement for modelling context in the smart meeting application (Chen *et al.*, 2004b). It defines typical concepts and relations for describing physical locations, time, people, software agents, mobile devices, and meeting events. A set of more general ontologies, named SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications), has been proposed for supporting pervasive computing applications. SOUPA (Chen *et al.*, 2004e) borrows terms from other standard domain ontologies such as FOAF (Friend Of A Friend) (Dumbill, 2002), DAML-Time (Hobbs & Pans, 2004), OpenCyc (Lenat & Guha, 1989), RCC (Region Connection Calculus) (Borgo *et al.*, 1996), and the Rei Policy Ontology (Kagal *et al.*, 2003). These ontologies have gained consensus within their respective communities. Additionally, SOUPA references specific pervasive computing ontologies like CoBrA-ONT and the MoGATU BDI (Belief, Desire, and Intentions) Ontology (Perich *et al.*, 2004).

SOUPA offers a formal and well-structured way to model context, and thus provides rich semantics for programming. It also allows policies to be defined to support trust and privacy. This is demonstrated in CoBrA's EasyMeeting application (Chen *et al.*, 2004a), in which the ontologies facilitate knowledge sharing and work with logic inference rules to reason about the context. SOUPA also applies the BDI ontology to express the beliefs, preferences, intentions, and desires of an agent or a user, which makes it possible to rank the priorities of plans and goals.

3.2 Gaia

Gaia is an infrastructure for smart spaces, which are pervasive computing environments that encompass physical spaces (Roman *et al.*, 2002). The main characteristic of Gaia is that it brings the functionality of an operating system to physical spaces. It employs common operation system

functions (including events, signals, file systems, security, and processes), and extends them with context, location awareness, mobile computing devices, and actuators. Using this functionality, Gaia integrates devices and physical spaces, and allows the physical and virtual entities to seamlessly interact.

Ontologies are introduced in Gaia as an efficient way to manage the diversity and complexity of describing resources (e.g. devices and services). First of all, they work as a system specification for configuration management by providing a standard taxonomy of the different kinds of entities (including applications, services, devices, users, and data sources). Therefore, these ontologies are beneficial for semantic discovery (McGrath, 2000), matchmaking (Trastour *et al*., 2001), interoperability between entities (McGrath *et al*., 2003b), and interaction between human users and computers (Ranganathan *et al*., 2004b). Additionally, the Gaia ontologies are used to make Gaia systems context-aware (Ranganathan & Campbell, 2003). They model contextual information including physical, environmental, personal, social, application, and system contexts. They describe the relations between different entities and establish axioms and constraints on the properties of the entities that must be satisfied. Gaia represents context in a predicate form, for example, *Location* (*chris*, *entering*, *room 3231*), where a subject belongs to the set of persons or things (e.g. 'chris'), the location predicate is a verb or preposition (like 'entering'), and a location, which may be a room or a building (e.g. 'room 3231').

In Gaia, the *Ontology Server* is responsible for loading and validating ontologies from DAML+OIL documents, and composing ontologies into a combined ontology for the entire system. It is also capable of serving logical queries to the *Knowledge Base* that represents the dynamically integrated ontology base. The *Ontology Explorer* is a graphical user interface (GUI) that allows users to browse and search ontologies, and to interact with other entities in the space (McGrath *et al*., 2003a). The ontology explorer offers a clearer understanding of the semantics of a system. This makes it easier to write rules that determine context-sensitive behaviour. The explorer also helps human users to interact directly with the constituent parts of a pervasive computing environment.

3.3 GLOSS

A GLObal Smart Space (GLOSS) is a software infrastructure to support the interactions between people, artefacts, and places, while taking account of both context and movement on a global scale (Dearle *et al*., 2003b). By exploiting the features of physical spaces, GLOSS uses people's location and movement as a source of task-level context and as a guide to provide appropriate information or services. Therefore, GLOSS facilitates the low-level interactions (such as tracking a user's location) that are driven by high-level contexts (such as a user's task).

GLOSS provides a large and diverse range of services that are deployed at geographically appropriate locations (Dearle *et al*., 2003a). Those location-aware services demonstrate how to detect, convey, store, and exploit location information based on the GLOSS ontologies.

GLOSS accommodates both service heterogeneity and evolution. The GLOSS ontologies describe a small set of concepts for a universe of discourse. These concepts provide the precise understanding of how services (physical and informational) are used and how users interleave various contexts at run time. This allows different services to be implemented without the duplication of basic mechanisms, and abstracts over specific details of technologies. In terms of software evolution mechanisms, GLOSS permits the dynamic rearrangement of low-level interconnection topologies and the components that they connect (Dearle *et al*., 2003b). Service evolution is driven by the GLOSS ontologies, which include explicit and implicit descriptions of high-level contexts (physical and computational).

The GLOSS ontologies are designed using a top-down approach, starting with the universal object that represents any entity in the system. This general object is sub-classed into all of the actors and artefacts that are significant in the GLOSS environment, including a person's profile, location, a mode of transport, time, and an activity (Coutaz *et al*., 2003). GLOSS provides

comprehensive and well-formed ontologies; its When and Where ontologies are built in an orthogonal structure that can be extended easily.

3.4 ASC

Aspect-Scale-Context (*ASC*) is a model for describing contexts and their relationships using ontologies as fundamental (Strang *et al.*, 2003b). A *context* is a set of contextual information characterizing entities (like a person, place, or a general object) relevant for a specific task in their relevant aspects. An *aspect* is a classification, symbol- or value-range, whose subsets are a superset of all reachable states, grouped in one or more related dimensions called *scales* (Strang *et al.*, 2003a). A scale specifies fine-grained representation formats for an aspect, for example, a distance aspect has multiple scales such as meter, kilometers, and nautical mile. The ASC model shows how contextual information may be used to characterize a state of an entity under a specific aspect.

Ontologies implemented in ASC facilitate service discovery and service interoperability on the context level. *CoOL*, the Context Ontology Language (Strang *et al.*, 2003a), is derived from ASC to facilitate ontology-based contextual interoperability. CoOL is divided into two subsets: the *CoOL Core*, which projects ASC model into various common ontology languages such as OWL and DAML+OIL, and F-Logic (Kifer *et al.*, 1995); and *CoOL Integration*, which is a collection of schema and protocol extensions as well as common sub-concepts of ASC. CoOL is used to enable context interoperability and context-awareness during service discovery and execution.

Strang *et al.* (2003a) describe an overall architecture of the ASC-based system that focuses on their context provider domain, which includes CoOL-based knowledge (i.e. facts and ontologies), rules, and an inference engine *OntoBroker* (Decker *et al.*, 1999). The inference engine uses rules to derive new knowledge from existing ontologies and facts, to validate consistency within one ontology, and to assert inter-ontology relationships.

3.5 SOCAM

The Service-Oriented Context-Aware Middleware (*SOCAM*) is an architecture that enables the building and rapid prototyping of context-aware services in pervasive computing environments (Gu *et al.*, 2004). The middleware abstracts various physical spaces from which contexts are acquired into a semantic space where contexts can be easily shared and accessed by context-aware services.

The CONtext ONtology (CONON) is an ontology-based context model, in which a hierarchical approach is adopted for designing context ontologies (Gu *et al.*, 2004). The ontologies include a common upper ontology for the general concepts in pervasive computing (such as person, location, computing entity, and activity) and domain-specific ontologies that apply to different sub-domains (like smart homes). The context model supports multiple semantic contextual representations like classification, dependency, and quality of context. Using CONON, two types of contextual reasoning tasks are supported: ontology reasoning with description logic, and user-defined reasoning by defining specific rules in first-order logic.

The CONON ontologies help to share a common understanding of the structure of contextual information from users, devices, and services so as to support semantic interoperability and reuse of domain knowledge. They also support efficient reasoning mechanisms so as to check the consistency of context and deduce higher-level, implicit context from raw context. A context-aware home scenario is implemented in the prototype system to demonstrate the use of CONON (Gu *et al.*, 2004).

3.6 Miscellaneous ontologies

Besides the above ontology-based systems, there are other systems that build application ontologies for the description of contextual information. The GAS (Gadgetware Architectural Style)

ontology aims to provide a common language for communication and collaboration among the heterogeneous devices that constitute a ubiquitous computing environment (Christopoulou & Kameas, 2005). The GAS ontologies, implemented in DAML+OIL, contain semantic descriptions and interrelations between basic components (*eGadgets*) of ubiquitous computing applications (which they call *eGadgetWorlds*). The capabilities of eGadgets are made visible through *plugs* and connected together by *synapses*. GAS supports the building of eGadgetWorlds by providing specific rules for plugs compatibility and eGadgets replacement feasibility. Such ontologies help to discover potentially available services when a component where the current service is located breaks down.

The CoDAMoS Project (Preuveneers *et al*., 2004) builds an adaptable and extensible ontology for creating context-aware computing infrastructures, ranging from small embedded devices to high-end service platforms. This ontology is classified into four basic concepts: *User*, *Environment*, *Platform*, and *Service* (Preuveneers *et al*., 2006). It works at both the lower level for context modelling and the upper level for the whole system specification. It is used for application adaptation, automatic code generation and code mobility, and generation of devices and specific user interfaces.

Some pervasive computing systems develop task-specific ontologies for particular topics. Three of the most popular topics are trust (such as Toivonen & Denker, 2004; Mcnamara *et al*., 2006), policy management (such as Kagal *et al*., 2003, 2006; Weeds *et al*., 2004; Sriharee *et al*., 2004), and location (such as Millard *et al*., 2004; Flury *et al*., 2004; Sashima *et al*., 2004; Tafat *et al*., 2004). Toivonen and Denker define ontologies to capture context-sensitive messaging that includes message sender, receiver, and mediating network. They focus on a messaging trust problem, that is, to determine the degree of trust the receiver of a message should assign to a message. McNamara *et al*. employ ontologies to describe the services based on quality of service and mobility. This will help the requester of a service or a resource to decide which provider to rely on, depending on their trustworthiness and mobility patterns. Kagal *et al*. define the Rein ontology that is a Web-based policy management framework. The Rein ontology has a relatively small base and includes a few powerful terms that define the access control domain, and allows policies and policy languages to be reused and extended. Among the location ontologies, Millard *et al*. (2004) model *Where* and *What* a user is doing at any given time. Flury *et al*. (2004) define the semantic basis of location information for device-based services in pervasive computing environments. Besides the above typical popular concepts, ontologies are more and more applied to express user preferences in order to provide more suitable and customized services (such as Held *et al*., 2002; Mylonas *et al*., 2006).

Several novel ontology-based applications in pervasive computing have also been developed, including context-driven service adaptation and mobility (Preuveneers *et al*., 2006), service composition (Robinson *et al*., 2004; Ni & Slomon, 2005), and health care (Fook *et al*., 2006).

3.7 Summary

The five surveyed pervasive computing systems above (CoBrA, Gaia, GLOSS, ASC, and SOCAM) define domain ontologies to represent the primitives of pervasive computing. These ontologies are used to represent, manipulate, program, and reason with context data. Other application- and task-specific ontologies were introduced, which aim to solve particular pervasive computing problems. In the next section, we will analyze the domain ontologies more carefully with respect to ontology-modelling best practices, and their contributions to the research themes outlined in Section 1.

4 Analysis and evaluation of ontology-based models in pervasive computing

We now analyze the use of ontologies in the pervasive computing systems described in the previous section from a number of perspectives. We start by comparing the ontologies from the

perspective of ontology engineering, and evaluating them using the criteria introduced in Section 2.4. After the analysis, we also offer some suggestions on each criterion. Next, we assess the systems from the perspective of the strategic themes outlined by the Disappearing Computer initiative: in the representation and manipulation of the fundamental contextual concepts in pervasive computing (in Section 4.2); in privacy and trust management (in Section 4.3); in service description, discovery, and matching (in Section 4.4); and in the themes of interaction design (in Section 4.5). Finally, we outline an additional theme, that of modelling uncertainty in pervasive computing (in Section 4.6), and summarize the analysis (in Section 4.7).

It should be noted that we do not perform an evaluation of the 'essential infrastructure' theme. This theme is related to the characteristics that a pervasive computing system should exhibit, such as hardware infrastructure for input and output interaction, communication infrastructure from the small to large scale, and core enabling middleware services (Skordas *et al.*, 2004; Streitz & Nixon, 2005). Since the development and deployment of ontologies and ontological engineering techniques are unable to tackle this theme, it falls outside the remit of this survey and is omitted.

4.1 Ontology modelling

Modelling the elements of a pervasive computing environment is the most visible part of applying ontologies in pervasive computing. Each of the surveyed systems makes significant contributions in modelling context. They each have various structures to organize contexts and have various conceptualizations. The extensibility of their structures and the quality of their conceptualizations reflect the strength of ontologies employed in their context-modelling process. They have encoded their ontologies using the standard ontology languages discussed in Section 2.3 (shown in Table 1). This section will illustrate their structures, and apply the ontological design principles introduced in Section 2.4.2 to assess their conceptualizations.

4.1.1 Evaluation of structure

SOUPA (Chen *et al.*, 2004e) organizes its ontologies in a radiating manner into SOUPA core and extension. The SOUPA Core ontologies define generic vocabularies (including *Person*, *Agent*, *Event*, *Space*, *Time*, *Action*, *Policy*, and *BDI*) that are universal for different pervasive computing applications. By extending the core ontologies, the SOUPA Extension ontologies define task-dependent vocabularies for supporting specific types of applications, and provide examples for future ontology extensions.

The CONON ontologies organize their upper ontology and lower domain-specific ontologies into a tree hierarchy. The upper ontology captures the general context knowledge contained in pervasive computing (using CompEntity, Location, Person, and Activity). The lower ontologies can extend the general concepts in the upper ontology, or define additional concepts to suit particular applications, such as home, office, and vehicle applications.

The ASC and GLOSS ontologies start building their ontologies with the basic aspects like Time, Place, and Event. They employ the top-down approach to capturing concepts: starting with the most general contexts first and then extending them into specific applications by sub-classing. The context can also be customized by composition.

The Gaia ontologies classify context in parallel structures depending on its nature: physical, environmental, informational, personal, social, application, and system contexts.

All the reviewed models do a good job in extensibility in that they provide abstract concepts in the domain. Application-specific ontologies can be extended by sub-classing from domain ontologies. We conclude by posing some questions to help developers to achieve extensibility: which structure is more suitable for extension, either extending from basic concepts or developing manifold contexts in parallel; what kinds of general classes should be decided to be extracted from the domain; how concepts are captured and described to make their meaning integral and distinct; and so on.

Table 1. The ontology languages used in the surveyed systems

Ontology-based Models	Ontology Languages
CoBrA (SOUPA)	OWL
Gaia	DAML+OIL
GLOSS	XML
ASC (CoOL)	OWL, DAML+OIL
SOCAM (CONON)	OWL
GAS	DAML+OIL
CoDAMoS	OWL

4.1.2 Evaluation of conceptualization

Clarity means that a term can be uniquely identified and distinguished from other terms through necessary and sufficient conditions. All of the reviewed ontologies specify the restrictions for some concepts. However, those restrictions are relatively simple, and the strength of ontologies in specifying rules has not been fully exerted. They concentrate on whether a term has a certain property or whether a cardinality restriction has been exceeded for a certain property. Our survey has shown that SOUPA is the best of the reviewed ontologies that define their terms explicitly. Other ontologies (like Gaia, GLOSS, CONON, and ASC ontologies) do not specify as many constraints or restrictions as the SOUPA ontologies. However, the CONON and ASC ontologies introduce some semantics (like classification, dependency, and quality) that are more expressive in defining certain terms (such as activity).

Coherence requires that a concept should be defined consistently. Each of the reviewed systems defines its terms in self-consistent ontologies, but mismatches exist with the other reviewed systems ontologies. This limits the ability to share information and communication between them. As suggested in Section 2.4, it is better to choose the most appropriate definitions for the terms from existing mature ontologies. Among the surveyed ontologies, only SOUPA defines appropriate meanings for concepts from the consensual ontologies, including DAML-Time, OpenCyc, and RCC.

Ontological commitment requires that an ontology should make just enough claims on a domain so that it may be general enough to be usable in any application in that domain. It is advantageous that classes, associated properties, and involved constraints should serve for all of the general problems in the domain. GLOSS's Where and When ontologies are general enough to satisfy this criterion. The Where ontologies only capture the intrinsic aspects of location including physical representation, geometric region, and symbolic representation, and their basic mapping relationships. Therefore, they are a good foundation for building application ontologies to describe specific situations in the location-related domain. For example, if this location ontology was applied to a particular application in an indoor environment, it could be customized by making more specific claims on it (e.g. establishing relationships between rooms). On the other hand, CONON's location ontology includes many application-specific concepts (such as temperature and lighting) in its location ontology, so its location ontology is not general enough to satisfy the criterion.

Orthogonality requires that the defined concepts should be mutually exclusive from each other, which makes it easier to share and reuse ontologies. SOUPA's person ontology demonstrates good orthogonality properties. A person can be described from different aspects such as their contact information or social relationships. Each aspect corresponds to an independent ontology.

Encoding bias requires that the general ontologies should be independent of specific symbol-level encoding. Among our reviewed systems' ontologies, the ASC location ontologies support only two coordinate reference systems [the World Geodetic System 1984 (WGS84) (NIMA, 2004) and the Gauss–Krueger coordinate system (DMA, 1989)].

Extensibility requires that ontologies should be extensible to allow them to be reused easily by other applications in a specific domain. For an ontology to be extensible, new terms should be easily

Table 2 Evaluation of the surveyed ontologies using ontology design principles

Ontology-based Models	CoBrA (SOUPA)	Gaia	GLOSS	ASC (CoOL)	SOCAM (CONON)
Coherence	√				
Clarity	√			√	√
Extensibility	√	√	√	√	√
Ontological Commitment	√		√		
Orthogonality	√		√		
Encoding Bias	√	√	√		√

integrated in the existing terminologies without much modification or confusion. Section 4.1.1 showed that all of the surveyed ontologies introduced here are extensible in their structures. However, when discussing the finer-grained parameters of extensibility (such as at the term-, or property-level), extensibility is mostly dependent on the satisfaction of the *ontological commitment*, *orthogonality*, and *encoding bias* criteria.

Table 2 shows the results of our analyses of the ontologies using the evaluation criteria. We find that they do not completely satisfy every requirement for a formal ontology. Our analysis shows that formal development methodologies should be used when developing ontologies for pervasive computing. They will help to abstract the concepts with orthogonal meaning, extract basic concepts, and specify explicit and appropriate definitions. A more detailed analysis of the surveyed systems' adherence to these criteria is given throughout the next section.

4.2 Representing context for pervasive computing

There are two main applications of ontologies in pervasive computing: modelling context and reasoning about it. Since context determines the behaviours in pervasive computing systems, it is necessary to make it clear what context is and what features it has. The definitions around context are evolving continuously. In Dey (2001), the definition of context puts emphasis on relevancy: any entity (including person, location, and artefact) is contextual if it is relevant to an interaction or application. Coutaz & Rey (2002) divides context into *primary context* that is closely related to users, tasks, and a period of time; and *peripheral context* that is not central to, but may have an impact on, the task. The peripheral context is classified into physical, social, system, and user environment. Most subsequent definitions of context follow on from these.

The challenge of modelling context is how to capture, process, and exploit it to provide the correct behaviour in the correct form to the correct user at the correct time in the correct place (Streitz & Nixon, 2005). To date, many contextual models have been published in the pervasive computing field (such as Strang & Linnhoff-Popien, 2004; Chen & Kotz, 2000). The earliest model introduced by Schilit *et al*. (1994) was the key-value model that borrowed the database mechanism to put the value of contextual information as an environment variable (or key). Later, the traditional software engineering methodologies were applied to represent the context: the *graphical model* (Henrichsen *et al*., 2002) like ORM (Object-Role modelling and UML) and the *object-oriented model* (Schmidt *et al*., 1999). The graphical model provides a clear and intuitive view of context by describing the facts and properties as nodes and the relationships between them as edges. The object-oriented models make use of their own features of encapsulation and reusability to introduce an efficient abstraction and classification mechanism for contextual modelling. Compared with these contextual modelling approaches, the logic model (Ghidini & Giunchiglia, 2001) does not concern itself about how the context is organized or represented. It provides a formal and abstract context model about how to reason with part of the potentially available context and how to solve the compatibility among different contexts. However, the ontological models have the advantages of the object-oriented and logic-modelling approaches. They provide a formal way to model context into well-structured terminologies, and also support formal reasoning

mechanisms by defining the axioms and constraints. Strang & Linnhoff-Popien (2004) point out that the ontological approach has been considered a most promising approach for modelling contextual information.

4.2.1 Context characteristics

Henrichsen *et al*. (2002) summarized the characteristics of contextual information: it is dynamic; it should be possible to represent it in different ways; it should take account of uncertainty; and it should cater for the inherent interrelationships of different pieces of context. Pervasive computing systems are usually interested in both the past and future states of the context, as well as the current state. Coutaz *et al*. (2005) point out that context is not simply a state but part of a process. It is not sufficient for the system to behave correctly at a given instant; it must behave correctly during the time in which users are involved in a process. Hence, the temporal characteristics should be associated with context while modelling.

Pervasive computing applications absorb contextual information from a wide range of sources. Contexts can be represented in different forms at different levels of abstraction. Then contexts can be classified into static and dynamic types, and the latter can be further classified into profiled, sensed, and derived types. Data in pervasive systems are characterized as having an inherent level of uncertainty. Thus, different types of contexts should be associated with the levels of confidence and reliability, which should lead to more accurate context reasoning. Section 4.6 describes this in more detail.

Context data are often associated with pre-defined rules. These rules specify how a result is derived by inferring from one or more other pieces of context information (Henrichsen *et al*., 2002). According to this correlation, a change in one fact may lead to an automatic change in another fact, which is called *fact dependency*. For example, if there was a dependency between a person's location and his/her activity, an activity could be inferred according to the person's current location and schedule. Not only may context dependency result in automatic adaption but it also could be used to check the consistency and integrity of contextual information. This may be achieved by integrating the existing information and the knowledge inferred from other dependent information.

We suggest representing multiple context classification, quality, and dependency relations when modelling context. Among all the reviewed ontologies, only the CONON and ASC ontologies support these particular characteristics when modelling context. The CONON ontologies introduce a new property element to capture the properties of context classification (e.g. 'defined' or 'sensed') that describes a facet of the provenance of the associated data and objects. They introduce a dependency property element ('dependsOn') to capture the existence of a reliance relationship between one entity to another. An extensible ontology for quality constraints is constructed with a number of quality parameters such as accuracy, resolution, freshness, certainty, and provenance. They capture the dimensions of quality relevant to the attributes of entities and interrelationships between entities.

The ASC model expresses contextual classifications through a quality aspect that consists of static, dynamic-profiled, dynamic-sensed, and dynamic-derived. Other basic quality *aspects* are also introduced such as time-stamp or period. The dependence relation is expressed by the corresponding intra/interoperations between the *scales* that a pair of *context* information is based on.

4.2.2 Analysis of key contexts

In this section, we evaluate the way that the most common elements of primary context in pervasive computing are represented. These are generally held to be *location*, *agent* or *person*, *time*, and *activity*. These concepts permeate all of pervasive computing, while common peripheral or task-specific contexts may be introduced for specific applications (such as music, weather, and settings of the room).

Location: Location is the most important form of context in pervasive computing systems today. It is also a surprisingly subtle concept: a person's or device's location may be specified in

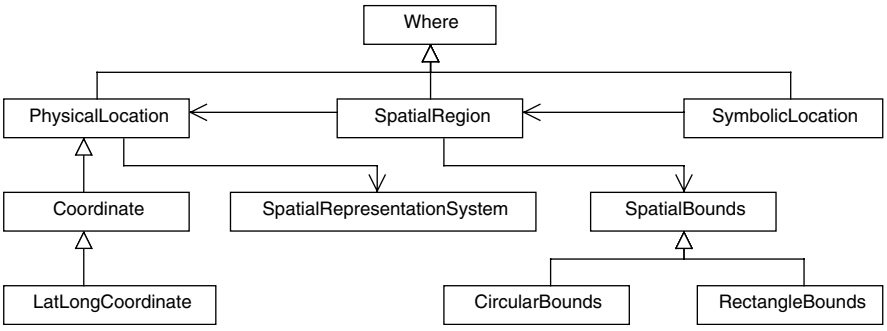

Figure 1 Location ontologies in GLOSS

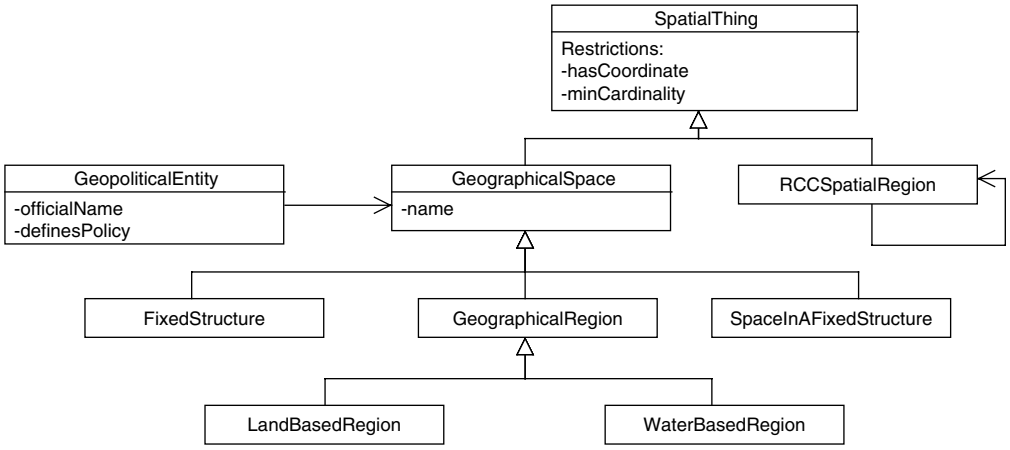

Figure 2 Location ontologies in SOUPA

a number of complementary ways, each of which may be considered 'optimal' for some class of applications (Dobson, 2005).

GLOSS's location ontology (illustrated in Figure 1) classifies location into three types: physical locations, spatial regions, and symbolic locations (Rey, 2005). The physical location corresponds to coordinate the representations of location with different spatial representation systems. The spatial region ontology is used to describe the geometric features (i.e. shapes) of a region. The symbolic location is the logical name for the space. These representations are related to each other: a symbolic location corresponds to one or more spatial regions; and a spatial region consists of a set of physical locations (i.e. coordinates). This structure is very general and can be flexibly extended to diverse representations of location. However, although GLOSS supports a rich structure for defining spatial locations and regions, it does not exploit the spatial relationships between defined locations.

SOUPA's location ontology (illustrated in Figure 2) is an update of CoBrA-ONT's location ontologies. SOUPA designs location using a top-down approach, with a general concept for location at the top. This general location object divides spatial information into geographical spaces and symbolic spaces (such as geo-political entities). Geographical spaces can specify particular policies to restrict the accessibility to the spaces. For instance, some places may be restricted with gender accessibility (e.g. toilets). This satisfies the orthogonality principle very well. Geographical spaces are extended into geographical regions (land-, or water-based), spaces with fixed structure, and spaces within the spaces with fixed structures. SOUPA can represent more spaces and provide richer semantics than GLOSS. Additionally, SOUPA's location ontology is based on the OpenCyc spatial ontologies and RCC. The former defines the vocabularies for expressing symbolic representation of space, and the latter for expressing spatial relations for qualitative spatial reasoning.

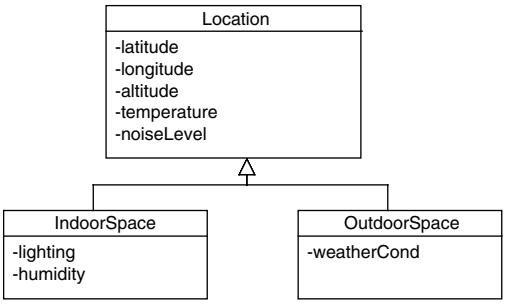

Figure 3 Location ontologies in CONON

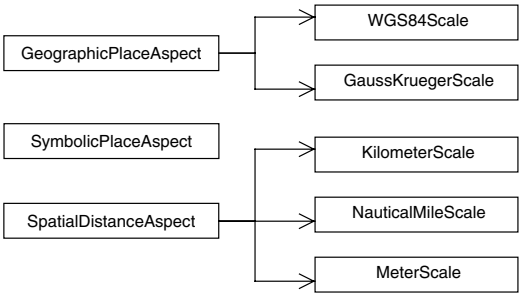

Figure 4 Location ontologies in ASC

Although location in SOUPA can be represented using coordinates, it ignores the possibility that there might be multiple spatial coordinate reference systems. If there are two different representations of location using different coordinate systems [such as a global and absolute representation scheme like GPS and a local and absolute scheme as used by Crickets (Priyantha *et al.*, 2000)], it will be hard to reconcile them with this location ontology.

CONON's location ontology (illustrated in Figure 3) is sub-classed to IndoorSpace and OutdoorSpace (Wang *et al.*, 2004). This has been done with particular applications in mind, that is, their smart home and outdoor applications—and is not flexibly extensible beyond them. Furthermore, location is described both by physical properties (such as longitude, latitude, altitude) and other properties, such as temperature, noise level, and lighting. The latter properties are environmental contexts and are only needed by particular situations. They should be defined separately in lower ontologies, such as environment or weather ontologies to satisfy the ontological commitment and orthogonality criteria.

ASC's location ontologies describe Place (illustrated in Figure 4) using three aspects: geographicPlaceAspect, symbolicPlaceAspect, and spatialDistanceAspect. The geographic aspect covers the geographic position information that supports two coordinate reference systems: WGS84 and Gauss-Krueger. Thus, a coordinate representation used in this location ontology is limited to these two types, which breaks the encoding bias criterion. The symbolic aspect is the string-based description for position information. However, it is hard to represent a physical region (such as the regions defined in GLOSS and SOUPA) using both aspects; to express spatial relationships; and to catch the mapping relation between geographical and symbolic spaces. The spatial distance aspect supports multiple distance scales such as meter, kilometer, and nautical mile. ASC's location ontology is the only one of the reviewed ontologies that introduce *distance* with different scales. However, the distance scales also break the encoding bias criterion, because they restrict the length representations into a number of limited (albeit popular) scales.

Among these location ontologies, we conclude that SOUPA's and GLOSS's location ontologies are better than CONON's and ASC's location ontologies. The former can support rich semantics of expressing location and spatial relationships. To illustrate, let us assume that there are two spaces, each of which can be described as a cube with a set of characteristic coordinates, and

can be labelled with a symbol. CONON's and ASC's location ontologies can express the spaces with coordinates and symbols only. However, the coordinates chosen to describe a space mostly depend on the shape of the space. For instance, coordinates for a cube space are that of eight corner points of the space; coordinates for a sphere space are that of a centre point of the space. Therefore, CONON's and ASC's location ontologies do not provide region locations, which makes their coordinate representation intractable and hard to catch the geometric properties of a space. SOUPA's and GLOSS's location ontologies can easily support the multiple location representations. However, if an adjacency spatial relationship between the two spaces must be expressed, SOUPA's location ontologies are more suitable than those of GLOSS.

In conclusion, we suggest some characteristics that a general ontology for location should support:

1. The ability to represent different types of locations including
   - *Coordinate locations*: A location can be physically represented with a set of coordinates. Currently, a number of location-sensing systems can provide coordinate representations when tracking an object's position (Hightower & Borriello, 2001). The coordinate values offered by these location-sensing technologies may be absolute or relative, and they may use different reference systems (such as WGS84). A general location model should be able to consume diverse types of coordinates to support multiple reference systems without restricting itself to any of them.
   - *Regions*: A region can be represented geometrically in two- or three-dimensions. They can be described with coordinates in different ways according to different shapes. SOUPA's region ontology is a good example of this.
   - *Symbolic locations*: A symbolic location encompasses abstract ideas of where an object is, for example, 'in Computer lab 003', 'next to a mailbox', or 'on a train approaching Dublin' (Hightower & Borriello, 2001). A general location ontology should support a logically spatial entity that can offer human-friendly labels. Symbolic locations are useful when referring to the locations from external contexts; SOUPA defines policies for locations referring to their symbolic labels rather than their physical definitions.
2. *The ability to map between physical and symbolic locations*: There should be a mapping between physical (both coordinates and regions) and symbolic locations. SOUPA's and GLOSS's location ontologies both provide good examples of these types of mappings.
3. *The ability to exploit rich spatial relationships*: Diverse spatial relationships should be exploited so that rich semantics about spaces can be expressed. In SOUPA's location ontology, regions support multiple spatial relationships using the RCC ontologies, which include the following two types:
   - *Hierarchical spatial relationships*, which define containment relationships, such as a space is contained within its super-space.
   - *Adjacency spatial relationships*, which define overlapping, adjacency, and disjointedness relationships. These relationships will allow connectedness relationships to be determined and help with path-finding algorithms. Additionally, the concept of distance could be introduced to better quantify the level of connectedness between locations (ASC includes such a distance metric).

Agent/Person: The agent/person ontology is used to describe actors in a system. Compared to the contexts that have relatively fixed and well-understood properties (e.g. location or time, which have their groundings in physics), an agent or a person context is more subjective. Systems that concentrate on different scenarios specify different roles and characteristics for a person or agent. This introduces subjectivity when choosing the properties to describe these ontologies, and each of the reviewed systems uses different approaches based on their individual perspectives.

SOUPA's person ontology is the most comprehensive of the surveyed ontologies for describing people. It provides an encompassing ontology to define an agent, which includes both human and software agents (or computing entities). A computing entity is characterized by a set of mentalistic notions such as knowledge, belief, intention, and obligation. The properties of a person agent involve the basic profile information (including name, gender, age, and date of birth), and the contact information (including email, mailing address, phone number, and homepage) (Chen *et al.*, 2005).

SOUPA references several classic domain ontologies in its agent and person ontologies, like the FOAF and BDI ontologies. The FOAF ontology expresses and reasons about a person's contact profile and social connections with other people. It allows the creation of information systems that support online communities for various people. The BDI ontology describes an abstract semantic model for representing and computing over a user's or an agent's profile in terms of their prioritized and temporarily ordered actions, beliefs, desires, intentions, and goals (Perich *et al.*, 2004). SOUPA uses this model to help independently developed agents to share a common understanding of their 'mental' states, so that they can cooperate and collaborate. The agents also help to reason about the intentions, goals, and desires of the human users of a system.

CONON has two separate agent classes: one for a computing entity, and one for a person. The computing entity ontology has properties reflecting the services, applications, devices, network capabilities, and agents in a pervasive computing system. The person ontology models a human being with the basic person profile (like name, situation, and age) and the contact information (like home address). However, this person ontology is application-specific, and it is not flexibly extensible. GLOSS defines an actor ontology to describe any entity that acts on behalf of a system or a person. This ontology defines the sensors and actuators through which the actor interacts with the system. CoOL does not have specific ontologies for agents or people.

In conclusion, the person ontology is relatively subjective and application-specific, and it is hard to describe every aspect about a person in a general way. We believe the person ontologies should be built with the *orthogonality* and *ontological commitment* criteria in mind. A set of lower independent profile ontologies should be built, each of which would reflect the characteristics of one aspect of a model of a person. These profile ontologies can then be customized and combined to satisfy particular application requirements. SOUPA's person ontology is a good example of this, since it simply defines a person's identity and contact profiles. Other profile ontologies could be constructed to describe a person's health status, working information (such as job title or salary), social associations (by using the FOAF ontology), or preference information (by using the BDI ontology). Since these profiles are application-specific, they should not be defined in a general person ontology. Therefore, if a person ontology needs to encompass many possible profiles about people in general, it should define them orthogonally so that application developers can extend and customize them as they wish.

Time: The temporal ontology is used to describe different temporal representations and temporal sequence relationships. Time is an important concept, because it is closely related to most contexts, especially to activity ontologies (activity ontologies are discussed next). However, the temporal ontology does not ordinarily get much attention, since it is universally recognized as a relatively simple and fixed concept. Neither CONON nor ASC build a specific temporal ontology.

It would be beneficial to develop a formal temporal ontology to exploit the features and various relationships of time. Most current temporal ontologies only reflect the physical nature of time, such as an instant of time in UTC format (Chen *et al.*, 2004e), and segments of time between instants. To satisfy the encoding bias criterion, it should be possible to represent physical time in any format (i.e. not limiting the system strictly to UTC). SOUPA exploits diverse temporal relationships such as startsSoonerThan, startsLaterThan, startsSameTimeAs, endsSoonerThan, endsLaterThan, endsSameTimeAs, startsAfterEndOf, and endsBeforeStartOf. GLOSS's temporal ontologies introduce the concepts of symbolic time (such as morning, lunch time, and summer). These are important concepts for reasoning in scheduled applications. For example, we can add

an activity in such a description: 'meeting with Erica at 4 o'clock tomorrow'. The system could then recognize that tomorrow is one day after the current date, translate it into a physical time, and then execute further actions.

A good ontology for time should consider the temporal relationships existing between physical times, and support the mapping between the symbolic and physical time—GLOSS's temporal ontology has good examples of such mappings. Finally, it should reflect the assorted characteristics of time and temporal semantics—SOUPA provides a good example of this, defining temporal sequences.

Activity: An activity ontology should model any action that can be performed by an agent of a system. Activity itself is a composite context, which can combine multiple orthogonal contexts such as location, time, agent, and device. In SOUPA, the event ontology is a general concept for the occurrences of activities, schedules, and sensing events (Chen *et al.*, 2004). The fundamental contexts of SOUPA's event ontology consist of a location and time. It is general and can be flexibly extended in particular situations. One example of an extension of this is SOUPA's *Meeting* ontology.

CONON's activity ontology is classified into deduced activities (including movie, dinner, shower, and cooking), which are inferred from known contextual information, and scheduled activities (such as party, meeting, or anniversary) that are explicitly profiled (Wang *et al.*, 2004). The activity ontology consists of the basic contexts: location; starting time and ending time; the computing entities that are used; and the people that are engaged in the activity. ASC's activity ontology use these contexts and also make it possible to extend activity to include dependency and activity type (i.e. static, dynamic-profiled, dynamic-sensed or dynamic-derived). GLOSS's activity ontology just models what a GLOSS-enabled person is doing at a particular time (Coutaz *et al.*, 2003).

Based on the above analysis, we outline some opinions about the activity ontology. The activity ontology should be general and comprise only the most basic and necessary contextual information (such as location and time). There exist various ways for classifying activities:

- The source of provided activity information: directly defined/sensed or indirectly by inferring (as done in CONON's activity ontology);
- The function of the activity, such as whether it is sensing, providing a service, or scheduling (as done in SOUPA's activity ontology);
- The duration of the activity: whether it occurs at an instant or over a time period;
- The location of the activity: whether it occurs in a static or moving location; whether it occurs indoor or outdoor.

Usually, an activity falls into multiple classifications; that is, an activity possesses more than one characteristic. For example, the activity 'our monthly research meeting' is provided by a calendar; it schedules the future work for a group; it usually lasts for an hour; and it occurs in the boardroom of our building. We suggest that a general activity ontology should support the above four orthogonal types. Thus a particular activity can be defined from different task-specific perspectives.

4.2.3 Reasoning about context

Typical pervasive computing environments are characterized as having large amounts of continuously changing contextual information. Pervasive systems must be able to perform context reasoning to facilitate dynamic adaptation to the changing environment, that is, to be context-aware. If context data are represented using ontologies, it would be possible to make context reasoning more powerful and precise, by using ontology reasoning mechanisms in such a system (Ranganathan & Campbell, 2003).

One of the key features of the CoBrA ontology is its ability to support ontology reasoning (Chen *et al.*, 2004b). Properties about a particular person, place, and activity can be described by distributed heterogeneous sources, and the contexts of these individual entities can be

dynamically inferred through classifications. First-order logic is used to infer facts from the existing relations. For instance, if a person is in a room, and that room is in the university, then CoBrA infers that the person is in the university. This is known as *part-of* reasoning, and has been widely used in pervasive computing using OWL's *part-of* relation, (Chen *et al*., 2003; Flury *et al*., 2004; Christopoulou & Kameas, 2005). CoBrA's Context Reasoning Engine uses these relationships to make inferences about the facts that are not explicitly stated in the knowledge base.

Gaia uses ontology models represented as context predicates to describe the individual components of a pervasive computing system. It uses a reasoning engine based on descriptive logic to make sure that these models are consistent when combined into a model of the system as a whole. It also allows the ontology server to answer logic queries about its ontologies, such as satisfiability, subsumption (e.g. whether a concept is subsumed by a given description), and equivalence (e.g. whether one concept is equivalent to another). Gaia also allows developers to define rules that determine context-sensitive behaviour using their ontologies.

CONON supports two aspects of context reasoning: checking the consistency of context, and deducing high-level, implicit context from low-level, explicit context (Wang *et al*., 2004). Context reasoning is performed using first-order predicates. The reasoning tasks are divided into two categories: with ontology reasoning using description logic (in a similar manner to that done by CoBrA); and user-defined reasoning using first-order logic.

The ASC model provides an inference engine (the *OntoBroker*) to infer conclusions about the context based on the ontologies built with CoOL. The reasoner can derive new knowledge about entities, aspects, scales and contextual information.

We agree with the analysis of Ranganathan *et al*. (2004b) that DAML (and its underlying description logic) is insufficient for reasoning about the context in pervasive computing. These logics neither deal well with the quantitative concepts, such as order, quantity time, and rate, nor with spatial models. In a sense, this is unsurprising: by using ontological models throughout pervasive computing, we require the logics capable of dealing with a wide (and indeed extensible) range of reasoning styles. OWL is derived from DAML+OIL, which makes some substantive changes such as the removal of qualified number restrictions, the ability to define symmetry of properties, and the absence in abstract syntax of some abnormal DAML+OIL constructs (Patel-Schneider *et al*., 2002). However, these extensions do not overcome the insufficiencies mentioned earlier.

4.3 Privacy and trust

The issues of privacy and trust are of paramount concern if pervasive computing applications are to become popular outside the research laboratory. CoBrA adopts a policy-based approach to protect user privacy. Policies are defined using SOUPA's policy ontology (Chen *et al*., 2004d). Using this ontology, users can define customized policy rules to permit or forbid different computing entities to access their private information. The policy reasoning algorithm uses a description logic inference engine and the description logical constructs of OWL to decide whether an action for accessing some user private information is permitted. Since it is often infeasible to define explicit policy rules for every individual action in a domain application, CoBrA uses meta-policies that determine the behaviour when policy rules are not defined. These meta-policies can be either conservative, in which case the system assumes all actions are forbidden; or liberal, in which case it assumes all actions are permitted.

Chen *et al*. (2004c) implemented a CoBrA prototype that supports privacy protection in an intelligent meeting room environment. They also demonstrated that SOUPA's policy ontology and its associated algorithms can be used to develop intelligent agents that can provide user privacy protection in a pervasive context-aware environment.

Although Gaia does not currently support rich trust or privacy concerns in their system, Ranganathan *et al*. (2004b) state that the topics of security, privacy, and access control must be addressed in future research of the Gaia system.

4.4 Discovery

One of the core problems in pervasive computing is the question of how to provide for new devices that wish to enter the environment. Since these may not be aware of the configuration of the environment or the services available, they must undergo a discovery or matchmaking process to best integrate themselves.

Gaia uses DAML+OIL to achieve semantic discovery, as it supports some of the operations required for semantic discovery. It also allows the definition of relations between concepts. The use of ontologies and semantic discovery replaced scripts and ad hoc configuration files that were used in Gaia previously. Each entity is associated with a document that describes its properties. Gaia's ontology server poses logical queries involving classification and subsumption of concepts to find appropriate matches. Other entities in the environment may query the ontology server to discover the classes of components that meet their requirements. Matchmaking uses ontologies to determine a set of concepts that fulfil the intersection of the requirements of two or more parties, such as a supplier and a consumer using the matching algorithms described by Trastour *et al.* (2001). The evaluation of whether a concept C_1 matches another concept C_2 is determined by an equivalence relation or a subsumption relation. C_1 is considered to *match* C_2, when C_1 is equal to or subsumed by C_2; or when they share a common sub-concept; or when they share a common super-concept without incompatibilities. The matched result is a set of classes that are semantically compatible to the query class.

The DAML-based Web Service ontology (DAML-S) supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of Web Services (Burstein *et al.*, 2001; Burnstein *et al.*, 2002) CoOL extends DAML-S with the service context to offer a more formal description of a service's contextual interoperability. The service context in CoOL consists of two parts: the context obligation, which specifies the obligations of a service in terms of the context of its execution; and the context binding, which is used to establish a virtual link from an atomic process of a service to a specific aspect of the context (Strang *et al.*, 2003a). This formal semantic service description offers a common understanding of the relations between services and their associated contexts (Strang & Linnhoff-Popein, 2003). This facilitates context-awareness and contextual interoperability during service discovery and execution.

Christopoulou *et al.* (2004) use their GAS ontology for discovery in the event of a component failure. All components have ontologies describing their interfaces and available services. If a connection (or synapse) between two or more components is broken, the ontology manager attempts to find an alternative component that offers the same service (i.e. that has the same ontological description) for the service to be resumed.

4.5 Interaction design

Pervasive computing environments are characterized as having many different interaction interfaces. The diversity of possible interfaces brings up two issues. One is the development of useful human–computer interactions (HCI) for end-users, and another is the exposure of these interfaces in a standard open way for developers of pervasive systems. A review of HCI for end-users is beyond the scope of this survey, since it is more closely related to other disciplines. Instead, we will focus on the interaction design between a pervasive computing system and its application developers.

Gaia uses ontologies to provide additional user interfaces to allow human users to interact with pervasive environments (Ranganathan *et al.*, 2004b). Their *Ontology Explorer* allows users to browse the ontologies that describe the environment. This is similar to a class browser, except that it has information about all the entities of the system, not just the software classes. Furthermore, the Ontology Explorer allows the human user to interact with the environment by exposing the queries and commands that can be made to its components (McGrath *et al.*, 2003a)—if a component is a database, queries may be sent and results returned; if the component exposes a command, it may be acted on.

By using ontologies to describe the different applications and the commands that can be sent to them, Gaia's Ontology Explorer simplifies the task of writing rules for context-sensitive behaviour. It allows developers to construct conditions out of the various types of contexts available and allows them to choose the action to be performed on these contexts. These actions can be chosen from the list of possible commands that can be sent to this application as described in the ontology. Developers can thus incorporate context-sensitivity to applications very quickly (McGrath *et al.*, 2003a). It should also be noted that the Gaia ontologies have human-understandable comments describing their function embedded in them, which are exposed to the Ontology Explorer.

4.6 Modelling uncertainty

Pervasive computing offers different challenges for ontology developers than conventional computer science applications. Data in pervasive computing environments may be generated by untrustworthy or inaccurate sources, and so should be taken 'with a grain of salt'. Because components of a pervasive computing environment deal with the real world, they come with certain caveats: sensors in the field are inherently inaccurate, since they could break down; or they could report inaccurately because they come up against a phenomenon for which they have not been designed. Wherever creating ontologies for real-world-sensed data (e.g. ontologies for the medical diagnosis and engineering domains), the issue of modelling uncertainty should be dealt with.

Since uncertainty must be taken into account when dealing with pervasive systems, it should be possible to describe the concepts of accuracy, confidence, uncertainty, and provenance with respect to context data, and to represent them as part of their ontological description. With these descriptions in place, particular reasoning mechanisms on ontologies need to be designed to support efficient and precise reasoning on the data.

In this survey, we found that only Gaia and CONON undertake this challenge. Gaia tries to capture and make sense of the imprecise and conflicting data inherent in dealing with real-world data (Ranganathan *et al.*, 2004a). An uncertainty model is developed based on a predicate representation of contexts and associated confidence values. The predicates' structure and semantics are specified in ontologies that can be used to check the predicates' validity, to simplify the definition of context predicates in rules, to facilitate interoperating between different systems, and to further reduce the possibility of uncertainty when interpreting context information. To reason about uncertainty, Gaia employs various mechanisms such as probabilistic logic, fuzzy logic, and Bayesian networks, each of which is advantageous under different circumstances. For instance, Gaia uses Bayesian networks to identify causal dependencies between different events. The networks are trained with real data so as to get more accurate probability distributions for their events.

CONON attempts to express the uncertainty of context by associating metadata that describes the quality of each context datum. Gu *et al.* (2004) have defined four types of quality parameters: accuracy, which reflects the estimated error of a measurement; resolution, which reflects the smallest perceivable element; certainty, which reflects the probability of the reading being accurate; and freshness, which reflects the time a measurement was generated and its expected lifetime. CONON also introduces a dependency tag that can be associated with any measurement. This allows relationships to be built between different contextual information, and makes it easier for developers to set down user-defined reasoning rules.

4.7 Summary

This section provided an analysis of the pervasive systems described in Section 3 with respect to their use of ontologies. We began in Section 4.1 with an analysis of how each of these systems' ontologies are engineered with respect to the recognized best practices in ontology engineering. Our conclusions (summarized in Table 2 in Section 4.1) were that all of the surveyed ontologies satisfy the extensibility criterion and most satisfy the encoding bias criterion. SOUPA is the

Table 3 Evaluation of the surveyed ontologies with respect to pervasive computing themes

Ontology-based Models	CoBrA (SOUPA)	Gaia	GLOSS	ASC (CoOL)	SOCAM (CONON)
Context and Programming	√	√	√	√	√
Privacy and Trust	√				
Discovery		√		√	
Interaction Design		√			
Modeling Uncertainty		√			√

Table 4 The themes that are targeted by the reference ontology-based models. (N.B. Gaia does not provide details of these core ontologies in their publications, so we cannot evaluate their ontologies and so leave the corresponding evaluation column blank.)

Ontology-based Models	CoBrA (SOUPA)	Gaia	GLOSS	ASC (CoOL)	SOCAM (CONON)
Location	√		√	√	√
Person/Agent	√		√		√
Time	√		√		
Activity	√		√	√	√

most consistent set of ontologies, since it imports many of its concepts from external, consensual domain ontologies. The SOUPA, CONON, and GLOSS ontologies satisfy the clarity requirement best, since they define the terms unambiguously with only the most necessary and sufficient conditions. When it comes to ontological commitment, GLOSS performs best—the other ontologies tend to mix higher- and lower-level concepts as well as concepts from different domains (which also reduces orthogonality).

Next, we analyzed the ontologies from the perspective of the themes for future research and development in pervasive computing. These are summarized below in Table 3.

Table 4 examines how each ontology tackles the theme of *Context and Programming* by describing how they are used to model the concepts of location, time, person (or agent), and activity. Since these higher concepts are so fundamental, the approaches used to describe them will have a significant impact on the modelling of lower-level ontologies. Each system has their particular advantages. Table 3 shows which key contexts are defined in each reference system. SOUPA defines rich temporal relationships in their time ontologies, and uses the popular BDI and FOAF ontologies to define the person concept. The GLOSS ontologies provide a formal structure for the time and location concepts. CONON and CoOL provide an interesting classification for activity ontologies, by distinguishing between deduced and scheduled activities. SOUPA, Gaia, CONON, and CoOL use reasoning techniques to ensure both the consistency of the model describing a pervasive computing system and to derive new knowledge from existing knowledge.

CoBrA uses ontologies to tackle the problems of *privacy and trust* by allowing users to define policies using ontologies to specify what data can be shared and with whom. When a user tries to access data, CoBrA first reasons about the relevant trust and privacy policies to determine whether access would be given. Both Gaia and CoOL use ontologies to manage *service discovery* in their respective systems. Gaia uses semantic Web matchmaking tools to determine a set of concepts that fulfil the intersection of the requirements of two or more parties (i.e. between users and services). CoOL associates contextual information with a service, so as to enable contextual interoperability during service discovery and execution. The interfaces of Gaia's components are modelled using ontologies, which are exposed to end-users. Gaia uses an ontology explorer to allow users and developers to directly *interact with the environment* by browsing through its ontological description. When it comes to managing the uncertainties inherent with pervasive computing, only CONON and Gaia *model uncertain data* with ontologies. Both of them make it

possible to attach metadata to a piece of context data, which describes the confidence that the system has in its value. This allows these systems to interpret and manage the uncertain data more accurately.

5 Conclusion

Ontologies are being increasingly applied in computer science to areas that require the exchange of information with significant structure and diverse semantics. Pervasive computing provides a particularly stark example of these characteristics, and in this paper we have reviewed a number of attempts to deploy the ontological techniques within pervasive systems. We evaluated the ontologies against the recognized criteria of both ontology engineering and pervasive systems design. We conclude that, while each provided important contributions, no single ontology was sufficiently rich to capture all the essential facets of pervasive systems.

Some of the deficiencies apply generally to many ontology applications. It is often hard to know the boundaries of the domain being modelled. If we assume a known boundary (the 'closed world' assumption), we may make stronger statements about the absence of particular information, may identify the attributes and relationships of interest *a priori*, and may perform stronger checks on completeness and consistency. Such bounding is often undesirable, however, and the 'open world' assumption makes it harder to reason about both the elements of interest and the knowledge base itself.

A clearer distinction needs to be drawn between upper (information exchange) and lower (information collection) ontologies. The former may subsume several of the latter, allowing diversity while retaining the ability to perform flexible queries, albeit with possibly reduced accuracy. It is reasonable to criticize several of the systems surveyed as attempting to provide a single system-wide description rather than building complexity through composition, as is intended within the Semantic Web.

Composition is clearly important within an open pervasive system. The need to describe privacy, security, and trust perhaps illustrates this most clearly: these are the emerging issues that affect all aspects of a system's collection, representation, exchange and use of information. This is clearly orthogonal to any particular context, and should therefore be specified separately; however, the need for trust may affect the design of other ontologies and the mechanisms used to access information.

However, the clearest omission concerns the certainty of information. Pervasive computing is typically highly sensor-driven, and both physical and virtual sensors (those dealing with online information) provide only *evidence of fact* rather than facts themselves. A person's location, for example, is not *captured* by a location sensor: rather, their location must be inferred by fusing the evidence available from a range of sources. The definition of various location ontologies does not capture this distinction between evidence and consensus; nor does it allow confidence measures to be assigned to information—both functions that are critical in real-world pervasive systems.

Finally, even the best ontology can only capture the information used to drive a system: it cannot capture the dynamic behaviour the system will exhibit as a result. Nevertheless, representing the information uniformly using ontologies should simplify the creation, composition, and analysis of pervasive computing systems. This allows us to gain greater confidence in the predictable behaviour of systems as their complexity grows. A proper focus on ontology engineering will help to facilitate this.

Acknowledgement

This work is partially supported by Science Foundation Ireland under grant numbers 05/RFP/CMS0062 'Towards a semantics of pervasive computing' and 04/RPI/1544 'Secure and predictable pervasive computing'.

The authors would like to thank the anonymous *Knowledge Engineering Review* reviewers for their constructive and detailed comments. Thanks also to Stephen Knox, Mikoláš Janota, Ross Shannon, and Adrian K. Clear.

References

Bachimont, B., Isaac, A. and Troncy, R. 2002 Semantic commitment for designing ontologies: a proposal. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*. London, UK: Springer-Verlag, pp. 114–121, ISBN 3-540-44268-5.

Bebee, B., Mack, G., Fritzson, A. and Weishar, D. 2003 Towards rich semantics in a grid architecture for information awareness. In *Proceedings of IEEE Aerospace Conference*, March 2003, vol. 6. pp. 2981–2990.

Bechhofer, S., Horrocks, I., Goble, C. and Stevens, R. 2001a OilEd: a reason-able ontology editor for the semantic web. In *Proceedings Joint German/Austrian Conference on Artificial Intelligence (KI2001)*, 2174 Lecture Notes in Computer Science. Vienna, Springer-Verlag, pp. 396–408.

Bechhofer, S. K., Goble, C. A. and Horrocks, I. 2001b DAML+OIL is not enough. In *Proceedings of the Semantic Web Working Symposium*. IEEE Computer Society Press, pp. 151–159.

Benjamins, V. R., Fensel, D. and Gómez-Pérez, A. 1998 Knowledge management through ontologies. In Reimer, U. (ed.), *Proceedings of the Second International Conference on Practical Aspects of Knowledge Management (PAKM-98)*, Vol. 13, CEUR Workshop Proceedings. CEUR-WS.org, 1998.

Bernstein, A., Provost, F. and Hill, S. 2005 Toward intelligent assistance for a data mining process: an ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering* **17**, 503–518.

Bonino, D., Corno, F., Farinetti, L. and Bosca, A. 2004 Ontology driven semantic search. *WSEAS Transaction on Information Science and Application* **1**, 1597–1605.

Borgo, S., Guarino, N. and Masolo, C. 1996 A pointless theory of space based on strong connection and congruence. In Aiello, L. C., Doyle, J. and Shapiro, S. (eds.), *Proceedings of Principles of Knowledge Representation and Reasoning*. San Francisco, CA: Morgan Kaufmann, pp. 220–229.

Borst, W. N. 1997 *Construction of Engineering Ontologies*. PhD thesis, Center for telematica and information technlogy, University of Tweently, Enschede, NL.

Brank, J., Grobelnik, M. and Mladenic, D. 2005 A survey of ontology evaluation techniques. In *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, Ljubljana, Slovenia.

Burstein, M. Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K. and Zeng, H. 2001 DAML-S Draft Release (May 2001). `http://www.daml.org/services/daml-s/2001/05/`.

Burstein, M. H., Hobbs, J. R., Lassila, O., Martin, D., McDermott, D. V., McIlraith, S. A., Narayanan, S., Paulucci, M., Payne, T. R., and Sycara, K. P. 2002 DAML-S: Web Service Description for the Semantic Web. In Horrocks, I. and Hendler, J. A. (eds.), *Proceedings of the First International Semantic Web Conference*. LNCS, Vol. 2342. London: Springer-Verlag, pp. 348–363.

Burton-Jones, A., Storey, V. C., Sugumaran, V. and Ahluwalia, P. 2005 A semiotic metrics suite for assessing the quality of ontologies. *Data Knowledge Engineering* **55**(1), 84–102.

Chandrasekaran, B. Josephson, J. R. and Benjamins, V. R. 1999 What are ontologies, and why do we need them? *IEEE Intelligent Systems* **14**(1), 20–26, ISSN 1541-1672.

Chen, G. and Kotz, D. 2000 A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

Chen, H., Finin, T. and Joshi, A. 2003 Using OWL in a Pervasive Computing Broker. In *Proceedings of the Workshop on Ontologies in Agent Systems (OAS 2003)*, Melbourne, Australia.

Chen, H., Finin, T. and Joshi, A. 2004a Semantic web in the context broker architecture. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PERCOM'04)*. Washington, DC: IEEE Computer Society, pp. 277–286, ISBN 0-7695-2090-1.

Chen, H., Finin, T. and Joshi, A. 2004b An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review* **18**(3), 197–207.

Chen, H., Finin, T. and Joshi, A. 2004c A Pervasive Computing Ontology for User Privacy Protection in the Context Broker Architecture. Technical Report TR-CS-04-08, University of Maryland, Baltimore County.

Chen, H., Finin, T. and Joshi, A. 2005 *Ontologies for Agents: Theory and Experiences*, chapter of The SOUPA Ontology for Pervasive Computing. Whitestein Series in Software Agent Technologies. Springer.

Chen, H., Perich, F., Chakraborty, D. Finin, T. and Joshi, A. 2004d Intelligent agents meet semantic web in a smart meeting room. In *Proceedings of the Third International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 2004)*, New York City, NY.

Chen, H., Perich, F., Finin, T. and Joshi, A. 2004e SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *Proceedings of the first International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA.

Christopoulou, E., Goumopoulos, C., Zaharakis, I. and Kameas, A. 2004 An ontologybased conceptual model for composing context-aware applications. In *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management*.

Christopoulou, E. and Kameas, A. 2005 Gas ontology: an ontology for collaboration among ubiquitous computing devices. *International Journal of Human-Computer Studies* **62**(5), 664–685, ISSN 1071-5819.

Colomb, R. M. 2002 *Quality of ontologies in interoperating information systems*. Technical Report ISIB-CNR-TR-18-02, Padova, Italy.

Corcho, O. Fernandez-Lopez, M. and Gomez-Perez, A. 2003 Methodologies, tools and languages for building ontologies: where is their meeting point? *Data Knowledge Engineering* **46**(1), 41–64, ISSN 0169-023X.

Coutaz, J., Dearle, A., Dupuy-Chessa, S., Kirby, G. N. C., Lachenal, C., Morrison, R., Rey, G. and Zirrintsis, E. 2003 Working document on gloss ontology. Technical Report D9.2, Global Smart Spaces Project IST-2000-26070.

Coutaz, J., Crowley, J. L., Dobson, S. and Garlan, D. 2005 Context is key. *Communications of the ACM* **48**(3), 49–53, ISSN 0001-0782.

Coutaz, J. and Rey, G. 2002 Foundations for a theory of contextors. In *Proceedings of the Computer-Aided Design of User Interface (CADUI 2002)*.

Dearle, A., Kirby, G. N. C., McCarthy, A. and Diaz y Carballo, J. C. 2003a An information flow architecture for global smart spaces. Technical Report D15, Global Smart Spaces Project IST-2000-26070.

Dearle, A., Kirby, G. N. C., Morrison, R., McCarthy, A., Mullen, K., Yang, Y., Connor, R. C. H., Welen, P. and Wilson, A. 2003b Architectural support for global smart spaces. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM'03)*. London, UK: Springer-Verlag, pp. 153–164, ISBN 3-540-00393-2.

de Bruijn, J. 2003 Using ontologies—enabling knowledge sharing and reuse on the semantic web. Technical Report DERI-2003-10-29, DERI.

Decker, S., Erdmann, M., Fensel, D. and Studer, R. 1999 Ontobroker: ontology based access to distributed and semi-structured information. In Meersman, R., Tari, Z. and Stevens, S. M. (eds.), *IFIP Conference Proceedings, DS-8*, vol. 138. Kluwer, pp. 351–369. ISBN 0-7923-8405-9.

Denny, M. 2002 Ontology Building: A Survey of Editing Tools. `http://www.xml.com/pub/a/2002/11/06/ontologies.html`.

de Vergara, J. E. L., Villagrá, V. A. and Berrocal, J. 2002 Semantic management: advantages of using an ontology-based management information meta-model. In *Proceedings of the 9th Workshop of the HP OpenView University Association (HP-OVUA)*.

Dey, A. K. 2001 Understanding and using context. *Personal and Ubiquitous Computing* **5**(1), 4–7, ISSN 1617-4909.

DMA. 1989 The universal grids: niversal Transverse Mercator (UTM) and Universal Polar Stereographic (UPS). Technical Report DMATM 8358.2, Defenses Mapping Agency.

Dobson, S. 2005 Leveraging the subtleties of location. In Bailly, G., Crowley, J. and Privat, G. (eds.), *Proceedings of Smart Objects and Ambient Intelligence*, pp. 175–179, New York, NY, USA 2005. ACM Press. ISBN 1-59593-304-2.

Dobson, S. and Nixon, P. 2004 More principled design of pervasive computing systems. *Humán computer interaction and interactive systems*, 3425 in Lecture Notes in Computer Science: 292–305.

Dumbill, E. 2002 Finding friends with XML and RDF: FOAF. `http://www-106.ibm.com/developerworks/xml/library/x-foaf.html`.

Eckstein, R., Tolksdorf, R. and Bizer, C. (eds.) 2004 *International Workshop on Semantic Web Technologies in Electronic Business (SWEB 2004)*.

Erdmann, M., Maedche, A., Schnurr, H.-P. and Staab, S. 2001 From manual to semiautomatic semantic annotation. In Linköping *Electronic Articles in Computer and Information Science*, vol. 6, (002), 2001. ISSN 1401-9841.

Fensel, D. 2001 *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, 1st edn. Springer, ISBN 978-3540416029.

Fernandez, M., Gomez-Perez, A. and Juristo, N. 1997 METHONTOLOGY: from ontological art towards ontological engineering. In *Proceedings of the Spring Symposium Series on Ontological Engineering*. Stanford, USA, pp. 33–40.

Flury, T., Privat, G. and Ramparany, F. 2004 OWL-based location ontology for context-aware services. In *Proceedings of the Artificial Intelligence in Mobile Systems (AIMS 2004)*, pp. 52–57. SFB 378-Ressourcenadaptive Kognitive Prozesse, September 2004.

Fook, V. F. S., Tay, S. C., Jayachandran, M., Biswas, J. and Zhang, D. 2006 An ontology-based context model in monitoring and handling agitation behaviour for persons with dementia. In *Proceedings of the Fourth*

Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM'06), Washington, DC, USA:2006. IEEE Computer Society, pp. 560–564, ISBN 0-7695-2520-2.

Ganesh, S., Jayaraj, M., Kalyan, V., Murthy, S. and Aghila, G. 2004 Ontology-based web crawler. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)* vol. 2, Washington, DC: IEEE Computer Society, pp. 337–341, ISBN 0-7695-2108-8.

Ghidini, C. and Giunchiglia, F. 2001 Local models semantics, or contextual reasoning = locality + compatibility. *Artifical Intelligence* **127**(2), 221–259, ISSN 0004-3702.

Gil, Y., Motta, E., Benjamins, V. R. and Musen, M. A. (eds.) 2005 *Proceedings of The SemanticWeb—the 4th International SemanticWeb Conference (ISWC'05)*, Galway, Ireland, November 6-10, vol. 3729 of Lecture Notes in Computer Science, Springer. ISBN 3-540-29754-5.

Gruber, T. R. 1993 A translation approach to portable ontology specifications. *Knowledge Acquisition* **5**(2), 199–220, ISSN 1042-8143.

Gruber, T. R. 1995 Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* **43**, 907–928.

Grüninger, M. and Fox, M. S. 1995 Methodology for the design and evaluation of ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI-95)*.

Gu, T., Wang, X. H., Pung, H. K. and Zhang, D. Q. 2004 An ontology-based context model in intelligent environments. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04)*, Society for Computer Simulation. pp. 270–275.

Guarino, N. 1998 Formal ontology and information systems. In *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems (FOIS'98)*, Trento, Italy, June 1998. IOS Press, pp. 3–15.

Guarino, N., Carrara, M. and Giaretta, P. 1994 Formalizing ontological commitment. In *Proceedings of National Conference on Artificial Intelligence (AAAI'94)*, Seattle, WA, pp. 560–567.

Guidetti, V. 2002 Intelligent Information Integration Systems: Extending a Lexicon Ontology. Master's thesis, Computer Science, University of Modena and Reggio Emilia.

Ha, Y.-G., Sohn, J.-C. and Cho, Y.-J. 2005 Owler: a semantic web ontology inference engine. In *Proceedings of the 7th International Conference on Advanced Communication Technology (ICACT'05)*, February 2005, vol. 2, IEEE Communications Society, pp. 1077–1080, ISBN 89-5519-123-5.

Haarslev, V. and Möller, R. 2003 Racer: A core inference engine for the semantic web. In Sure, Y., Corcho, O. (eds.), *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON'03)*, volume 87 of CEUR workshop proceedings, pp. 27–36. CEUR-WS.org 2003.

Held, A., Buchholz, S. and Schill, A. 2002 Modeling of context information for pervasive computing applications. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*.

Henricksen, K., Indulska, J. and Rakotonirainy, A. 2002 Modeling context information in pervasive computing systems. In *Proceedings of the First International Conference on Pervasive Computing (Pervasive '02)*, London, UK, 2002. Springer-Verlag, pp. 167–180, ISBN 3-540-44060-7.

Hightower, J. and Borriello, G. 2001 Location systems for ubiquitous computing. *IEEE Computer* **34**(8), 57–66.

Hobbs, J. R. and Pan, F. 2004 An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)* **3**(1), 66–85, ISSN 1530-0226.

Jones, D., Bench-Capon, T. and Visser, P. 1998 Methodologies for ontology development. In *Proceedings of the IT and KNOWS Conference, XV IFIP World Computer Congress*, Budapest, August 1998.

Kagal, L. Korolev, V., Chen, H., Joshi, A. and Finin, T. 2001 Centaurus: a framework for intelligent services in a mobile environment. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCSW'01)*, Washington, DC: IEEE Computer Society, pp. 195–201, ISBN 0-7695-1080-9.

Kagal, L. Finin, T. and Joshi, T. 2003 A policy based approach to security for the semantic web. In *Proceedings of the second International Semantic Web Conference (ISWC 2003)*, Sanibel Island, Florida, USA.

Kagal, L., Berners-Lee, T., Connolly, D. and Weitzner, D. J. 2006 Using semantic web technologies for policy management on the web. In *Proceedings of The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI'06)*.

Kalfoglou, Y. and Schorlemmer, M. 2003 Ontology mapping: the state of the art. *Knowledge Engineering Review* **18**(1), 1–31, ISSN 0269-8889.

Kifer, M., Lausen, G. and Wu, J. 1995 Logical foundations of object-oriented and frame-based languages. *Journal of ACM*, **42**(4), 741–843, ISSN 0004-5411.

Lenat, D. B. and Guha, R. V. 1989 *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc., ISBN 0201517523.

López de Vergara, J. E., Villagrá, V. A., Berrocal, J., Asensio, J. I. and Pignaton, R. Semantic management: application of ontologies for the integration of management information models. In *Proceedings of the Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, pp. 131–134.

McGrath, R. E. 2000 Discovery and its discontents: discovery protocols for ubiquitous computing. Technical Report UIUCDCS-R-99-2132, Department of Computer Science University of Illinois Urbana-Champaign, Urbana, March 2000.

McGrath, R. E., Ranganathan, A., Campbell, R. H. and Mickunas, M. D. 2003a Use of ontologies in a pervasive computing environment. Technical Report UIUCDCS-R-2003-2332 UILU-ENG-2003-1719, Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, Illinois.

McGrath, R. E., Ranganathan, A., Mickunas, M. D and Campbell, R. H. 2003b Investigations of semantic interoperability in ubiquitous computing environments. In *Proceedings of the 15th IASTED International Conference on Parallel And Distributed Computing And Systems (PDCS'03)*, *Marina del Rey, CA, USA*.

McGuinness, D. L. and van Harmelen, F. 2004 OWL web ontology language overview. W3C Recommendation, World Wide Web Consortium.

McNamara, L., Mascolo, C. and Capra, L. 2006 Trust and mobility aware service provision for pervasive computing. In *Proceedings of Workshop on Requirements and Solutions for Pervasive Software Infrastructures (co-located with Pervasive 2006)*, *Dublin, Ireland*.

Millard, I. C., Roure, D. D. and Shadbolt, N. 2004 The use of ontologies in contextually aware environments. In *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management, in association with Ubicomp2004, Nottingham*, *UK*, pp. 42–47.

Miller, E. 1998 An introduction to the resource description framework. *D-Lib Magazine*, ISSN 1082-9873.

Mostowfi, F., Fotouhi, F. and Aristar, A. 2005 Ontogloss: an ontology-based annotation tool. In *Proceedings of E-MELD Workshop on Morphosyntactic Annotation and Terminology, July 2005*.

Mylonas, P., Vallet, D., Fernández, M., Castells, P. and Avrithis, Y. 2006 Ontology-based personalization for multimedia content. In *Proceedings of the third European SemanticWeb Conference—Semantic Web Personalization Workshop*, *Budva, Montenegro, June 2006*.

Ni, Q. and Sloman, M. 2005 An ontology-enabled service oriented architecture for pervasive computing. In *Proceedings of the International Symposium on Information Technology: Coding and Computing (ITCC'05)*, vol. 2. Las Vegas, NV: IEEE Computer Society, pp. 797–798.

NIMA. 2004 Department of defense world geodetic system 1984—its definition and relationships with local geodetic systems. Technical Report TR8350.2, National Imagery and Mapping Agency, Bethesda, USA.

Noy, N. F., Fergerson, R. W. and Musen, M. A. 2000 The knowledge model of protege-2000: Combining interoperability and flexibility. In *Proceedings of the 2nd International Conference on Knowledge Engineering and Knowledge Management (EKAW 2000)*, *Juan-les-Pins, France, 2000*.

Noy, N. 2005 Order from chaos. *ACM Queue* **3**(8), 42–49, ISSN 1542-7730.

Obrst, L., Wray, R. E. and Liu, H. 2001 Ontological engineering for B2B E-commerce. In *Proceedings of the international conference on Formal Ontology in Information Systems (FOIS'01)*. New York, NY: ACM Press, pp. 117–126, ISBN 1-58113-377-4.

Pandey, S. K. and Mishra, R. B. 2005 Knowledge discovery and ontology-based services on the grid (a survey report). In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)*, December 2005, pp. 1033–1038.

Patel-Schneider, P. F., Hayes, P., Horrocks, I. and van Harmelen, F. 2002 Web ontology language (OWL) abstract syntax and semantics. W3C Working Draft, World Wide Web Consortium, November 2002.

Perich, F., Joshi, A., Finin, T. and Yesha, Y. 2004 On data management in pervasive computing environments. *IEEE Transactions on Knowledge and Data Engineering*, **16**(5): 621–634, ISSN 1041–4347.

Preuveneers, D. Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V. and De Bosschere, K. 2004 Towards an extensible context ontology for ambient intelligence. In *Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI 2004)*, pp. 148–159.

Preuveneers, D., Vandewoude, Y., Rigole, P., Ayed, D. and Berbers, Y. 2006 Context-aware adaptation for component-based pervasive computing systems. In *Adjunct Proceedings of the 4th International Conference on Pervasive Computing*, *Dublin, Ireland, 2006*, pp. 125– 128.

Priyantha, N. B., Chakraborty, A. and Balakrishnan, H. 2000 The cricket location-support system. In *Proceedings of the 6th ACM MOBICOM*, *Boston, MA, August 2000*.

Ranganathan, A. and Campbell, R. H. 2003 An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing* **7**, 353–364.

Ranganathan, A., Al-Muhtadi, J. and Campbel, R. H. 2004a Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing* **3**(2), 62–70, ISSN 1536-1268.

Ranganathan, A., McGrath, R. E., Campbell, R. H. and Mickunas, M. D. 2004b Use of ontologies in a pervasive computing environment. *Knowledge Engineering Review* **18**(3), 209–220.

Rey, G. 2005 *Contexte en Interaction Homme-Machine*: le contexteur. PhD thesis, Université Joseph Fourier.

Robinson, J., Wakeman, I. and Owen, T. 2004 Scooby: middleware for service composition in pervasive computing. In *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing (MPAC'04)*, New York: ACM Press, pp. 161–166, ISBN 1-58113-951-9.

Roman, M., Hess, C. K., Cerqueira, R., Ranganathan, A., Campbell, R. H. and Nahrstedt, K. 2002 Gaia: a middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, Oct–Dec 2002, pp. 74–83.

Sashima, A., Izumi, N. and Kurumatani, K. 2004 Location-aware middle agents in pervasive computing. In *International Conference on Wireless Networks*, pp. 820–828.

Schilit, B., Adams, N. and Want, R. 1994 Context-aware computing applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, *Santa Cruz, CA, December 1994*, pp. 85–90.

Schmidt, A., Beigl, M. and Gellersen, H.-W. 1999 There is more to context than location. *Computers and Graphics* **23**(6), 893–901.

Skordas, T., Tsarchopoulos, P., Ronchaud, R., Streitz, N., Nixon, P., Bannon, L., Coutaz, J., Gellersen, H., Kameas, A., Kyng, M. et al. 2004 EU-NSF joint advanced research workshop: The disappearing computer. Workshop Report and Recommendation., April 2004. `www.smartlab.cis.strath.ac.uk/EC-NSF/`.

Sriharee, N., Senivongse, T., Verma, K. and Sheth, A. P. 2004 On using ws-policy, ontology, and rule reasoning to discover web services. In Aagesen, F. A., Anutariya, C. and Wuwongse, V. (eds.), *Proceedings of Intelligence in Communication Systems, IFIP International Conference (INTELLCOMM 2004)*, vol. 3283 of Lecture Notes in Computer Science. Springer, pp. 246–255, ISBN 3-540-23893-X.

Staab, S., Studer, R., Schnurr, H.-P. and Sure, Y. 2001 Knowledge processes and ontologies. *IEEE Intelligent Systems* **16**(1), 26–34, ISSN 1541-1672.

Strang, T. and Linnhoff-Popien, C. 2003 Service interoperability on context level in ubiquitous computing environments. In *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet*, *L'Aquila/Italy, January 2003*.

Strang, T., Linnhoff-Popien, C. and Frank, K. 2003a CoOL: A Context Ontology Language to enable Contextual Interoperability. In Stefani, J.-B., Dameure, I. and Hagimont, D.(eds.), *Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003)*, LNCS 2893, Paris/France, November. Springer Verlag, pp. 236–247.

Strang, T., Linnhoff-Popien, C. and Frank, K. 2003b Applications of a context ontology language. In *Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCom'03)*, Split/Croatia, Venice/Italy, Ancona/Italy, Dubrovnik/Croatia, pp. 14–18. ISBN 953-6114-64-X.

Strang, T. and Linnhoff-Popien, C. 2004 A context modeling survey. In *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004*, *Nottingham/England, September 2004*.

Streitz, N. and Nixon, P. 2005 Introduction. *Communication of the ACM* **48**(3), 32–35, ISSN 0001-0782.

Sure, Y. and Domingue, J. (eds.) 2006 The semantic web: research and applications. *The 3rd European Semantic Web Conference*, *ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*, vol. 4011 of Lecture Notes in Computer Science. Springer, ISBN 3-540-34544-2.

Sure, Y. Staab, S. and Studer, R. 2002 Methodology for development and employment of ontology based knowledge management applications. *ACM SIGMOD Record* **31**(4), 18–23, ISSN 0163-5808.

Sure, Y., Gomez-,Perez, Daelemans, W., Reinberger, M.-L., Guarino, N. and Noy, N. F. 2004 Why evaluate ontology technologies? because it works! *IEEE Intelligent Systems* **19**(4), 74–81, ISSN 1541-1672.

Tafat, A., Courant, M. and Hirsbrunner, B. 2004 A generic coordination model for pervasive computing based on semantic web languages. In *Proceedings of the 9th International Conference on Applications of Natural Languages to Information Systems, Natural Language Processing and Information Systems (NLDB'04)*, Salford, UK, pp. 265–275.

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P. and Aleman-Meza, B. 2005 OntoQA: Metric-based ontology quality analysis. In *Proceedings of the IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources (ICDM'05)*, *Boston, MA*.

Toivonen, S. and Denker, G. 2004 The impact of context on the trustworthiness of communication: an ontological approach. In *Proceedings of the ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*, *Hiroshima, Japan*.

Trastour, D., Bartolini, C. and Gonzalez-Castillo, J. 2001 A semantic web approach to service description for matchmaking of services. Technical Report HPL-2001-183, HP Laboratories Bristol.

Uschold, M. and Gr uninger, M. 1996 Ontologies: principles, methods, and applications. *Knowledge Engineering Review* **11**(2), 93–155.

Uschold, M. and Jasper, R. 1999 A framework for understanding and classifying ontology applications. In *Proceedings of the Workshop on Ontologies and Problem-Solving Methods (as a part of IJCAI'99)*, *Stockholm, Sweden*.

Vargas-Vera, M., Domingue, J., Motta, E., Shum, S. B. and Lanzoni, M. 2001 Knowledge extraction by using an ontology-based annotation tool. In *Proceedings of the Workshop Knowledge Markup and Semantic Annotation (K-CAP'01)*, *Victoria Canada*.

Varzi, A. and Vieu, L. 2004 Formal ontology in information systems. In *Proceedings of the Third International Conference (FOIS'04)*. Turin, Italy, IOS Press.

Wang, X. H., Gu, T., Zhang, D. Q. and Pung, H. K. 2004 Ontology based context modeling and reasoning using OWL. In *Proceedings of the Workshop on Context Modeling and Reasoning (CoMoRea'04)*, pp. 18–22.

Weeds, J., Keller, B., Weir, D., Owen, T. and Wakemna, I. 2004 Natural language expression of user policies in pervasive computing environments. In *Proceedings of the LREC Workshop on Ontologies and Lexical Resources in Distributed Environments* (*OntoLex 2004*), *Lisbon, Portugal*.

Weiser, M. 1991 The computer for the 21st century. *Scientific American* **265**(3), 94–104.

Zhang, P. and Li, Z. 2005 Ontology assisted web services discovery. In *Proceedings of the IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)*. Los Alamitos, CA: IEEE Computer Society, pp. 45–50, ISBN 0-7695-2438-9.