Towards an extensible context ontology for ambient intelligence

Peer-reviewed author version

Preuveneers, D.; VAN DEN BERGH, Jan; Wagelaar, D.; Georges, A.; Rigole, P.; CLERCKX, Tim; Berbers, Yolande; CONINX, Karin; Jonckers, V. & De Bosschere, K. (2004) Towards an extensible context ontology for ambient intelligence. In: Second European Symposium on Ambient Intelligence. p. 148-159.

DOI: 10.1007/b102265
Handle: http://hdl.handle.net/1942/1739

Towards an extensible context ontology for Ambient Intelligence

Davy Preuveneers[1], Jan Van den Bergh[2], Dennis Wagelaar[3], Andy Georges[4], Peter Rigole[1], Tim Clerckx[2], Yolande Berbers[1], Karin Coninx[2], Viviane Jonckers[3], and Koen De Bosschere[4]

[1] Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium,
{davy.preuveneers, peter.rigole, yolande.berbers}@cs.kuleuven.ac.be,
http://www.cs.kuleuven.ac.be
[2] Expertise Centre for Digital Media, Limburgs Universitair Centrum
Universitaire Campus, B-3590 Diepenbeek, Belgium,
{jan.vandenbergh, tim.clerckx, karin.coninx}@luc.ac.be,
http://www.edm.luc.ac.be
[3] System and Software Engineering Lab, Vrije Universiteit Brussel
Pleinlaan 2, B-1050 Brussels, Belgium,
dennis.wagelaar@vub.ac.be, vejoncke@info.vub.ac.be,
http://ssel.vub.ac.be
[4] Department of Electronics and Information Systems, Ghent University
St.-Pietersnieuwstraat 41, B-9000 Gent, Belgium,
{ageorges, kdb}@elis.UGent.be,
http://www.elis.UGent.be/paris

Abstract. To realise an Ambient Intelligence environment, it is paramount that applications can dispose of information about the context in which they operate, preferably in a very general manner. For this purpose various types of information should be assembled to form a representation of the context of the device on which aforementioned applications run. To allow interoperability in an Ambient Intelligence environment, it is necessary that the context terminology is commonly understood by all participating devices. In this paper we propose an adaptable and extensible context ontology for creating context-aware computing infrastructures, ranging from small embedded devices to high-end service platforms. The ontology has been designed to solve several key challenges in Ambient Intelligence, such as application adaptation, automatic code generation and code mobility, and generation of device specific user interfaces.

1 Introduction

Small portable devices, such as PDAs and mobile phones, are becoming more widespread. As a consequence, people are expecting the functionalities provided by these devices to increase. GSMs with a quite extensive amount of organizer software, games and multimedia services are no exception, rather they are rapidly

becoming a default asset in everyone's life. As devices grow more powerful, with respect to computing power and autonomy, we expect the software on such embedded devices to become more advanced too. Additionally, at home and at work, embedded systems start getting a foothold. Home automation systems for example are no longer the rare expensive gadgets they used to be. Observing these trends, the IST Advisory Group (ISTAG) [1] has concluded that within a few years, real *Ambient Intelligence* (AmI) environments will emerge. In such environments, devices will communicate and interact independently, without immediate user interaction. The devices will make decisions based on a variety of factors, including user preferences and the presence of other users in the near neighbourhood.

To accomplish this, devices need to be aware of contextual information within their environment. In order to sort out any information that may characterize the situation of a person or a computing device, it is a must to structure the large amount of data so that synthesizing of valuable information from varying sources is possible. The resulting structured data is called the *context* of the device. The context thus describes all the relevant information to allow software on a device to semi-automatically interact in a well-defined way with its environment. The context model proposed in this paper will be used in the CoDAMoS project [2] to solve several key challenges in the area of Ambient Intelligence by supporting context-driven adaptation of mobile services.

A short overview of the context requirements to support an Ambient Intelligence environment is given in section 2. In section 3 we describe related work on the modeling of context and their shortcomings. We then present our context ontology proposal in section 4 and end this paper with a conclusion and future work in section 5.

2 Requirements for Ambient Intelligence

The aim of AmI computing infrastructures is to provide intelligent services to the user by targeting software towards a specific context before delivery, and adapting it to a changing context after delivery. More specifically, it will require integration of state-of-the-art concepts within several computer science research domains, such as application adaptation, code mobility in nomadic environments, automatic code generation and context-aware user interfaces. Therefore, detailed context information should be provided to be able to accomplish these objectives, resulting in the following requirements for a basic context model:

R.1 Application adaptivity: With dynamic environments and changing contexts in mind, it is important that applications support some degree of adaptivity. Hence, up-to-date information about the user, available services and host platforms, network connectivity, time, location and other sensed data should be included in the context model to assist appropriate application adaptation.

R.2 Resource awareness: As resources on embedded devices are sometimes too limited to run certain services, sufficient information about maximum

and currently available resources, such as processing power, memory, battery life time and bandwidth, is needed to consider service adaptation or service relocation for lowering resource usage.

R.3 Mobile services: When the location of a user changes over time, whole services or parts thereof must be able to migrate almost instantaneously. Therefore, detailed information about the execution platform should allow autonomous migration when, for example, compatible virtual machines exist on two different platforms.

R.4 Semantic service discovery: Semantic discovery based on context information enhances key-value based matching protocols by automatically incorporating search criteria that are relevant for the current user or device.

R.5 Code generation: By specifying the operating system, drivers, software libraries and virtual machines on an embedded device, code generation can be used to generate a dedicated implementation of a high-level service specification to broaden the range of devices on which services can be deployed.

R.6 Context-aware user interfaces: Services at the end-user side that have to work within tight resource boundaries on mobile devices need user interfaces that are adapted to their context of use. User interfaces can further adapt dynamically if the context changes over time.

These requirements allow mobile services to be designed in a generic way, with functional variations to be generated for a range of platforms, but also so that they can adapt to context elements such as other services and resources available in their context.

3 Related Work

Context-awareness is a hot research domain, with interesting topics such as context modeling, formal context languages for specifying facts and interrelationships, and infrastructure support for querying and reasoning on contextual information using an inference engine.

The Context Ontology Language (CoOL) [3] is an ontology-based context modeling approach, which uses the Aspect-Scale-Context (ASC) model where each *aspect* (e.g. spatial distance) can have several *scales* (e.g. kilometer scale or mile scale) to express some *context* information (e.g. 20). Mapping functions exist to convert context information from one scale to another. CoOL is very useful for describing concepts with an inherent metric ordering such as in requirement *R.2*, though less practical for expressing *scales* for *aspects* as in requirement *R.1*. Chen *et al.* [4] propose a context broker architecture (CoBrA) using an ontology to describe *persons*, *places* and *intentions*. Less emphasis is put on the notion of services and related aspects, such as user interfaces and mobile devices on which these services are deployed, needed to fulfill the above requirements. Gu *et al.* [5] present a service-oriented context-aware middleware (SOCAM) based on a context model with *person*, *location*, *activity* and *computational entity* (such as a device, network, application, service, etc.) as basic context concepts. The

notion of mobile services seems to be beyond the scope of this context model. Henricksen and Indulska [6] propose a context model that describes context based on several types of facts (e.g. sensed, static and profiled) subject to constraints and quality annotations.

Some general description frameworks for expressing context are the Resource Description Framework (RDF) [7] and the Web Ontology Language (OWL) [8]. Other languages are built on top of these frameworks, but are more tailored to describing context. These include the Composite Capability/Preference Profiles (CC/PP) [9] and the User Agent Profiling Specification (UAProf) [10]. All have been used to specify context. Korpipää et al. [11] use RDF to describe sensor and derived sensor data on mobile devices. CC/PP was used by Indulska et al. [12], but found to be too limited to describe complex context models. OWL, on the other hand, allows the definition of more complex context models and is used in several approaches [3–5].

4 Extensible Context Ontology

Considering the fast evolution in the hardware and software industry, it is important that decisions made today regarding our context specification are adaptable and extensible. Thus, we should remain as conservative as possible, keeping open the options for change in our context model. We therefore opted to define a basic, generic context ontology[1]. Ontologies provide classes of objects, relationships and domain constraints on their properties. By mapping concepts in different ontologies, structured information can be shared. Hence, ontologies are good candidates to express meaning within our context specification.

4.1 General Overview

We determined four main entities around which we built our ontology. These are based around the most important aspects in context information, which are also, sometimes partially, discussed in [13–15]:

User: The user plays an important role within Ambient Intelligence. The appliances within its environment should adapt to the user, and not vice versa. Important properties include a user's profile, but also his preferences, mood and current activity.

Environment: The environment in which the user interacts is an important aspect of the context specification. It consists of time and location information, and environmental conditions, such as temperature and lighting.

Platform: This part is dedicated to the hardware and software description of a specific device. This includes among other things specifications of the processor, available memory and bandwidth, but also information about the operating system and other available software libraries.

[1] The current implementation of our context ontology in OWL can be found at www.cs.kuleuven.ac.be/cwis/research/distrinet/projects/CoDAMoS/ontology/

Service: Services provide specific functionality to the user. Specifying semantic and syntactic information sustains easy service discovery and service interaction using a well-defined service interface.

Every device will contain its own context specification with a full description of its provided services, while containing pointers to relevant information on devices in its environment. An overview of the proposed context ontology[2] is given in figure 1.

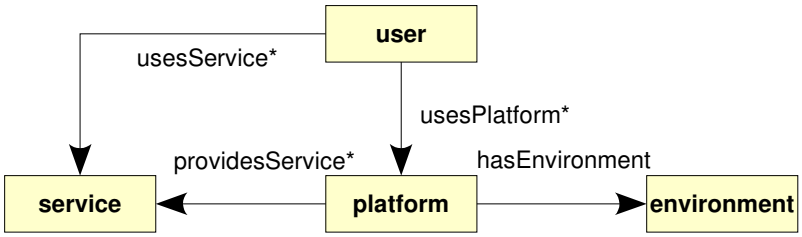

Fig. 1. Context ontology overview

4.2 User

According to Dey [15], context information is only relevant if it influences a user's task. This is why the *user* should take a central place in the Ambient Intelligence philosophy. Collecting information about his context enables applications and *services* to improve the usability of appliances. By accomplishing requirements *R.1* and *R.6*, it is possible to adapt the application as well as the user interface to the user's *preferences*. In the ontology a distinction is made between a user's preference, such as a preference for using small fonts, and his profile, containing facts such as gender, name and current employer. While the former may be subject to the current situation, the latter remains more or less static.

When a user performs a *task*, this can be subdivided into several *activities*. Clerckx *et al.* [16] show it is possible to link context information to a model describing the tasks a user can perform while using an application. The user fulfills a certain *role*, e.g. the project manager who is heading off to work for a meeting or the considerate father who picks up his children from school. Hence, people have different roles, but also different *moods*, and their personal preferences may depend on both issues. For example, consider the project manager, drowning in work, who does not want to be disturbed unless for urgent matters. Figure 2 shows the relevant user concepts and relationships.

[2] A (*) means a relationship with multiplicity of 1 or more.

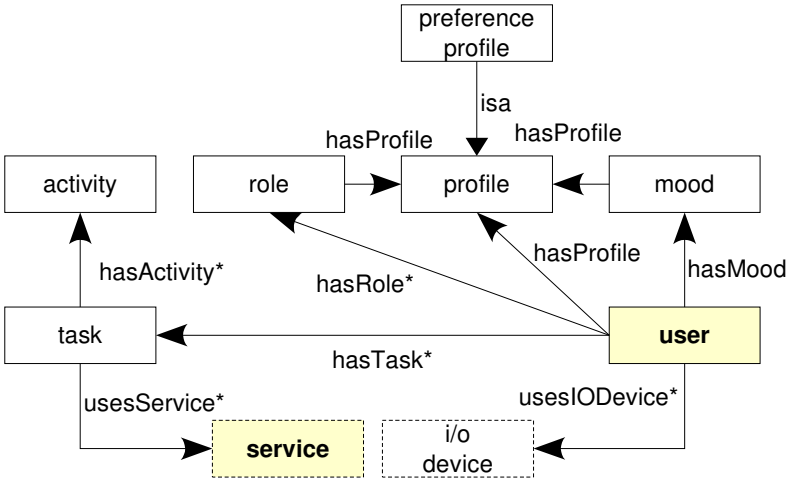

Fig. 2. User ontology concepts

4.3 Environment

A user is not a singular entity in an ambient environment. He interacts through various devices with his environment and with other people. This environment continuously provides information that allows him to make well-informed decisions or that can influence his behaviour. However, the diversity of entities that can be sensed or measured is enormous, if not infinite. It is therefore useless to try to describe everything within the surroundings of a user or a device. As user mobility is a key aspect within Ambient Intelligence, important concepts in this part of the context specification to meet requirements *R.1* and *R.3* include: *location*, *time* and some *environmental conditions*. For example, due to some cloudy weather and heavy rain outside, the home automation system might decide to turn on the lights. Of course, this is not needed in the middle of the night or if nobody is at home. Figure 3 gives an overview of the ontology concepts and relationships for the environment.

Another issue is that this information might be sensed by varying sources with different accuracies, with possibly conflicting measurements. It is very important that we are reasonably confident about the accuracy of the derived information within the context specification. Note that the environment is not directly related to the user, but rather through the used platform: The environment is always sensed through a device. By explicitly specifying this, it is possible to reason about several properties of the sensed environment that require knowledge of the measuring device, e.g. accuracy.

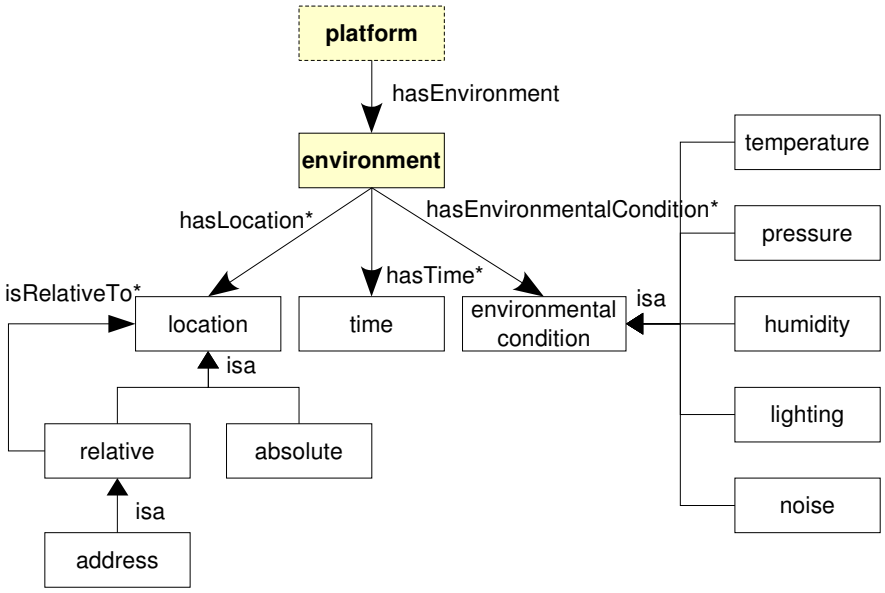

Fig. 3. Environment ontology concepts

4.4 Platform

The *platform* section of the ontology provides a description of (i) the software that is available on the device for the user or other services to interact with, and (ii), the hardware which specifies the resources of the device. Since the presence of certain hardware and software elements in devices can vary, only the relevant entries of the context specification are filled in. An overview of this part of the context specification is shown in Figure 4.

The software installed on the device The available software on a device is specified for the following reasons: (i) a service may require certain functionality to run, thus before deployment the service provider should be able to check for the presence of said functionality, and (ii) automated service builders must know for which software platform they are generating code. Hereby, we fulfill requirements *R.3* and *R.5*.

Software that is available on the device can be described by the following required parameters, or properties in the context specification:

Name: The software component name, e.g. *Java Media Framework*.
Edition: The software edition, if applicable, e.g. *Enterprise Edition*.
Version: The software version, e.g. *2.11*.

While we will in general generate code [17] for a high-level API, such as the Java API, it sometimes may be necessary to drop to a lower level, such as

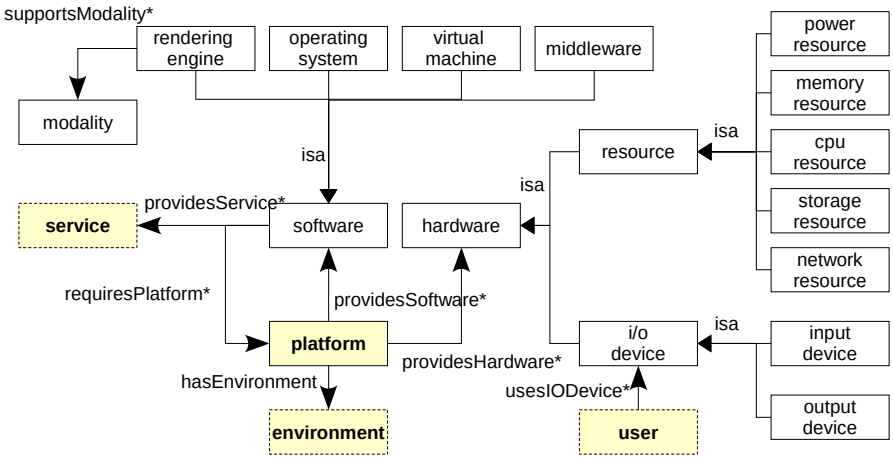

Fig. 4. Platform ontology concepts

the operating system or C-library. Therefore, we define the various software components, ranging from the lowest level to the high-level API's. By specifying an exact edition or version of the software components, we can generate code that is optimised for usage with these components.

Operating system: When code is generated for this level, it is necessary for the code generator to know about the API offered by the operating system (system calls) and e.g. the C-library (if any) present on the system. Examples are Windows CE 3.0 and Linux-2.4.19/glibc-2.3.2.

Virtual machine: If a virtual machine is present, the code generator should know what type of machine-independent representation the machine accepts and what API is offered by it. Examples are J2EE [18], J2ME [19] and .NET [20]. We also need to know the vendor and version of virtual machines. We have shown in [21] that the JVM can have a significant influence on the execution behaviour of a workload (JVM + application + input), especially for short or small applications.

Middleware: Besides the operating systems and virtual machines that are present, additional 'middleware' packages and libraries may have been installed as well, e.g. a CORBA broker [22].

Rendering Engine: This forms the backend for rendering a user interface on the particular device supporting at least one modality. Examples are QT, Java Swing and Windows Forms.

The hardware of the device For software deployment purposes, it is important that the context specifies the hardware in the device, such as the CPU type and properties, the available memory, networking capabilities, etc.

If one wants to deploy a piece of software, obviously it should fit on the device, both statically, and dynamically (at runtime). Furthermore, for e.g. multi-media applications, it is important that deadlines can be met. Consider for example the decoding of a video sample. The user wants smooth rendering of the video-frames, making it necessary to decode each frame in time. Thus, if the performance of the video decoding software is too low, these deadlines will not be met.

We distiguish five hardware resources that should be described in the context to accomplish requirements *R.1*, *R.2*, *R.3* and *R.5* for the device to support service mobility or service profiling: (i) the CPU, (ii) storage (permanent), (iii) memory (volatile), (iv) power, and (v) network capabilities. Each of these have several properties that are important for code-generation and for subsequent performance estimation. For the latter, we should know e.g. the cache and TLB size, the branch predictor used, etc., as they are used in the performance model we are developing. This model is needed to see if the generated code can actually run on the device, or if a simpler version should be instantiated.

4.5 Services

In several computer science domains the concept of services refers to a computational entity that offers a particular functionality to a possibly networked environment. Typical examples of where this term is used are in the domains of web services, telematics, residential gateways and mobile services. Although the previous domains target different users, they all have in common that these services are deployed to offer users a certain functionality using a well-defined interface, hereby providing a comfortable way for a user to achieve his goals. Our research is focussed on how services can dynamically interact and be aware of and be adapted to the current context, while keeping certain QoS aspects in mind. A user should be able to discover services in his environment and invoke them without too much hassle. This research involves requirements *R.1*, *R.2* and *R.3*. These services might be composed of other existing services and be adapted to personal preferences and to the device on which it is being employed. Hence service descriptions should be detailed enough to make this possible.

In figure 5 we give an overview of the main concepts regarding services. Typically, a user wants to employ a service to accomplish a specific task. He therefore interacts with some I/O device (a touchscreen, keyboard, voice recognition, etc.). Services will generally be implemented using software modules being provided on a device. Hence, each platform can host several services and/or employ several remote services in the neighbourhood when the necessary network infrastructure is present.

The level of detail at which services are described in the context specification of a device, depends on where these services are hosted. Each device is responsible for having a full description of its own services, including how it can be interfaced by other services. A high-level description of the services in its neighbourhood is more than adequate enough for doing service discovery using the context specification of the device to see if we are interested in a service and would

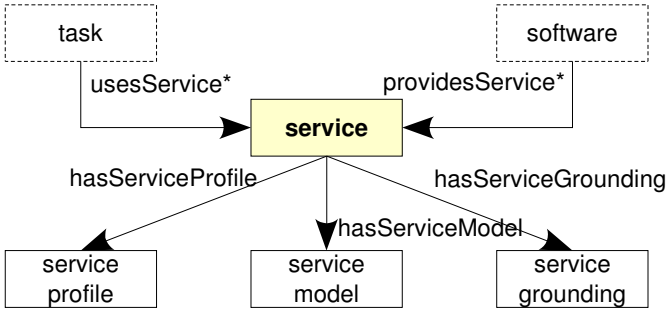

Fig. 5. Service ontology concepts

like to receive more detailed information about it. Information about required protocols and message formats can be negotiated later on if necessary, hence keeping bandwith usage to a minimum by only sending required information.

We therefore provide a multi-level service description, by extending our context ontology with a service ontology called OWL-s [23]. Although this ontology is tailored to web services and the semantic web [24], it also provides a rich and standardized framework to describe services in general. The Semantic Web community, using the OWL-s ontology specification, addresses the problem of having a lack of semantics within WSDL [25] service descriptions by adding a semantic layer based on the following concepts:

Service profile: It provides a human readable description of the functionality of the service by specifying its inputs and outputs, information about the service provider, a quality rating and other attributes that can be used for service discovery.

Service model: It describes what happens when the service is carried out, by giving more detailed information about the control-flow and data-flow involved in using the service so that the user or agent could perform an in-depth analysis of whether the service meets its needs.

Service grounding: The service grounding deals with implementation details by specifying a communication protocol, message formats, other service specific details.

5 Conclusion and Future Work

The necessity of ontologies for the establishment of context-aware pervasive computing systems is broadly acknowledged. In this paper, we presented a basic, generic ontology for the description of context information.

The ontology is currently expressed in OWL, but could also be expressed in other ontology languages. It consists of four basic context entities: (i) *user*, the central concept in context-aware computing, (ii) *environment*, the description of

relevant aspects of the user's surroundings , (iii) *platform*, the hardware and software of the device or devices through which a user interacts with the application or services and (iv) *service*, functionality offered in the user's environment.

Based on the gained experience and the feedback of industrial partners the context ontology will be further refined. Extensions, inevitable for the realization of concrete case studies for the CoDAMoS project [2], and refinements will be related to the presented basic ontology.

Further attention will be paid to how emerging standardized ontologies for various aspects of context information will relate to the established ontology. When needed for our research objectives or accomplishment of case studies, relations between the ontologies will be specified to enhance our current ontology.

6 Acknowledgements

The CoDAMoS (Context-Driven Adaptation of Mobile Services) project IWT 030320 is directly funded by the Flemish Institute for the Promotion of the Scientific-Technological Research in the Industry (IWT – Vlaanderen).

References

1. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: ISTAG, Scenarios for Ambient Intelligence in 2010. http://www.cordis.lu/ist/istag-reports.htm (2001)
2. The CoDAMoS Project: Context-Driven Adaptation of Mobile Services. http://www.cs.kuleuven.ac.be/distrinet/projects/CoDAMoS/ (2003)
3. Strang, T., Linnhoff-Popien, C., Frank, K.: CoOL: A Context Ontology Language to enable Contextual Interoperability. In Stefani, J.B., Dameure, I., Hagimont, D., eds.: LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003). Volume 2893 of Lecture Notes in Computer Science (LNCS)., Paris/France, Springer Verlag (2003) 236–247
4. Chen, H., Finin, T., Joshi, A.: An Ontology for Context-Aware Pervasive Computing Environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review (2003)
5. Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q.: An Ontology-based Context Model in Intelligent Environments. In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, California, USA (2004)
6. Henricksen, K., Indulska, J.: A Software Engineering Framework for Context-Aware Pervasive Computing. In: Second IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society (2004) 77–86
7. Beckett, D.: RDF/XML Syntax Specification (Revised). http://www.w3.org/TR/rdf-syntax-grammar/ (2003)
8. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.1 (2004)
9. Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M.H., Tran, L.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. http://www.w3.org/TR/2003/PR-CCPP-struct-vocab-20031015/ (2003)

10. FORUM, W.: UAProf User Agent Profiling Specification (1999, amended 2001)
11. Korpipää, P., Mätyjärvi, J., Kela, J., Keränen, H., Malm, E.J.: Managing Context Information in Mobile Devices. IEEE Pervasive Computing, Mobile and Ubiquitous Systems **2** (2003) 42–51
12. Indulska, J., Robinson, R., Rakotonirainy, A., Hendricksen, K.: Experiences in Using CC/PP in Context-Aware Systems. In Stefani, J.B., Dameure, I., Hagimont, D., eds.: LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003). Volume 2893 of Lecture Notes in Computer Science (LNCS)., Paris/France, Springer Verlag (2003) 224–235
13. Schilit, B.N., Adams, N.I., Want, R.: Context-Aware Computing Applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA, IEEE Computer Society (1994) 85–90
14. Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Laerhoven, K.V., de Velde, W.V.: Advanced Interaction in Context. In: Handheld and Ubiquitous Computing, HUC'99, Proceedings. Volume 1707 of Lecture Notes in Computer Science., Karlsruhe, Germany, Springer (1999) 89–101
15. Dey, A.K., Salber, D., Abowd, G.D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal **16** (2001) 97–166
16. Clerckx, T., Luyten, K., Coninx, K.: Generating Context-Sensitive Multiple Device User Interfaces from Design. In: Pre-Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces, CADUI'2004, 13-16 januari 2004, Edited by Robert J.K. Jacob, Quentin Limbourg and Jean Vanderdonckt, Funchal, Isle of Madeira, Portugal (2004) 288–301
17. Wagelaar, D.: Towards a Context-Driven Development Framework for Ambient Intelligence. In: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCS 2004 Workshops), IEEE Computer Society (2004)
18. Shannon, B.: Java^{TM}2 Platform: Enterprise Edition Specification. Sun Microsystems, Inc. (2001) Version 1.3.
19. Sun Microsystems, Inc.: Java 2 Micro Edition website. (2003) [Online] http://java.sun.com/j2me/.
20. Platt, D.S.: Introducing Microsoft .NET. 3rd edn. Microsoft Press (2003)
21. Eeckhout, L., Georges, A., De Bosschere, K.: How Java Programs Interact with Virtual Machines at the Microarchitectural Level. In: Proceedings of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2003), Anaheim, CA, USA, ACM (2003) 169–186
22. Object Management Group, Inc.: The Common Object Request Broker: Architecture and Specification. (2002) Version 3.0.
23. The OWL Services Coalition: OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.html (2003)
24. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. http://www.scientificamerican.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 (2001)
25. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315 (2001)