A Formally Specified Ontology Management API as a Registry for Ubiquitous Computing Systems

Alexander Paar[1], Jürgen Reuter[1]
John Soldatos[2], Kostas Stamatis[2], Lazaros Polymenakos[2]
[1]Institute for Program Structures and Data Organization (IPD),
Universität Karlsruhe (TH), Am Fasanengarten 5, 76128 Karlsruhe,
Germany
alexpaar@acm.org, reuter@ipd.uka.de
http://www.ipd.uka.de/Tichy
[2]Athens Information Technology
19,5 km Markopoulou Peania, Ave.
jsol@ait.edu.gr, ksta@ait.edu.gr, lcp@ait.edu.gr
http://www.ait.edu.gr

Abstract. Recently, several standards have emerged for ontology markup languages that can be used to formalize all kinds of knowledge. However, there are no widely accepted standards yet that define APIs to manage ontological data. Processing ontological information still suffers from the heterogeneity imposed by the plethora of available ontology management systems. Moreover, ubiquitous computing environments usually comprise software components written in a variety of different programming languages, which makes it even more difficult to establish a common ontology management API with programming language agnostic semantics. We implemented an ontological Knowledge Base Server, which can expose the functionality of arbitrary off-the-shelf ontology management systems via a formally specified and well defined API. A case study was carried out in order to demonstrate the feasibility of our approach to use an ontological Knowledge Base Server as a registry for ubiquitous computing systems.

1 Introduction

With the recent emergence of Semantic Web technologies like RDF(S) [1], DAML+OIL [2], and their common Description Logics (DL) [3] based successor OWL [4] numerous ontologies have been developed to conceptualize a plethora of domains of discourse [5]. This paper introduces an approach to model a ubiquitous

Please use the following format when citing this chapter:
Paar, Alexander, Reuter, Jürgen, Soldatos, John, Stamatis, Kostas, Polymenakos, Lazaros, 2006, in IFIP International Federation for Information Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 137–146

computing domain of discourse with the Web Ontology Language OWL. This effort was carried out in course of the CHIL research project [6], which builds non-intrusive services aiming to introduce computers into the loop of humans. In order to implement such services, a semantic middleware is being developed that fuses information provided by numerous perceptual components. Each perceptual component (e.g. image and speech recognizers, body trackers, etc.) contributes to the common domain of discourse. The Web Ontology Language OWL was used to replace previous domain models based on particular programming languages.

The CHIL software environment comprises perceptual components written in a variety of different programming languages. In contrast, existing ontology management systems typically provide only very limited connectivity with respect to natively supported programming languages and remoting protocols. The CHIL Knowledge Base Server [7] was developed to adapt off-the-shelf ontology management systems to a formally specified and well defined API. The Knowledge Base Server remedies three major shortcomings of existing ontology management systems. First, a number of programming languages are natively supported providing programming language specific client libraries. Second, almost arbitrary remoting protocols can be hosted in order to greatly improve connectivity compared to the majority of existing APIs that can only be used locally. Third, in contrast to existing ontology management APIs, the Knowledge Base Server interface specification relies on formally specified semantics. The latter feature is a main difference to related work on interface specifications of ontology management APIs such as the DIG protocol, the FaCT system or off-the-shelf ontology management systems such as Jena or Protégé.

The DIG protocol [8], which is a simple API for a general Description Logics system, is one representative of a class of interface definitions that consist of simple mechanisms to *tell* and *ask* DL knowledge bases. These mechanisms follow foundational aspects that have been well-studied over time [9]. Many previous frame-oriented knowledge representation systems such as the Generic Frame Protocol [10] and OKBC (Open Knowledge Base Connectivity) [11] also embody such distinctions. The DIG specification merely defines an XML schema that has to be used along with HTTP as the underlying communication protocol. There is no specific support for a particular programming language. Also, the KRSS specification [12], which is an earlier approach to define a number of tell- and ask operations that a DL system should implement, was tightly bound to the LISP [13] syntax, which may not be adequate for programmers who prefer other languages. Note also that other DIG 1.0 implementations (such as the FaCT reasoner [14, 15]), require further application server software. In [16] a CORBA interface to the FaCT system is proposed. Beyond the fact that CORBA may not be an appropriate remoting technology in today's service oriented- and XML based computing environments, the authors in [16] acknowledge that "*the CORBA IDL does not support the definition of the kinds of recursive data types that may be required for the representation of DL concepts and roles*". This is why an XML based workaround was devised to pass ontological concepts and roles as single data items. Previous approaches to augment DL knowledge base interfaces with remoting

capabilities include the wines- [17] and stereo [18] configuration demonstration systems.

Stanford University's Protégé [19], HP Labs' Jena [20], and Karlsruhe University's KAON [21] are all representatives for off-the-shelf ontology management systems, which do not rely on rigid formal semantics for their APIs specification. Moreover, none of these systems supports several programming languages and remoting protocols in order to cope with highly heterogeneous distributed computing environments (such as ubiquitous/pervasive computing environments).

Building on the advantages and merits of the Knowledge Base Server, this paper introduces a formal specification of an ontology management API for the Web Ontology Language OWL using a combination of the Z notation and Description Logics terminology. Moreover, it presents a case study that demonstrates the usefulness of a programming language agnostic remotable ontology management API in the domain of ubiquitous computing. The rest of the paper is structured as follows. Section 2 illustrates – for the most part by example – how the CHIL Knowledge Base Server API was formally specified using a combination of the Z notation and Description Logics terminology. A case study on the use of the CHIL Knowledge Base Server as a registry for a ubiquitous computing system is presented in Section 3. An overview of ongoing- and future work is given in Section 4, which includes also concluding remarks.

2 The CHIL Knowledge Base Server API

Based on the terminology for Description Logics as proposed in [3], formal specifications were devised for methods of the CHIL Knowledge Base Server interfaces `IAskingTBox`, `IAskingABox`, `ITellingTBox`, and `ITellingABox` for asking and telling the ABox and TBox of an OWL DL based knowledge base, respectively.[1] This formal specification was developed in order to make it possible to consistently adapt off-the-shelf ontology management systems. In particular, ambiguities had to be resolved that may be caused by informal specifications like "*This method returns all super classes of the given class*". In such cases it mostly remains unclear if the result set will contain the OWL top level concept `http://www.w3.org/2002/07/owl#Thing` or not.

With a more rigid specification, it would be clear if in an adapter class the top level concept from the result set of an adapted method had to be removed in case the underlying ontology management system yields it. In addition, a more formal specification is machine readable, such that result sets could be validated according to the specification.

For the formal specification of the Knowledge Base Server API, we use the Z notation [22, 23]. Since the API is specific to Description Logics, we added to the Z

[1] Java interface definitions and documentation can be found under *www.semantic-software.org*

notation the syntax and semantics of Description Logics as proposed in [3]. Additionally, the semantics of the ' $\sqsubseteq$'-sign, which in Z denotes a sub-bag relation, was over-written, such that it stands for the subsumption relation as defined by Description Logics.

In subsequent paragraphs we provide four examples on how methods from the Knowledge Base Server interfaces (`IAskingTBox`, `IAskingABox`, `ITellingTBox`, and `ITellingABox`) were formally specified.

The method `listDirectSubClasses(String owlClass)` from the `IAskingTBox` interface, which returns all classes that are directly subsumed by the given class `owlClass`, was defined as follows.

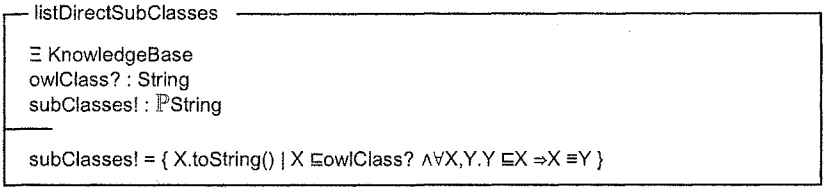

Fig. 1. Z notation of method listDirectSubClasses

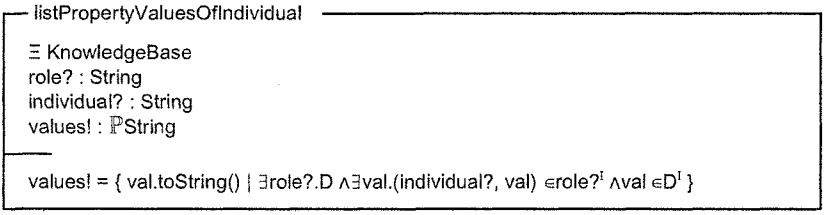

Fig. 2. Z notation of method listPropertyValuesOfIndividual

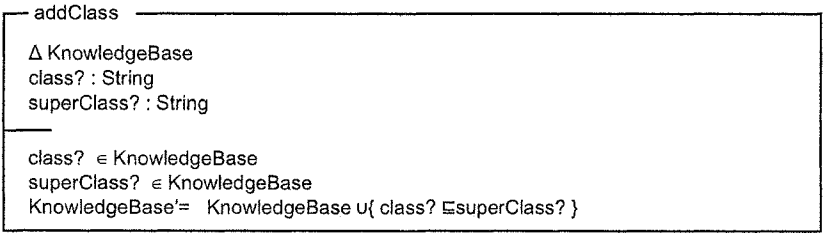

Fig. 3. Z notation of method addClass

The method `listPropertyValuesOfIndividual(String role, String individual)`, which is defined in the `IAskingABox` interface, yields all values of role `R` of individual `IND`. The result set returned by this method was defined as depicted above.
The interface `ITellingTBox` comprises methods that can be used to modify the set of terminological axioms, which are defined in a knowledge base. The method `addClass(String class, String superClass)` adds a class `class`, which is subsumed by class `superClass`, to the ontology. Accordingly, the axiom `class` $\sqsubseteq$ `superClass` where `class`^{I} $\subseteq$ `superClass`^{I} is added to the knowledge base as shown in Fig. 3.
The method `addPropertyValueOfIndividual(String role, String individual, String value)`, which is defined in the `ITellingABox` interface, can be used to add a role assertion as depicted in Fig. 4.

addPropertyValueOfIndividual

Δ KnowledgeBase
role? : String
individual? : String
value? : String

---

role? ∈ KnowledgeBase
individual? ∈ KnowledgeBase
KnowledgeBase'= KnowledgeBase ∪{ role?(individual?, value?) }
where (individual?^{I}, value?^{I}) ∈role?^{I}

Fig. 4. Z notation of method addPropertyValueOfIndividual

3 Case Study: The Knowledge Base Server as a Registry for Ubiquitous Computing Systems

The CHIL Knowledge Base Server provides directory services for ubiquitous context-aware computing services. Context-aware services acquire and process information about their surrounding environment, which allows them to execute service logic based not only on input explicitly provided by end users, but also based on information that is derived implicitly. Implicit information is usually derived based on a rich collection of casually accessible, often invisible sensors. Sensor information is processed by middleware components in order to derive elementary context information. Perceptive interfaces and recognition algorithms process audiovisual streams and extract context relating to the identity and location of people and objects. Accordingly, information fusion algorithms can be used to recognize more complex contextual states, which are often characterized as *situations*. Identification of contextual states provides a basis for triggering service logic [24].

Service logic is likely to comprise a rich set of invocations to soft-computing services, including commands to sensors and actuating devices. Thus, non-trivial ubiquitous applications consist of a rich set of sensors, middleware for controlling sensors and actuating devices, perceptual interfaces deriving context cues from sensor streams, as well as information fusion components identifying complex contextual states.

These hardware and middleware components are characterized by extreme diversity in terms of functionality, underlying technologies and vendors. Moreover, ubiquitous computing environments are very dynamic in the sense that sensors, devices, computing resources, and services are likely to dynamically join or leave [24]. Managing heterogeneity and dynamism is crucial to facilitate application development and deployment. Key to dealing with diversity and dynamic environments is a directory service maintaining and providing information about all sorts of components.

While several approaches to directory services middleware exist, the Knowledge Base Server supported by an ontology management system has clear advantages over conventional technologies. Technologies such as UPnP (Universal Plug n' Play), SLP (Service Location Protocol) and UDDI (Universal Description, Discovery and Integration) provide mechanisms for registering and discovering resource and services. However, these mechanisms are not particularly tailored to the range of information and components that are essential to ubiquitous computing services. For example, UDDI and SLP are merely service oriented, while UPnP is very much device oriented. Furthermore, the context-aware, human-centric, pervasive nature of ubiquitous computing services, asks for intelligence in answering queries. Intelligence lies in the ability to infer information from existing sets of meta-data according to current context and user intention. As an example, given the number of different sensors in a smart space, a particular situation model or service may need to acquire a reference to the best-view camera for a particular situation e.g., the camera facing the door for recognizing a person entering a room. Thus, the main benefit of such a knowledge base is its semantic power in knowledge conceptualization.

The Knowledge Base Server has been adopted as a core component of the CHIL architecture [24]. This architecture provides the structuring principles for the CHIL services and mandates that semantic middleware components, sensors, devices, services and resources are registered to the Knowledge Base Server. Sensors, actuators, and devices register their status and capabilities to the Knowledge Base Server upon their bootstrapping. Perceptual components can then discover the sensor streams required for their operation and accordingly register themselves with the Knowledge Base Server. CHIL perceptual technologies comprise a rich collection of 2D-visual components, 3D-visual perceptual components, acoustic components, audio-visual components, as well as output perceptual components like multimodal speech synthesis and targeted audio.

Situation models, which are higher level components of the context-aware semantic middleware, are registered with the Knowledge Base Server as well. Situation models define combinations of perceptual component values towards

identifying complex contextual states. Prior to registering themselves they acquire information on perceptual components.

Following the registration of semantic middleware components, the knowledge base provides 'yellow pages' services to middleware elements that need to interact with these components. Also, service logic implementation can leverage the knowledge base 'registry' to acquire binding on sensors, actuating devices and other resources entailed in service logic development. Thus, the Knowledge Base Server acts as an intelligent registry that provides information about components. Information includes a component's physical location in the network, its vendor, its functionality (i.e. the kind of ontological information it can provide), and its operational status.

As already outlined the expressive power of the underlying ontology management system is clearly manifested in cases where there is a need to access information that must/can be inferred rather than being readily available. As a characteristic example consider the query: 'provide a list of cameras facing the door', which the ontology management system can answer even in cases when the camera properties do not explicitly contain information about its relative orientation to the door.

Apart from the semantic capabilities of the CHIL Knowledge Base Server, other benefits of the particular implementation come into foreground in view of the architecture depicted in Fig. 5, in particular.

Platform independence: Given the large number of smart rooms, technology providers and services in CHIL, it is vital to have several ways to access the CHIL Knowledge Base Server. Indeed, in the scope of CHIL services, several components implemented in different languages and running on different operating systems need to access the Knowledge Base Server.

Independency from particular ontology management systems: Similarly to platform independence, the ability to use different ontology management systems proved to be essential since different smart room providers are likely to opt for different ontology management platforms.

The architecture depicted in Fig. 5 has been implemented in one of the CHIL smart rooms, namely the Athens Information Technology (AIT). Early instantiations of this architecture relied on hard-coded communication between hardware and middleware components. The introduction of the Knowledge Base Server has greatly facilitated integration. This is evident in the scope of the 'memory jog' implementation, which is a non-obtrusive service providing pertinent information and assistance in the scope of meetings, lectures, seminars and presentations [25]. The 'Memory Jog' uses the registration services of the knowledge base to dynamically discover and invoke audio- and vision based person tracking components.

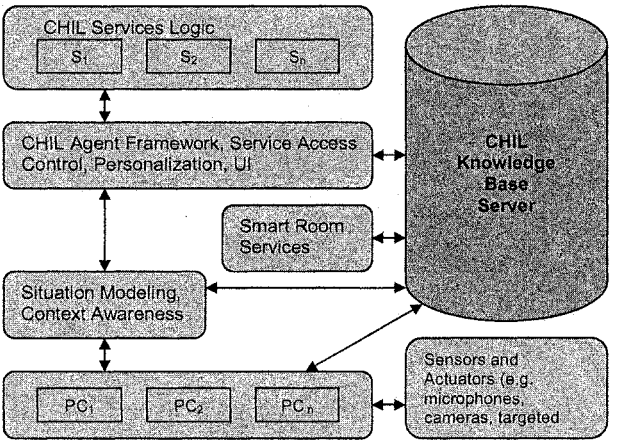

Fig. 5. The CHIL Knowledge Base Server as part of the CHIL semantic middleware

4 Conclusions and Future Work

We developed and implemented a pluggable architectural model for an ontological knowledge base server, which can be used to adapt off-the-shelf ontology management systems. Based on XML Schema Definition and on a combination of the Z notation and formal Description Logics terminology, a programming language independent API was defined. The API supports forwarding of exception information to clients in order to provide programmers with as much information as possible without being restricted to one particular programming language. The well defined ontology management API proved to be suitable both for developing auxiliary Eclipse plug-ins (e.g. for ontology visualization) and for accessing the Knowledge Base Server from a variety of perceptual components. Our case study, conducted in course of the CHIL project, showed that in order to benefit from common type systems defined by OWL ontologies, it is absolutely crucial to improve the connectivity of ontology management systems with a view to: (a) increasing their application scope and (b) to support a variety of different programming languages. The Knowledge Base Server proved to be a reliable backend for a semantic middleware that incorporates more than fifty (50) image- and speech recognition based perceptual components.

Current work on evolving the Knowledge Base Server is focused on further developing auxiliary Eclipse plug-ins that foster the integration of ontology engineering tasks in the software development process. In terms of using the Knowledge Base Server we envision additional applications, beyond the use of the Knowledge Base Server as a directory service. These include using the Knowledge

Base Server for inter-agent communication through establishing appropriate communication ontologies, as well as exploiting the Knowledge Base Server for reasoning on the whole range of concepts of the CHIL ontology. This could obviate the need for developing higher level context abstractions (e.g., Situation Models outlined in Section 3).

Acknowledgements

This work is part of the FP6 CHIL project (FP6-506909), partially funded by the European Commission under the Information Society Technology (IST) program. The authors acknowledge valuable help and contributions from all partners of the project.

References

1. W3C Recommendation: RDF Primer, http://www.w3.org/TR/rdf-primer/ (2004)
2. DARPA's Information Exploitation Office: DAML+OIL, http://www.daml.org/2001/03/daml+oil-index.html (2001)
3. Baader, F., Calvanese, D., McGuiness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook, Cambridge University Press (2003)
4. W3C Recommendation: OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/ (2004)
5. DARPA's Information Exploitation Office: DAML Ontology Library, http://www.daml.org/ontologies/ (2004)
6. Information Society Technology (IST) program FP6-506909, Computers in the Human Interaction Loop CHIL, http://chil.server.de/ (2004)
7. Paar, A., Reuter, J., Schaeffer, J.: A Pluggable Architectural Model and a Formally Specified Programming Language Independent API for an Ontological Knowledge Base Server, Australasian Ontology Workshop, Sydney, Australia (2005)
8. Bechhofer, S.:The DIG Description Logic Interface: DIG/1.0, University of Manchester, Oxford Road, Manchester M13 9PLA (2002)
9. Levesque, H.J.: Foundations of a functional approach to knowledge representation, Artificial Intelligence, 23 (1984) 155-212
10. Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D., and Rice, J.: The Generic Frame Protocol 2.0, Technical Report, Artificial Intelligence Center, SRI International, Menlo Park, CA (USA) (1997)
11. Chaudhri, V.K., Farquhar, A., Fikes, R., and Karp, P.D.: Open Knowledge Base Connectivity 2.0, Technical Report KSL-09-06, Stanford University KSL (1998)
12. Patel-Schneider, P.F., and Swartout, B.: Description-logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort, Technical report, AI Principles Research Department, AT&T Bell Laboratories (1993)
13. Graham, P.: ANSI Common LISP, Prentice Hall (1995)
14. Horrocks, I.: The FaCT system, Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX), volume 1397 of Lecture Notes in Artificial Intelligence (1998) 307-312

15. Horrocks, I.: FaCT and iFaCT, Proc. of the 1999 Description Logic Workshop (DL'99), CEUR Electronic Workshop Proceedings (1999) 133-135
16. Bechhofer, S., Horrocks, I., Patel-Schneider, P.F., and Tessaris, S.: A proposal for a Description Logic interface, Proc. of the 1999 Description Logic Workshop (DL'99), 33-36, CEUR Electronic Workshop Proceedings (1999)
17. Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnick, L.A., and Borgida, A.: Living with CLASSIC: When and how to use KL-ONE-like language, Principles of Semantic Networks, Morgan Kaufmann, Los Altos (1991) 401-456
18. McGuiness D.L., Resnick, L.A., and Isbell, C.: Description Logic in practice: A CLASSIC application, Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI) (1995) 2045-2046
19. Stanford University School of Medicine: Protégé knowledge acquisition system, http://protege.stanford.edu/ (2003)
20. HP Labs: Jena 2 - A Semantic Web Framework, http://www.hpl.hp.com/semweb/jena.htm (2004)
21. KAON 2, Universität Karlsruhe (TH), Germany, http://kaon2.semanticweb.org/ (2005)
22. Spivey, J.M.: The Z Notation: A Reference Manual, Prentice-Hall International Series in Computer Science Prentice Hall; 2nd edition (1992)
23. ISO/IEC, Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics, ISO/IEC 13568:2002 (2002)
24. Soldatos J., Pandis I., Stamatis K., Polymenakos L., Crowley J., 'A Middleware Infrastructure for Autonomous Context-Aware Computing Services', accepted for publication to the Computer Communications Magazine, special Issue on Emerging Middleware for Next Generation Networks (2005).
25. Soldatos, J., Polymenakos, L., Pnevmatikakis, A., Talantzis, F., Stamatis, K., Carras, M.: Perceptual Interfaces and Distributed Agents supporting Ubiquitous Computing Services. In: The Proc. of the Eurescom Summit 2005 (2005) 43–50.
26. Pandis I., Soldatos J., Paar A., Reuter J., Carras M., Polymenakos L., 'An Ontology-based Framework for Dynamic Resource Management in Ubiquitous Computing Environments', in the Proc. of the 2nd International Conference on Embedded Software and Systems, Northwestern Polytechnical University of Xian, P. R. China, December 16-18. (2005).