OWL-SF – Distributed OWL-based Reasoning on Objects in the Real World

Marko Luther[1], Bernd Mrohs[2], Raju Vaidya[2], Matthias Wagner[1]

[1]Future Networking Lab
DoCoMo Communications Laboratories Europe
Munich, Germany
<*lastname*>@docomolab-euro.com

[2]Fraunhofer Institute FOKUS
Open Communication Systems
Berlin, Germany
<*lastname*>@fokus.fraunhofer.de

Abstract

We present OWL-SF, a prototype for OWL-based situational reasoning in context-aware applications. The implementation addresses major challenges that arise from the distributed nature and heterogeneity of context-aware systems. OWL-SF scenarios can be conducted as real time simulations that connect to actual objects in the real world.

1. Motivation

Major challenges arise when realizing situation- and context-aware services. First, contextual information needs to be represented in ways that enables contextual reasoning. The next challenge concerns the distributed nature of context information. As a consequence, context data often must be collected from distributed context sources before reasoning can be initiated. The third challenge is to deal with the heterogeneity of context sources and service-providing devices. Devices delivering context information and providing services have different interfaces and computing capabilities. However, context providers and context consumers have to be able to communicate in a homogenous way. Furthermore, to support also resource limited devices, situation-aware service provisioning techniques should demand as little processing power as possible.

We are approaching these challenges through OWL-SF, a distributed OWL-based Service Framework, in combining emerging technologies: W3C's ontology language OWL [1] is used to capture high-level context elements in semantically well founded ways. Devices, sensors and other environmental entities are encapsulated using OMG's Super Distributed Objects technology [8] and communicate using the Representational State Transfer model (REST) [5]. OWL-SF is written in Java as a toolbox that simulates parts of an SDO-based environment and also connects to real world objects that are encapsulated as SDOs. Real world objects that are currently encapsulated include mobile phones, door panels, light sensors and switches. A concise evaluation of OWL-SF based on early case studies is published elsewhere [7].

2. Situational Reasoning

One important aspect in ubiquitous computing environments is to support people in their everyday life by providing situation-aware services. Information that can be used to characterize the situation of a person, place or object is usually called context [1]. Important providers of context information are personal and global information sources (e.g., personal information managers, travel information services), but also sources that provide local environmental information (e.g., proximity and light sensors). Bringing all context information together enables reasoning about people's situation. In situation-aware service provisioning, the detected situation is in turn used to offer the currently most relevant services to the customer. The offered service might even be further adjusted by taking additionally available context information into account.

Based on OWL-SF, scenarios for intelligent call forwarding are realized that target the office environment and demonstrate context-aware presence management to enable intelligent call forwarding. Demonstration scenarios are populated with several persons and mobile phones that hold different status and are placed in different locations. Depending on the identified context (location of a person, the people in proximity, the current time, etc.) the situation of selected actors can be classified and further used to configure intelligent call forwarding.

3. OWL-based Context Representation

To enable the intelligent use of context information for future applications, one key requirement is the proper description of context elements. Semantic-based descriptions permit the interoperability of applications by enabling advanced knowledge retrieval as well as profound reasoning in an adequate fashion. Therefore, semantic-based descriptions play a fundamental role in the ongoing development of the Semantic Web and we expect also the next generation of context-aware applications and services to profit substantially from an explicit semantic layer.

The actual situation of an actor and its availability for phone calls is decided by using concept classification. Figure 1 illustrates a fragment of the corresponding OWL context ontology. We exemplarily sketch the OWL definition of two common office situations using standard DL syntax [1]. An individual describing a person's situation is classified as *Working*, when it is a *Situation* and its property *hasCurrentTime* relates to an individual of type *OfficeHour*.

$$Working \equiv Situation \cap (\exists\, hasCurrentTime.OfficeHour)$$

A person is in a *Meeting* situation if she is *Working*, she has a location classified as *MeetingRoom* and he is together with other persons.

Meeting ≡ Working ∩ (∃ isSituationOf.
(∃ hasLocation.(MeetingRoom ∩ (≥ 2 contains))))

4. Architectural Considerations

OWL-SF used OMG SDO technology to encapsulate context sources [5]. Super distributed objects are logical representations of hardware or software entities that are used to enable distributed interoperability. The term super distribution denotes a system where a massive number of objects is interacting without a centralized control. The characteristics of an SDO are defined by its properties and the services it offers. A service consists of a group of operations and is provided to other SDOs or applications using the defined SDO interfaces. The operations not only provide access to the functionality of the underlying device, but may also include access to internal representations of the device. The respective SDO interface maps device capabilities to specific operations. This allows the encapsulation of software and hardware entities through providing standardized management interfaces for discovery, monitoring, configuration, and reservation, as well as a service interface to access the functions of the object.

The functional architecture of the full OWL-SF framework is sketched in Figure 2 with OWL-SDOs and Deduction Servers (DSs) as the main building blocks. An OWL-SF application may be composed of multiple components of both types which can be added and removed dynamically at runtime. Deduction Servers provide reasoning support based on the ontology structures collected from the accessible OWL-SDOs and perform service calls on the OWL-SDOs based on the derived conclusions. Complementary, OWL-SDOs implement a certain program logic that encapsulates the underlying hard- and software components providing an OWL-enhanced SDO interface. In general, components of both types could provide the same interface (i.e., the DS could support the OWL-SDO interface or an OWL-SDO might contain a local reasoner). However, as the aimed functionality of both is conceptually different they are treated separately in the architecture description.

5. Related Work

OWL-SF has been influenced by the Context Toolkit (CT) [4], which structures contextual environments into Context Widgets that encapsulate context sources, interpreters that raise the level of abstraction of pieces of context and aggregators that collect multiple pieces of context together. However, the CT can only provides weak support for knowledge sharing and high-level context reasoning as context information lacks formal semantic descriptions.

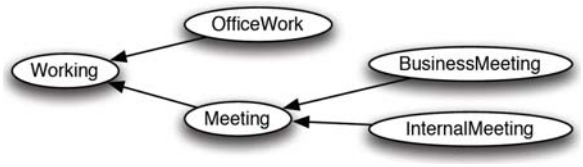

Figure 1: Fragment of the Situation Ontology

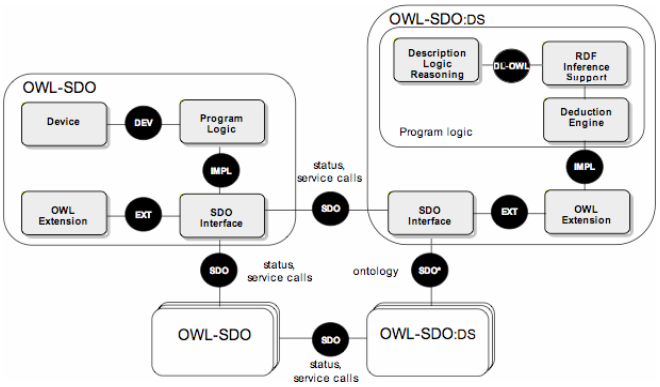

Figure 2: OWL-SF Functional Architecture

Other approaches such as CoBrA [3] and CALI [6] tackle these issues by using OWL descriptions, but do not define a fully distributed architecture. Both approaches assume one central reasoning engine. CoBra relies solely on rule reasoning which cannot be complete for OWL (not even for OWL-Lite). CALI tries to overcome the limitations of DL-based reasoning using a hybrid reasoning approach that tightly integrates DL and generic rule reasoning. However, such a tight combination might easily lead to undecidability as rules can be used to simulate role value maps. Furthermore, the initial results of our own experiments indicate that an ad-hoc approach to hybrid reasoning easily becomes computationally expensive.

Acknowledgements: This work has been partly performed in the framework of the IST project IST-2004-511607 MobiLife, which is partly funded by the European Union.

6. References

[1] F. Baader, D. Calvanese, D. McGuinness, et al. *The Description Logic Handbook.* Cambridge University Press, 2003.

[2] S. Bechhofer, F. van Harmelen, J. Hendler, et al. OWL Web Ontology Language reference. W3C Recommendation, 2004.

[3] H. Chen, T. Finin, A. Joshi. A context broker for building smart meeting rooms. In *Proc. of the Autonomous Systems Symposium.* AAAI Spring Symposium, 2004.

[4] A.K. Dey. Providing Architectural Support for Building Context-Aware Applications. PhD thesis, Georgia Institute of Technology (2000)

[5] T. R. Fielding. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, 2000.

[6] D. Khushraj, O. Lassila. CALI: context awareness via logical inference. In *Proc. of the Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications,* Hiroshima, Japan, 2004.

[7] B. Mrohs, M. Luther, R. Vaidya, M. Wagner, S. Steglich, W. Kellerer, and S. Arbanowski. OWL-SF – A Distributed Semantic Service Framework. In *Proc. of the 1st Workshop on Context Awareness for Proactive Systems (CAPS'05),* Helsinki, Finland, 2005.

[8] S. Sameshima, J. Suzuki, et al.: Platform Independent Model and Platform Specific Model for SDOs. Final recommended specification, OMG, 2004.

OWL-SF – Distributed OWL-based Reasoning on Objects in the Real World

Appendix: Demo Explanation

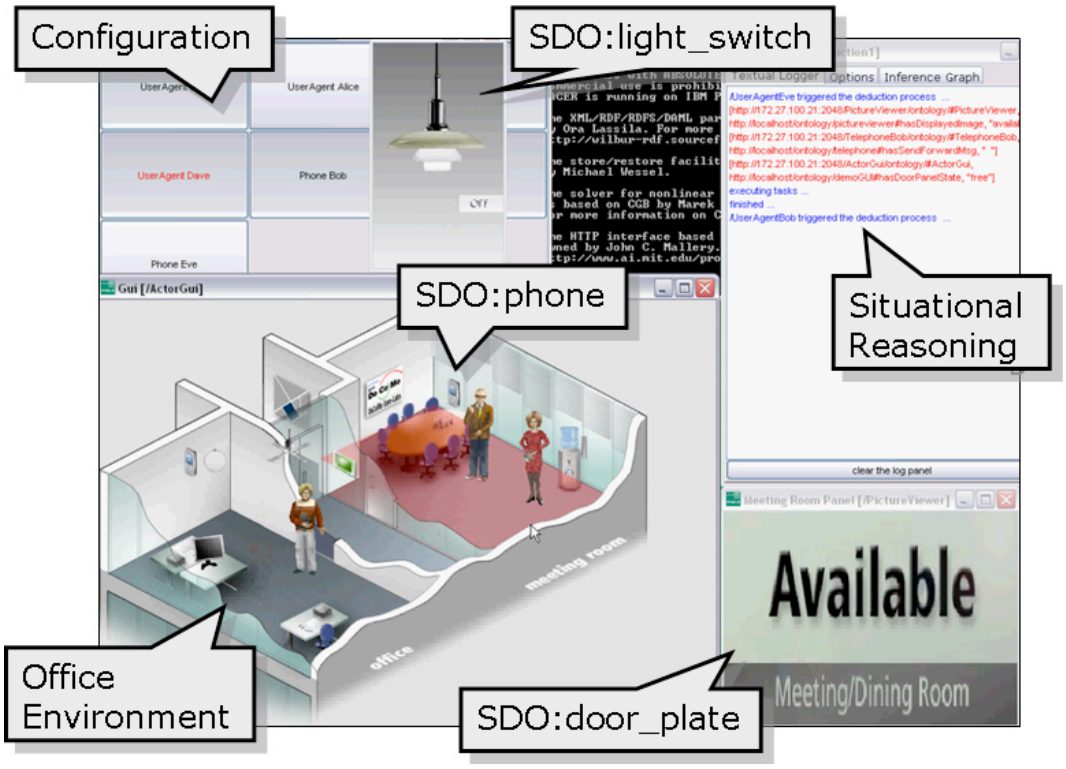

Figure 1: The OWL-SF Demonstration (annotated screenshot of running prototype).

Demonstration Setup: OWL-SF is implemented as a Java-based framework that features a simulation environment as well as connectors to SDO-encapsulated objects in the real world. Scenarios can be conducted as real time simulations that can connect to actual SDOs. The demonstrator features a GUI (cf. Figure 1) which visualizes people, mobile phones, rooms, door plates and other SDOs. Actual mobile phones that implement call forwarding based on OWL-SF reasoning support can be connected to the demonstrator via Bluetooth and/or Parlay APIs. The system connects to standard components for DL-based concept classification and reasoning. In addition, ambient devices can be influenced based on people's situation through the activation of non-chained configuration policies that are managed as XML rules.

Demonstration Path: We demonstrate an OWL-SF scenario that enables intelligent call forwarding in an office environment. The demonstration scenario is populated with several persons and mobile phones that hold different status and are placed in different locations. Persons and phones can be moved freely between locations. Depending on the identified context of key actors of the scenario (location of a person, the people in proximity, the current time, etc.) the situation of selected actors can be classified and further used to configure intelligent call forwarding.

Persons can either be in an office room, or in a meeting room, that is also used as dining room during lunch hour. Based on the location of a person, the people in proximity and the current time, the situation of that person is determined. The situation is then used to decide if that person is available for receiving phone calls, or if he is currently busy and phone calls should be redirected. A door panel in front of the meeting/dining room displays the current state of the room that is reasoned similar to the peoples' situations. Additionally, a jukebox in the meeting/dining room can be controlled by rules that can be modified by the user as any other rule in the system via a rule broker component. In the demonstrator, the time can be changed between the two abstract states office hour and lunch hour.