
Michael Schneider

Towards a General Object Memory

Received: date / Accepted: date

Abstract Most ubiquitous computing applications that involve smart objects need to deal with object-related information. Proprietary solutions to this problem increase the required implementation effort and hinder knowledge reuse between different applications. In this paper, we propose a general object memory model, which allows to physically and conceptually attach object-related information to smart objects.

Keywords Smart Objects · Knowledge Sharing · Semantic Web

1 Introduction

Most ubiquitous computing applications that are built on smart objects need to deal with object-related information in the one or other form. Such information includes but is not limited to static object models that describe general properties of an object, dynamic annotations added by the user or an application, and historic information about an object's former states or uses.

Unfortunately, most of today's applications use proprietary approaches to represent, organize, and store such information. This is inefficient, because it requires developers of smart object systems to solve similar problems and implement similar functionality over and over again. Secondly, proprietary approaches prevent different applications from sharing object-related information, which results in missed opportunities to generate inter-application benefits and an increased amount of work to construct individual object models for each application.

In order to ease the development of smart object applications and to support the exchange of object-related knowledge between different applications, we present our

ideas towards a general object memory model. The proposed model should be independent of any concrete technology that is used to implement a smart object and should cover an application range as broad as possible.

The rest of this article is organized as follows: First, we describe examples of smart object applications that we or other members of our group developed. We report about the experiences we collected, especially how object-related information is represented and managed. We show how our vision of a general-purpose object memory would change such applications. Next, we elaborate and discuss requirements of a general object memory model and present ideas, how such a model could be implemented. After a discussion on the representation of memory content we finish with a look on security and privacy issues.

2 Application Scenarios

Over the years, a multitude of applications that involve smart objects have been developed by members of our research group at the German Research Center for Artificial Intelligence (DFKI) and the artificial intelligence group at Saarland University.

The *Smart Shopping Assistant* (SSA, cf. [6]) application observes the users' interactions with RFID¹-tagged products via antennas placed in shelves and shopping trolleys. On displays mounted to the trolleys, context-dependant user support is provided by the display of detailed product information, product comparisons, or a list of recipes that could be prepared with the selected ingredients. For each product, object-related information including the product name, a textual description, images in different resolutions, the price, and a set of physical and technical properties is manually entered into an SQL database, and is related to the object through the globally unique identification number of the product's RFID transponder.

Michael Schneider
German Research Center for Artificial Intelligence
Tel.: +49-681-3025329
Fax: +49-681-3025020
E-mail: michael.schneider@dfki.de

¹ RFID: Radio Frequency Identification

The *Mobile ShopAssist* (MSA, cf. [9]) application runs on the user's PDA and allows him or her to interact multimodally with a set of smart products in order to compare them and query their features. Available input modalities are speech, handwriting, and selection-gestures (executed either by "pointing" on the PDA's display, or by picking up an RFID-tagged smart product from the shelf). Output modalities include text output on the PDA's screen and synthesized speech. In addition to general product information, the MSA requires an interaction grammar for each smart object, which defines the syntax of valid queries that can be posed on that object. The MSA keeps product-related information in individual, hand-written XML files. When the user is in front of a shelf, the files associated with the products in that shelf are downloaded to the PDA.

A third application from the domain of shopping is the *Digital Sommelier*, which supports a user in choosing the right wine. The Digital Sommelier is based on the SSA with two extensions: Product-related information is additionally read by a product-characteristic voice, and the wine bottles have been instrumented with wireless sensor boards (Smart-Its, cf. [1]), which permanently monitor the temperature, light exposure, and movement. This information is used to ensure the quality of the wine and to control the presentation of information in the store: Depending on the side of the bottle facing upwards (to the user), either detailed product information or the vendor's website is shown on an info display at the shelf. The Digital Sommelier retrieves all product information from a manually created SQL database, which also stores the information delivered by the smart products' sensors.

The *SPECTER* system (cf. [5]) is a personal assistant which keeps an artificial memory of the user's interactions with the environment. This memory can be used to reflect on past experiences, to construct a user model containing the user's general preferences, and to deliver ad-hoc assistance through a situation-aware trigger mechanism. A CD shopping scenario with one online (amazon.com) and two mock-up real world music stores with over 500 RFID-tagged CDs was implemented to test and evaluate the application. For each seen CD, an individual description, the price, and an optional user rating is stored in the memory. Support provided by *SPECTER* includes the display of product information and prices for any CD in the shop or in the memory, lists of owned and seen CDs, and the recommendation of CDs based on the user's ratings. In order to represent this huge amount of diverse information, an ontology-centered approach based on the Ontology Web Language² and the Suggested Upper Merged Ontology³ was chosen. Product-related information was automatically fetched from Amazon's E-Commerce Web service⁴ and converted into in-

dividual OWL instances. CDs are linked to the matching entries in the Amazon catalogue via Amazon's Standard Identification number (ASIN), which we extracted from visited web pages or stored on the CDs' RFID tags.

SharedLife takes the idea of *SPECTER* one step further by allowing users to share their recorded experiences with other users. As application domain we have chosen a grocery shopping and food preparation scenario. A corner stone in this scenario is the *SmartKitchen* (cf. [7]), an instrumented environment to automatically capture, share, and exploit semantically annotated cooking experiences. In record mode, the user's actions are observed via multiple video cameras, RFID-tagged products and kitchen utensils, and networked kitchen appliances. In playback mode, the recorded video is shown while the state of the environment is continuously compared with the recorded state. If for instance some ingredient is missing, the playback is paused until the problem is resolved. During both modes, the user is provided with context-dependant information about the currently handled object, like the temperature of a wine bottle (cf. the Digital Sommelier), or instructions on how to use a kitchen utensil. Like *SPECTER*, *SharedLife* uses an ontology-based approach to represent object-related knowledge. Knowledge about individual food items and kitchen tools is modelled by hand and placed on a web server in order to be sharable with other users. A mapping exists from physical objects' IDs to the according models' URIs. Information about the wine's sensors is imported via a proprietary bridge from the Digital Sommelier's database.

3 Vision: A General-Purpose Object Memory

Imagine that something like a general-purpose repository for object-related information exists for each object instance. Imagine further, that everybody and every application can add arbitrary information items to that repository and can retrieve items formerly added by others. Let us call this repository the *object's memory*. Assume that a unified interface exists to access an arbitrary object's memory, independent of the object's concrete implementation. How would the applications mentioned above have changed?

Together with each object, the manufacturer provides an initial version of the object's memory. This memory is populated with general information about the object: The object's name and class, images of the object in different resolutions, a description of the object's technical and functional details, a user manual, the URL of the companies support website, etc.. Such information is useful for all of the applications described above: Instead of entering the according information manually or searching for matching information on the web for each new object, the applications now can simply download the required data from the object's memory. Two of the applications presented above used speech output. Imagine,

² OWL, <http://www.w3.org/TR/owl-ref/>

³ SUMO, <http://www.ontologyportal.org/>

⁴ ECS, <http://aws.amazon.com>

that for each product an individual voice is added to the product’s memory by the merchant. All applications in the store, including the MSA and the Digital Sommelier, can use this voice such that a single product has a unique “look and sound” throughout the whole environment. Later, the user could override the default voice by placing an individual voice in the object’s memory. Imagine that all the sensors which monitor a wine bottle’s state in the Digital Sommelier application would not write their information in a database but in the according bottle’s object memory in some common format. Other applications like the SmartKitchen could access and exploit such information. Carried to the extreme, augmented memory systems like SPECTER or SharedLife could access and analyse an object’s complete history: Where and when was it bought at which price? How was it used? Was there any defect or service applied? Through the object memory as common exchange point for object-related information, applications can seamlessly share knowledge about smart objects.

4 Memory Organization and Access

In this section we propose our ideas toward a general object memory model. The model should be applicable to the broadest range of applications possible. Therefore, we pose the following four requirements on a general object memory model:

1. The memory model should support the storage of any kind of information item. The maximum size of the memory should not be limited in advance.
2. The memory model should be applicable independently of the technology that is used to implement a smart object. It should be applicable to smart objects with (active, i.e. Smart-Its) and without (passive, i.e. RFID) on-board processing capabilities.
3. The memory model should support memory access with (online) and without (offline) the presence of a network connection to some infrastructure.
4. The memory interface should be as simple and abstract as possible, while memory content should be represented in a generally understandable way.

In the following subsections we will discuss three questions which are related to the requirements above: Where is memory content stored? How is memory content accessed? How is memory content represented?

4.1 On-Board vs. Off-Board Storage

In principle there exist two possible ways to store smart-object-related information: Either physically on the smart object (which we refer to as *on-board storage*), or somewhere in the environment by using the smart object’s identity as an access key (which we refer to as *off-board*

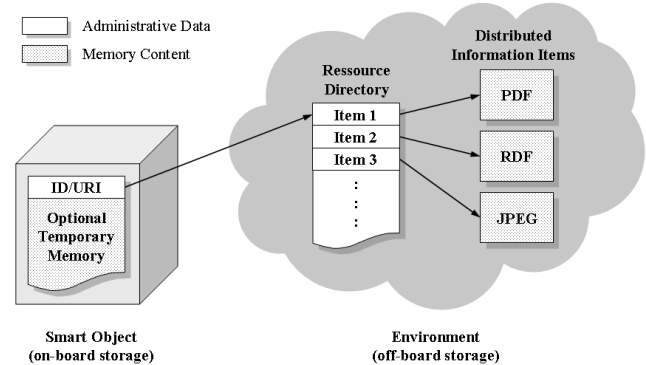


Fig. 1 Components of the general object memory model.

storage). Both approaches have their specific strengths and drawbacks. Information physically stored on the object is always available wherever this object is, and does not require the presence of a network connection to some external infrastructure. However, depending on the concrete hardware that is used to implement the smart object, the available storage space as well as the communication bandwidth may be extremely limited. Storing the information on some external infrastructure in most cases solves the issues of storage space and communication bandwidth limitations, but requires the presence of a network connection to that infrastructure whenever object-related information is stored, accessed, or updated. For a general object memory model, neither the limitation in storage space nor the requirement of a permanent network connection is acceptable.

Thus, we propose a hybrid approach: Object-related information can be stored either physically on the object (on-board), or conceptually attached to the object on some infrastructure in the environment (off-board). The on-board memory contains at least the object’s read only, unique id in form of a globally resolvable URI. This URI points to the object’s off-board repository. Further on-board storage space might or might not be provided by an object, depending on its technical implementation. The off-board storage is constituted by a central resource directory which refers to a set of distributed information items. While on-board storage might be used for small footprint temporary data, off-board storage is used to persistently store potentially large amounts of data. Synchronization between on-board and off-board storage is possible whenever a network connection to the off-board repository exists. Figure 1 gives an overview of the proposed object memory model and the relation between on-board and off-board storage.

The concrete amount of available on-board storage space heavily depends on the employed hardware and is in general not known in advance. Under this condition it is hard or even impossible to formulate a general and reliable mechanism to share this storage space fairly between an unknown number of unknown applications.

Additional information about the application domain is required to find a practical solution for this problem. It seems reasonable, that the environment has a sufficient understanding of the contained applications and their information needs in order to decide, how the on-board storage should be used in an optimal way⁵. For this reason, we propose to give the environment full access to the on-board memory and allow it to employ its own, domain-dependant mechanisms to assign on-board storage space to individual applications without the need to adhere to any concrete data organization or representation schemas. As a consequence, information consistency regarding the on-board storage is not ensured between different environments. As soon as an object leaves its current local environment, the semantic of the information stored on-board is potentially lost. We argue that this drawback is not as severe as it might seem: Often, only very few paths exist over which objects can enter or leave an environment. We assume that along those paths components can be installed, which initialize the object's on-board memory on entrance to the environment and back up the object's on-board memory on exit. Actually, with a network connection available at those few points, the on-board storage of an object can be smoothly synchronized with its off-board storage. Figure 2 shows a typical offline use case, where two offline accesses to the on-board storage are surrounded by an initial preparation and a final backup step. Additional synchronization steps may take place whenever an object is at a location where network access is available.

As an example, imagine a smart cargo box on a container ship. Although there is most probably no online connection during the travel on the open sea, the cargo box is always loaded/unloaded at a harbour with a crane, which is a good opportunity to synchronize on-board and off-board memory of the smart cargo box. As the whole process from loading to unloading is under the control of the shipping company, integrity of on-board memory items can be enforced.

In contrast to on-board storage, off-board storage is provided through an external infrastructure. Here, the two general choices are a centralized repository versus a decentralized or peer-to-peer repository. A centralized repository is reliable in the sense that the complete memory content can be accessed in one place, but it scales very bad when more and more information is added. A completely decentralized approach on the one hand implies the danger, that some information items might not be found or even lost, but on the other hand allows to manage an nearly unlimited amount of information items. Again, we propose a hybrid approach: Everybody who wants to add an information item to the off-board storage first has to make this item available under some globally accessible URI. He than adds this URI to the object's centralized resource directory. At any time, the

⁵ Note that this assumption might not necessarily hold and thus should be subject of discussion

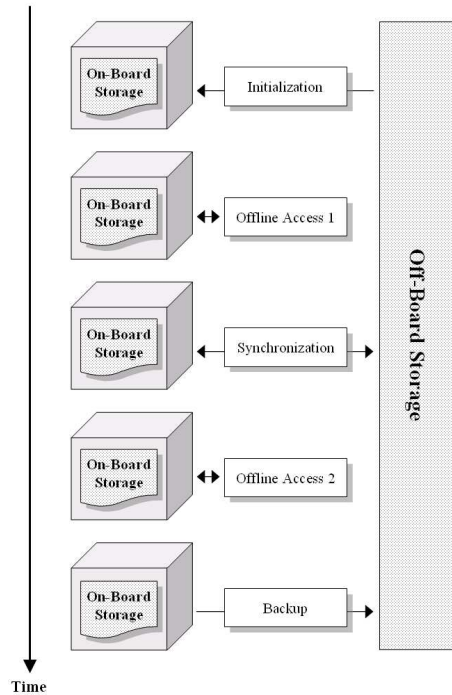


Fig. 2 Synchronization of on-board and off-board storage.

location of this directory can be resolved via the object's identity as described in section 4.2 (cf. figure 1).

Besides applications in the environment, active components that are part of the smart object itself might want to access the object's memory, i.e. in order to store sensed data. From the memory model's point of view, there is no difference between applications running in the environment or locally on the smart object. While the on-board storage can be accessed at all times, the off-board storage can only be accessed if there is some network connection available.

An open question is who is responsible for setting up and running an object's centralized resource directory in a real-world use case. A naive idea is that an object's resource directory is run by the object's manufacturer, as he would also be the first who is adding content to the memory. We will see later, that this might cause some problems. Other ideas involve non-profit organizations, commercial enterprises, or other special interest groups. However, in a research context, this problem in practise does not exist, as running a resource directory for relatively few objects does not consume an overly huge amount of resources.

4.2 Smart Object Interface

In order to support object memory functionality, smart objects need to meet two requirements: 1) There must exist some mechanism which allows an application/the environment to access the object's on-board memory.

Read access is mandatory; write access must be limited to the non-ID part of the on-board memory. 2) The on-board storage must contain at least the object's unique ID in form of a globally resolvable URI that references the object's resource directory. The ID is a null-terminated string that starts at memory offset 0. If further on-board storage is provided, it follows directly after the ID part and should be writable.

4.3 Memory Content Representation

A general object memory should be able to host all different kinds of information items. This ranges from binary data like images or sound files over human-oriented information like PDF files or HTML pages to semantically represented data like RDF models. To support the broad and automated exchange of object-related information, a semantic representation of data should be chosen wherever possible. In principle, each semantic representation language can be used for this purpose. However, for reasons of interoperability we propose to use a common standard language like RDF (Resource Description Framework), as it is widely used in the semantic-web and ubiquitous computing community. For all memory items, metadata about the item should be provided. This allows automated retrieval mechanism to identify and categorize relevant memory items. Such metadata should contain at least the type and purpose of the regarding item as well as the source of the item. An example would be the statement, that memory item `xy123` was added by `Sample Inc.` (the manufacturer), is a Word Document, and contains the object's English user manual.

Metadata is stored together with the URI of the actual information item in the object's resource directory. Such, an information retrieval application can decide if relevant information is associated with an object just by looking at the resource directory. Obviously, all metadata needs to be represented in a form a computer system can understand. We propose to use the "Gleaning Resource Descriptions from Dialects of Languages" language (GRDDL, cf. [3] and section 6) for this purpose. This format allows to hold a resource description in a human-readable (X)HTML format, and at the same time to transform it into an RDF representation. Thus, the object's resource directory can be directly accessed by humans while at the same time meta data can be exploited by automated reasoning processes.

5 Security and Privacy

Whenever information from various sources is collected and publicly shared, issues of information security, privacy, and trust arise. In this section we want to pose questions that are relevant to these topics in the context of object memories. However, as our field of expertise is

not IT security, we can not and will not give concrete answers to these questions today. Instead, we want to support a discussion on these topics by pointing out potential security and privacy issues. Some are common to each distributed system while others arise especially in the context of object memories. We hope that other researchers join this discussion and propose their view on and solutions to this important problem. Non-technical issues we identified include:

- *Correctness*: "Can I be sure about the content of an information item?" Information stored in an object memory might be faulty for many reasons, including intended and unintended ones. The issue of correctness is closely related to the aspect of trust.
- *Authenticity*: "Can I be sure about the source of an information item?" Under the aspect of trust, knowledge about the source of an information item is essential. It makes a difference if a positive review in a product's memory was added by a consumer protection organisation or by the manufacturer.
- *Completeness*: "Can I be sure that I see all memory content?" This might not be the case, for instance because the maintainer of the resource directory applies censorship for some reason. A manufacturer hosting his product's resource directories might choose to remove customers' critical reviews.
- *Integrity*: "Can I be sure that everybody sees the same memory content?" To talk and reason about an object in a multi-user scenario requires all participants to have consistent background information. This might not be given, as the provider of an information item could provide different versions of an item under the same URI.
- *Persistence I*: "Can I be sure that memory content will still be available tomorrow?" As in every distributed system, information sources may temporarily or permanently become unavailable. Applications may run into problems if important information vanishes.
- *Persistence II*: "Can I be sure that memory content will look the same tomorrow?" The content of a referenced URI can potentially change at some point in time. This might cause problems if different instances of a memory item are used in dependant applications.
- *Privacy*: "Can I be sure that only authorized users have access to sensitive information?" Eventually, not everybody should be able to access each and every information item. An object's service manual for example might only be readable by service staff. To impose such restrictions in an open and partly distributed system is not trivial.
- *Overload*: "Can I be sure, that the memory contains no spam?" One goal of the object memory is that anybody and any application can add object-related information freely. This entails the danger, that the object memory is flood with useless or unwanted information.

6 Related Work

The “*Gleaning Resource Descriptions from Dialects of Languages*” language (GRDDL, cf. [3]) was introduced to solve the problem of namespace documentation in XML. Although namespaces are defined by URIs, it is not clear what this URI resolves to. The namespace specification itself does not require any resource to be available at this URI. In [2] it was proposed that a namespace documentation should be provided under a namespace URI which – besides a textual documentation of the namespace – lists all namespace-related resources like schema descriptions, style sheets, renderer, etc. in a formal way. In an effort to create a namespace description language to combine the human-readability of HTML documents with machine-readable semantics, the Resource Directory Description Language (RDDL) was introduced. For each associated resource, the RDDL specified the nature (describing the chosen representation) and the purpose of the resource, as well as a link, where the resource can be downloaded. RDDL used the unpopular XLink standard and was difficult to convert to RDF, thus the Gleaning Resource Descriptions from Dialects of Languages language was developed. The core idea of GRDDL is to attach a custom XSLT style sheet to a human-readable XHTML page, which generates RDF statements from the GRDDL page’s content.

The idea of using GRDDL in the context of object memories is, that smart objects, if identified by URIs, constitute a “real-world namespace” on their own. As GRDDL is applicable to describe namespaces, it should be also applicable to describe smart objects and their related resources.

Related to the idea of object memories is the augmented world model described in [4]. The authors envision the underlying framework “as being a middleware that brings providers and consumers of location-based information together”. The described framework focuses on information that is bound to certain locations. Consequently, it heavily relies on services provided in the environment, there seem to exist no “on-board” components. In contrast to this, the memory model proposed in this paper is object-centered and also covers application scenarios, where no memory services are offered by the infrastructure. A combination of both approaches might be reasonable.

7 Conclusion and Outlook

With this paper, we want to initiate a discussion on the important topic of object-related knowledge sharing and reuse between heterogeneous applications, possibly spanning multiple environments. As “food for further thoughts”, we presented our ideas towards a general memory model for smart objects. We argued, that such a memory model should distinguish between information

stored on-board and off-board, in order to support the collection of huge amounts of information as well as to support the use case where no network connection to an off-board infrastructure is available. We proposed a hybrid approach to solve this issue, and further proposed to introduce synchronization points within the environment to update on-board and off-board memory. We proposed a general interface to object memories and discussed how memory content might be represented. We finished with an enumeration of security and privacy issues that evolve from using general object memories.

Upcoming work will have to approach the questions posed above. It will also include a prototype implementation of the proposed memory model to test the presented ideas in a realistic application scenario. Further questions that need to be addressed are the modelling of low-level memory content and the directed retrieval of relevant memory content.

Acknowledgements The research presented in this article was conducted within the project SharedLife. SharedLife is being sponsored by the German Federal Ministry for Education and Research (BMBF) under contract 01IWF03 from January 2006 through December 2008.

References

1. M. Beigl, H. Gellersen. Smart-Its: An Embedded Platform for Smart Objects. In Proceedings of the Smart Objects Conference (sOc) 2003, Grenoble, France (2003).
2. J. Borden, T. Bray. Resource Directory Description Language (RDDL), <http://www.rddl.org/> (February 2002).
3. D. Hazael-Massieux, D. Connolly. Gleaning Resource Descriptions from Dialects of Languages (GRDDL), W3C Coordination Group Note NOTE-grddl-20040414 (April 2004).
4. D. Nicklas, M. Großmann, T. Schwarz, S. Volz, B. Mitschang. A Model-Based, Open Architecture for Mobile, Spatially Aware Applications. In Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD), Redondo Beach, CA, USA (2001).
5. C. Plate, N. Basselin, A. Kröner, M. Schneider, S. Baldes, V. Dimitrova, A. Jameson. Recommendation: New Functions for Augmented Memories. In Adaptive hypermedia and adaptive web-based systems: Proceedings of AH 2006. Springer, Berlin, Germany (2006).
6. M. Schneider. Towards a Transparent Proactive User Interface for a Shopping Assistant. In Proceedings of the Workshop on Multi-User and Ubiquitous User Interfaces (MU3I) at IUI, Madeira, Portugal (2004).
7. M. Schneider. The Semantic Cookbook: Sharing Cooking Experiences in the Smart Kitchen. To appear in Proceedings of 3rd International Conference on Intelligent Environments (IE’07), Ulm, Germany (2007).
8. W. Wahlster, A. Kröner, D. Heckmann. SharedLife: Towards Selective Sharing of Augmented Personal Memories. In O. Stock, M. Schaerf (Eds.), Chapter in: Reasoning, Action and Interaction in AI Theories and Systems. Springer, Heidelberg, Germany (2006).
9. R. Wasinger, W. Wahlster. The Anthropomorphized Product Shelf: Symmetric Multimodal Interaction with Instrumented Environments. In E. Aarts, J. L. Encarnacao (Eds.), Chapter in: True Visions: The Emergence of Ambient Intelligence. Springer, Heidelberg, Germany (2006).