

# AUGMENTING INTERACTION IN INTELLIGENT ENVIRONMENTS THROUGH OPEN PERSONAL MEMORIES

Michael Schneider, Alexander Kröner, Rainer Wasinger

German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany

## ABSTRACT

One of the human strengths is to utilize past experiences to solve problems. This process can be technically supported by collecting and exploiting the history of a user's interactions with an intelligent environment. However, most of today's applications cannot or do not take on the additional burden of providing the user with sophisticated memory functionality. In this paper we present the concept of an open personal memory, which allows arbitrary applications to contribute knowledge to the memory with minimal extra effort. We describe a proof-of-concept implementation and show how users can exploit such memories.

## INTRODUCTION

Humans constantly utilize past experiences when faced with novel problems. This process can be technically supported by collecting and exploiting the history of the user's interactions with an environment and the applications present in such an environment. With an ever-growing number and increasing diversity of employed sensors, intelligent environments today provide the best conditions to observe such interactions. Accordingly, several research projects are currently investigating ways to build and exploit digital memories in intelligent environments.

A very simple form of exploitation is to provide the user with an interface to recall past experiences in order to reflect and thus better understand a certain situation or the way he or she used a certain tool. In contrast to the human memory a digital memory may contain a more complete coverage of a certain event and could thus expose coherences that are originally overlooked by the user. A more complex use of digital memories is described by the paradigm of *recomindation* [sic!] (6), which blends the concepts of recommendation and reminding. Recomindation is based on the assumption that explicit reminders of the user's past experiences and behavior can enhance a recommendation process. These reminders establish a naturally rich and personal context that helps the user to put the recommendations into perspective and to evaluate their applicability in the current situation. Further applications of digital memories are discussed together with related projects in section 2.

Although past and recent research results show that historic information can enhance the interaction with an intelligent environment or even with a single application in a multitude of ways, a widespread integration of such functionality in applications and environments seems far off. On the one hand, a native integration of such functionality is a burden many applications cannot or want not take. On the other hand, existing personal memory architectures are too restricted to allow arbitrary applications to participate. This is either because such architectures focus only on a closed and limited domain, or because they are mostly self-contained and ignorant about additional external sources of new memory content.

In order to ease and thus push the widespread application of memory-based functionality we propose the concept of open personal memories. Open personal memories allow arbitrary applications to contribute to a universal shared interaction memory with minimal extra effort.

In the following sections we will give a definition of an open personal memory and discuss its general properties. We will demonstrate the feasibility of the proposed approach by describing how we extended an existing application in a shopping scenario to contribute to an open personal memory, and how a user can exploit the collected information to discover novel interaction opportunities.

## RELATED WORK

The idea of creating artificial memory structures for user support is addressed by numerous and diverse research efforts. For instance, the early work (9) demonstrates how a user's recall can be supported by the memory of simple interactions automatically captured in an intelligent environment. Here the user communicates exclusively with the memory by means of a mobile device specifically designed for this setting.

Today, progress in sensor technology and widespread distribution of mobile devices with sensory capabilities allow for the realization of considerably more complex settings. In (8), the authors describe how records of nursing activities automatically captured in an intelligent environment can be exploited to avoid medical accidents in hospitals. An example of how

memories can support social matching is given by (13), where so-called experience logs of participants in an academic conference are applied to encourage later conversation between participants with common interests. In (16), a scrapbook that is both digital and physical in form is described to help the elderly in recalling, sharing, and reviewing their memories of life experiences based on photos. Through a series of usability studies, the scrapbook photo-album is tested for its ability to support memory sharing. An everyday life setting is addressed in (12), where the authors describe a software and hardware platform that exploits records of people's activities in order to provide access to community-related information.

These works combine personal memories with the idea of ambient intelligence. Other examples that capture user activities in memory-like structures are an approach that combines a document repository with automatically captured context data for retrieval (4), an approach that puts the user's desktop communication into the focus (10), and an approach to exploit the context of desktop activities (11).

The artificial memories researched in these works are "closed" in some sense – either specific devices are required or a scenario with a closed domain (and therefore a limited set of interactions) is addressed. Following the idea of ubiquitous user modeling (5), we want to explore how building and exploitation of personal memories can be achieved free from such constraints, with the long-term goal being to set up personal memories for any kind of intelligent environment.

## OPEN PERSONAL MEMORIES

An *open personal memory* is a collection of past events that is "open" in that arbitrary sources can contribute to the collection and "personal" in that the memory is owned by a single user who has full control over the stored information and the way this information is used. Such a memory could be implemented on a mobile device that is under the user's direct control like a PDA or a mobile phone, or – if network access is available in the environment – could be realized as a remote service hosted on a trusted server. A concrete implementation of an open personal memory will be presented in the next section.

Every time a user interacts with the environment (or more specifically with an application hosted by the environment), the application may contribute a high-level description of this interaction to the according user's open personal memory. In a real-world shopping scenario such an event posted by an intelligent shelf could for instance state that "the user has taken product

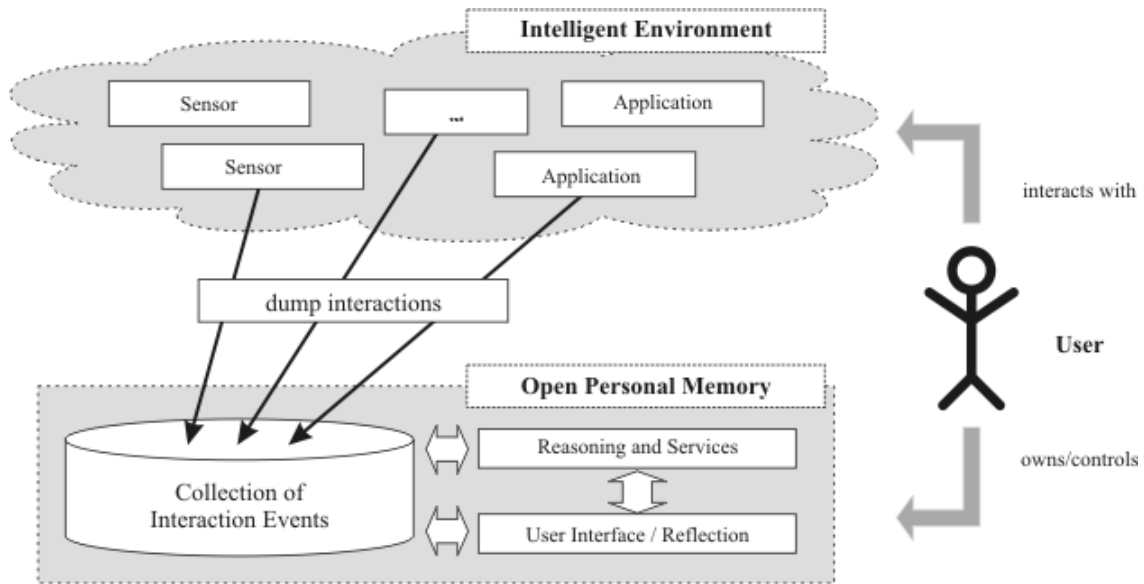
P out of the shelf" or that "the user requested the price of product P". We will see more complex examples of such events later when we discuss our proof-of-concept implementation.

Besides providing the infrastructure for the collection of interaction events, an open personal memory implementation also has to provide services that enable the user to exploit the memory content. In its simplest form this might just be a user interface that allows one to browse and search the memory for past events. More complex services may reason about the recorded interactions and may for instance discover and present regular interaction patterns or propose novel ways to interact with the environment. To control such services as well as the general operation, each open personal memory needs a user interface, which may be realized directly on the mobile device, or as a remote interface like a web portal or proprietary software client.

Privacy is always a concern in intelligent environments, and one might assume that by introducing open personal memories additional privacy challenges are posed. Fortunately this is not the case. First of all, open personal memories are completely passive (with one exception that will be discussed later) in that they consume information that is available in the environment anyway. Furthermore, this information is stored and processed on a device or service that is under the user's control. Assuming the device or service is carefully designed and implemented, none of the stored information is exposed without the user's permission. This of course does not free the intelligent environment from its role in securing user privacy, as an environment may still collect and process information about a user on its own, or might expose private information about a user's interactions to unauthorized third parties. However, these issues are independent of the application of open personal memories, and apply to all instrumented environments.

The overall setup is depicted in figure 1. A user entering an intelligent environment introduces his open personal memory to the environment. As he interacts with the environment, applications and sensors post their knowledge about these interactions directly to the user's memory. The user can at any time exploit the captured history of his interactions by invoking certain services provided by his open personal memory.

For reasons of simplicity, this paper will not cover technical aspects like the discovery of an open personal memory by interested applications or details of the communication between the involved components. Technologies that can be used for these purposes include for example WiFi, Bluetooth, UPnP, Jini, etc. In the following we assume that there exists some infrastructure in the environment that handles these technical aspects.



**Figure 1. General use-case of an open personal memory in an instrumented environment.**

## PROOF-OF-CONCEPT IMPLEMENTATION

In this section we will describe how we tested the concepts presented in this paper by extending the Mobile ShopAssist (14), an existing shopping assistant application, to contribute to an implementation of an open personal memory called SPECTER-Light. First we describe the Mobile ShopAssist application and see an example of typical interactions in a real world scenario. These interactions are posted to the user’s memory in parallel. We then present our open personal memory implementation SPECTER-Light and see how the interactions can be reviewed and exploited. At the end of this section we discuss technical details like the contribution API of our memory and the employed knowledge representation.

### Mobile ShopAssist

The *Mobile ShopAssist* (MSA) application was developed as a platform for demonstrating a wide range of different mobile and multimodal interaction possibilities. The primary context of the MSA is that of shopping, during which a user is able to interact multimodally with a set of items like “digital camera” products to compare them and query their features. The MSA supports the user throughout the buying process, including finding a product, querying and comparing products, and finally purchasing the product.

The MSA is designed specifically for mobile PDA devices and all of the processing (e.g. speech and handwriting recognition) is performed locally on the device itself. The MSA is multilingual (English, German), and interaction with the system is derived from the base modalities speech, handwriting, and selection-gesture, whereby selection-gesture refers to the selection of referents on the mobile device’s display

via a “pointing” action, and the selection of tangible objects in the real world via “pick-up” and “put-down” actions that are observed via RFID antennas attached to the shelves.

Interactions consist of a command and one or more object references, which can be provided to the system via speech, handwriting, and/or gesture. Typical commands include for instance product comparisons and queries over product features (e.g. price), while object references refer to shopping items (e.g. PowerShot S50).

As described in detail in (14) among the types of multimodal interaction that the MSA demonstrator supports are those that are temporally overlapped (i.e. different modalities overlapped in time) and those that are semantically overlapped (i.e. different modalities overlapped with respect to semantic content). The mobile and multimodal demonstrator also supports “anthropomorphized objects” and “direct” and “indirect” product interaction (15).

We will now give an example of a typical interaction between the user and the MSA. A dump of this interaction is posted in parallel to the user’s open personal memory, as we will see in the next section. Consider the following situation:

Mr. A from Saarbrücken plans to go shopping for a digital camera. Attracted by all of the sales he can see through some shop’s display window, Mr. A enters the shop and navigates his way to a real-world shelf of products containing digital cameras. He loads the Mobile ShopAssist application, which connects to the shelf of cameras (each shelf’s unique ID is provided by an infra-red ID beacon<sup>1</sup>) and synchronizes the shelf’s contents with the PDA. This downloads all relevant

<sup>1</sup> Eyed infra-red ID beacon, <http://www.eyeled.de>

product information including pictures for each of the products in the shelf. After synchronizing with the shelf, Mr. A sees a list of available cameras on his PDA. Now Mr. A begins to interact with the products.

An intuitive one-to-one mapping exists between physical shopping items and their digitally-represented counterparts. As a result, Mr. A is able to interact with the *digital* set of products visible on the PDA's display, but also with the *physical* set of products available on the shelf, and with a combination of *physical and digital* products making up the data space. Figure 2 demonstrates the latter case, in which Mr. A compares two products using the modalities speech and gesture: "Compare this camera <point on display> to this one <take box from shelf>". In reply to the request, the system provides the visual output shown in figure 4 where two cameras are displayed side-by-side and with their most relevant attributes also listed. This visual output is further accompanied by a spoken summary of the cameras, the speed of which can be increased, decreased, or stopped.

Speech being Mr. A's favorite modality, he interacts with the system using speech as the communication mode: "What is the price of the PowerShot S50?" The system replies: "PowerShot S50. Price. 120 Euro." Under a different environment context, Mr. A may have preferred to use a different modality or modality combination for interacting with the system, for example handwriting combined with intra-gesture, which would be more appropriate in contexts where a high level of background noise is present, or where user privacy is desired.

### Open Personal Memory Implementation

SPECTER-Light – a prototype implementation of an open personal memory – is based on the SPECTER

system (7). SPECTER was developed as a personal ubiquitous assistant, which records a history of the user's experiences in order to deliver ad-hoc and subsequent context dependant support. SPECTER was originally developed to perform deep analysis of semantically rich experiences, and thus exhibits a lot of functionality that is not needed to demonstrate the concept of an open personal memory. As a result, this paper only describes the core components of SPECTER. The interested reader can find a detailed description of the original SPECTER system in (7).

Figure 3 shows a schematic view of SPECTER's general architecture. Perceptions from the environment are initially stored in *short-term memory* as a stream of raw sensor data. After these perceptions have been filtered for duplicates, an abstraction process is applied to create a symbolic representation of the observed information. These abstractions are matched against a list of service triggers, which allow for a context-dependant invocation of services. The symbolic representation is then forwarded to the *long-term memory*. Here, all information is stored in a so-called context log. Additionally, related observations are combined into entries of a so-called personal journal. While the context log contains a fine-grained, data-centric representation of recorded experiences, the personal journal provides a condensed, user-centric view of the memory. In the personal journal, the user may optionally annotate certain entries with personal ratings and thus express his likes and dislikes.

By relating these ratings to their context, SPECTER can over time learn a user model. Machine learning is applied in order to extract features from the Context Log, which are statistically related to the user's preferences. In an interactive process during reflection (see section "Augmenting Interaction") these features may be refined and adjusted by the user based on semantically relationships defined in SPECTER's



Figure 2. MSA interaction illustrating the use of speech and gesture during a product comparison query.

ontology (2). Machine learning also is applied by a second component, which discovers spatial relationships by constructing and analyzing motion profiles from GPS data stored in the memory (3).

Last but not least, SPECTER provides the user with a user interface that allows him or her to introspect the content of the long-term memory and to invoke services or to define service triggers. In fact, two different kinds of user interfaces exist for SPECTER: A mobile user interface implemented for PDAs allows the user to perform basic interactions on-the-go, while a more comfortable user interface allows the user to review past experiences on his or her PC at home. Examples of these user interfaces are given in figures 5 and 9.

SPECTER is implemented as a remote service running on the user's PC at home, which is expected to have a permanent network connection. This way, the SPECTER implementation can be used independently of any concrete mobile device or platform. A remote interface provided via an HTTP connection allows the user to connect to SPECTER from many different kinds of devices. Additionally, this way the PC's superior processing and memory capacities can be utilized to store and process a potentially huge collection of recorded interaction events.

The minimal set of SPECTER components that is required to realize at least basic open personal memory functionality consists of the personal journal, the reasoning processes used for the introspection, the user interfaces to access and control the system, and various low level components which are responsible for the storage and management of memory entries.

In our SPECTER-Light system we additionally included the component that allows a user to rate entries of the personal memory in order to express his preferences. This information may be used in the introspection interface, as we will see later.

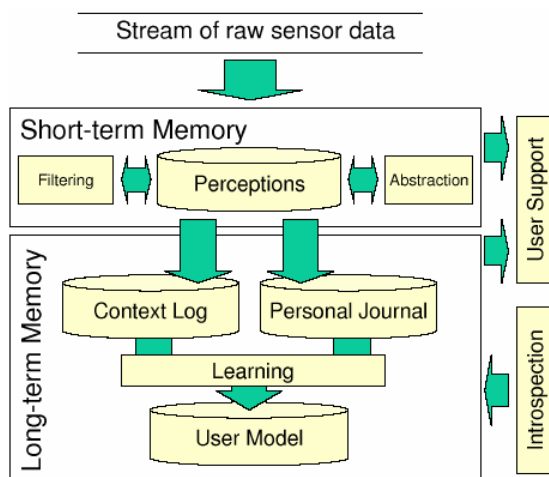


Figure 3. Architecture of the full SPECTER system.

We will now go back to our evaluation scenario. Let us recall Mr. A's interactions with the MSA. They have been recorded in SPECTER-Light's memory and can now be recalled via the reflection interface. A detailed description of the reflection process will be given when we discuss the exploitation of the memory content. For the moment, let us just look at what information about Mr. A's interactions has been recorded in his open personal memory.

Figure 5 shows some recent interactions displayed by SPECTER-Light as a simple list, and annotated with the source of the event (here an electronics store), the timestamp when the interaction was observed, and the user's rating of the according event. The information shown in the list originates from the interaction that Mr. A performed earlier on in the store while using the MSA (see previous section). In particular the list shows the kind of interaction (in this example "Price Request" and "Comparison Request") and the involved objects. Mr. A can "zoom" into any event by clicking on the according interaction description. In our example he clicks on the "Comparison Request" event and gets the details screen shown in figure 6.

The images of products and the icons beside the interaction descriptions are provided by the MSA as part of the interaction description. We will now see how such an interaction description is constructed and exchanged.

### Knowledge Representation and Sharing

Applications that want to contribute interaction events to an open personal memory have to represent these events in a way that is understandable and interpretable by both the human user of the memory as well as the automated reasoning processes of the memory implementation. A common solution in such cases is the use of an ontology, which defines a shared vocabulary for the exchange of semantically rich information. The ontology used in our open personal memory implementation is expressed in the Web Ontology Language (OWL<sup>2</sup>) and is based on the Suggested Upper Merged Ontology (SUMO<sup>3</sup>) and Mid-Level Ontology (MILO<sup>4</sup>). Interactions are represented as processes. Figure 7 shows an excerpt of the SUMO process hierarchy that contains the Comparing process, which the MSA uses to describe the camera comparison shown in figures 2 and 6.

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

<sup>3</sup> SUMO & MILO, <http://ontology.teknowledge.com/>

For an open personal memory implementation, the following properties from the process class are required:

- `agent`: The agent (or person) that performed the shared interaction. In most cases this will be the user itself, but could also be a proactive application that initiates an interaction with the user.
- `patient`: The items that are involved in the described process.
- `whenFn`: The time interval during which the process was performed.

Three other extremely useful and thus recommended properties are `located`, `subProcess`, and `realization`. The property `located` specifies the location where the described interaction took place. As the exact location may be hard to determine in some scenarios, this property is not strictly required. The `subProcess` property allows applications to give a more fine-grained description of the observed interaction, as is shown by the camera comparison example in figure 6. This property is optional, thus each application can decide on its own, how detailed it wants to describe an interaction.

The `realization` property is of special interest in combination with “replay” functionality, which will be covered in the section on “augmented interaction”. For now we just need to know that this property points to an instance of a `Process` class, which contains a description on how to replay/repeat the performed interaction. If the interaction is purely virtual, i.e. an invocation of a software function, this description can simply refer to some remote service which invokes the according function. If the interaction requires physical interaction of the user with some part of the environment, the description should be given in a human readable format, e.g. in a text document describing how the interaction can be repeated. In both cases the description must contain the required parameters and/or involved objects and their expected classes. As the replay description may become quite complex – especially if physical interaction of the user is required – providing this description is optional. If the description is not provided replay functionality will not be available for the associated event.

In order to represent information about the source of the shared interaction, the process instance that describes the observed interaction is wrapped into a `Perception`. Here, the property `agent` refers to the information source (in our example the MSA), and the `patient` property refers to the actual interaction `Process`.

Figure 8 shows the ontological representation of the camera comparison interaction that Mr. A performed in the electronics store while using the MSA. The shaded boxes represent individuals of the denoted classes. The

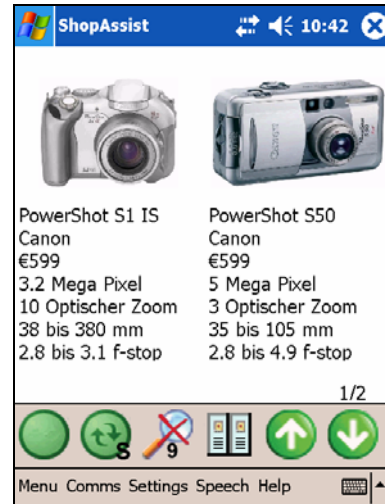


Figure 4. Result of a “comparison” interaction.

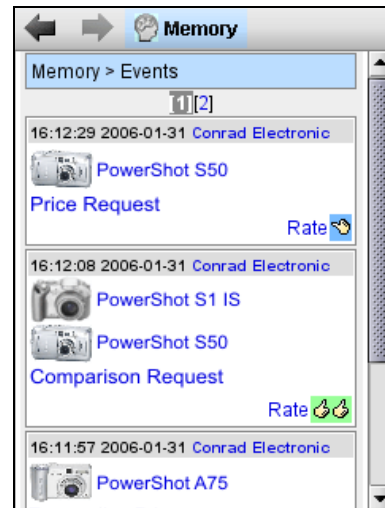


Figure 5. Interactions recorded in the memory.

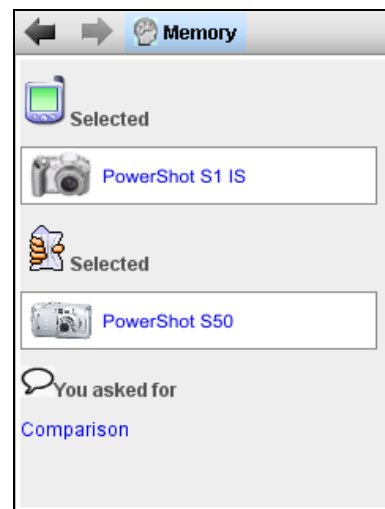


Figure 6. Details view of a comparison interaction.

arrows represent relationships between these individuals through the denoted properties. Due to space restrictions

and reasons of clarity we have omitted the representation of the Comparison process's sub processes as well as all properties used for the annotation of the individuals in a human readable format (textual description, pictograms, etc.), and the replay description. The latter in this case is quite simple and consists of a URL pointing to a web service interface of the camera comparison service. For parameters, the web service expects two individuals of the class camera. By calling this URL with appropriate parameters, a comparison of the specified cameras is displayed.

In order to reduce the effort for modeling the products in our demo setup, we simply reused product information that is available free-of-charge from Amazon's e-commerce web service<sup>4</sup>. The information provided by this service includes product names, short and long descriptions, pictures, prices, category information, technical properties, etc. and is available for all products that are listed on amazon.com. The model for each product can be fetched under a unique URL that contains the Amazon Standard Identification Number (ASIN). A script on our server decodes the requested URL and invokes Amazon's web service to retrieve the according product description. As the returned result is given in a proprietary XML format, the script finally generates a valid OWL model of the requested product information. Thus, a detailed model describing any product available at amazon.com can be generated on the fly.

External applications post their interaction events to the open personal memory via a very simple web service situated on an embedded web server. In order to contribute, applications post their interaction events in RDF/XML format to a certain URL provided by SPECTER-Light. For reasons of simplicity we assume that interested applications somehow know this URL. In fact, in our proof-of-concept implementation this URL was provided in the MSA's configuration file. In a

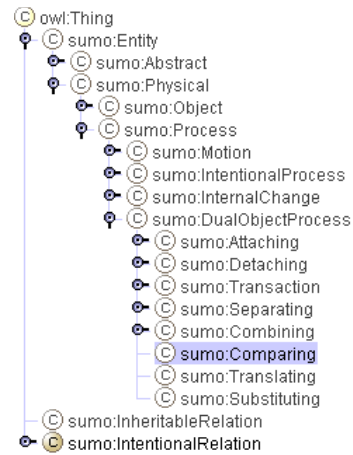


Figure 7. Interactions are described as processes.

realistic setup techniques like UPnP or Jini would have to be used in order to perform dynamic service discovery.

### AUGMENTED INTERACTION

In this section we describe how users can exploit the content of their open personal memories in order to review their past interactions and in order to augment their future interactions with the environment. We will first continue with a more detailed description of the reflection process, which we introduced in the last section. We will then have a look at more sophisticated services that exploit the rich semantic structure of the recorded interaction events.

### Reflection

Reflection is the process of manually reviewing past interactions recorded by the open personal memory. The

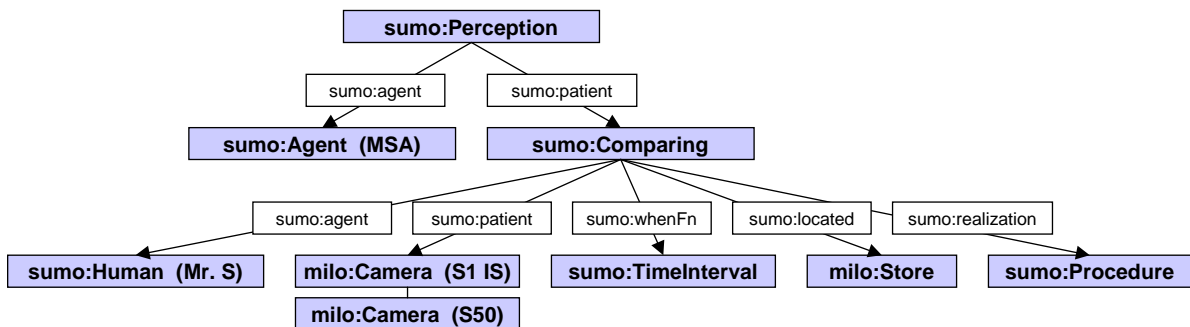


Figure 8. Exemplary representation of a camera comparison interaction. Shaded boxes represent individuals of the denoted classes, arrows with white boxes represent property relationships between individuals.

<sup>4</sup> Amazon E-Commerce Web Service,

user may perform reflection for different reasons: First, it may remind him or her about a concrete past event or interaction. Mr. A for instance – now home again – might wonder which cameras he had seen in a certain store. Remember that SPECTER-Light allows the user to assign ratings to interactions. Thus Mr. A could have marked his favorite camera in the store and would now be able to retrieve the name of this model at home. Another reason why reflection on past interactions is useful is to learn about the interaction possibilities and limitations inherent to certain environments. By reviewing the interactions in a limited part of the environment the user might draw connections between certain situations and interactions that before might have not been obvious to him or her in the heat of the moment.

SPECTER-Light allows the user to reflect on recorded interactions at any time, however the form of presentation and the available services depend on which concrete interface is used by the user.

The mobile interface offers features matched to the limited capabilities of mobile devices. Its web browser-like interface enables the user to explore SPECTER-Light's memory, either by consulting an ordered list of recorded interaction events (see figure 5) or by zooming in on a certain event (see figure 6). The ordered list representation displays the recorded interactions within their temporal and spatial context. The latter view shows the interaction's decomposition into sub processes (if provided by the source application) and allows the user to view details of the performed interaction.

The mobile user interface allows a user to provide feedback in various ways. As we have already seen, a user can assign ratings on experiences to express his likes or dislikes. Additionally, the user might create so-called reminder points. These temporal "bookmarks" indicate the need for reviewing some temporally near interaction. As an example, Mr. A could set a reminder point in the store after he performed an interesting comparison that he wants to review at home.

While such feedback can be submitted at any time, exploiting its full potential requires a more extensive reflection and introspection process, which is supported by SPECTER-Light's desktop interface. The corresponding user interface is shown in figure 9 and consists of a virtual character, a presentation screen, and a resized version the mobile interface.

This interface setup enables a mixed-initiative reflection process (1) of the previously recorded interactions. For the system, the reflection process provides an opportunity to acquire further feedback from the user, for instance, in order to adjust system services or to better guide the reflection process. For the user, this

extended reflection interface provides the ability to conveniently access and browse his or her open personal memory, review the previously set reminder points, or invoke additional services provided by the open personal memory application.

Another interesting feature that was mentioned previously is the open personal memory's ability to assist the user with a "replay" of past interactions. A *replay* is the process of repeating a previously performed interaction. Imagine Mr. A was disrupted during a camera comparison in the shop by an urgent phone call. Now that he is at home he has time to replay the recorded comparison without any hassle. Whenever a replay description (cf. previous section) is provided with an interaction event, an open personal memory can assist the user with the replay of the according interaction. If the interaction consists of an invocation of some application or electronic service, SPECTER-Light can perform this invocation on behalf of the user, for instance to bring up again a camera comparison. If the interaction requires physical interaction of the user with the environment or some item, SPECTER-Light can present the user with a description how to repeat the interaction manually.

The invocation of an external application or service performed by the open personal memory during a replay is the only situation where an open personal memory actively interacts with the environment, which is the exception we mentioned when discussing the privacy aspects of open personal memories. In deed, the active replay functionality can cause privacy concerns, as it enables the environment to reidentify a user over session bounds. However, the replay functionality is only triggered on explicit command of the user, so that he or she can at any time decide, if the expected gain from performing a replay outweighs the potential loss of privacy.



Figure 9. Reflecting on past experiences with SPECTER's desktop environment (compacted).

### Opportunity Discovery and Extended Replay

Besides the browsing of memory content as described above, an open personal memory may provide more sophisticated exploitation services that make use of the

rich semantic structure of the recorded interaction events.

One possibility for such exploitations is to search the open personal memory for interaction events that partially overlap with a given event (and thus are similar to this event). That way, given an interaction I with some item O at some place P, the user could for instance ask the following questions:

- What other interactions can be performed using item O?
- What other items can be used in interaction I?
- What other interactions are possible at place P?

All these questions can be answered by automatically analyzing the structure of the shared interaction events and by performing type inference based on the ontology's class hierarchy and recorded replay descriptions. By additionally taking into account the ratings that the user assigned to certain events, his or her preferences for certain interactions and/or objects can be reflected in the order of the returned propositions. Figure 10 shows how a list of items is retrieved from the memory that can be used in a camera comparison interaction. As one can see, additional location-sensitive filtering can be applied. Similar to the request for related objects also related services and other interaction possibilities at a certain location can be retrieved from the memory.

Another functionality that can be realized with open personal memories is so-called "extended replay". As we have already seen in the section on reflection, active replay can be used to repeat previously performed interactions. Now we combine this functionality with our search for items that can be used together with a certain interaction – in this case the interaction we want to replay. In this way, we can exchange a-posteriori the involved subjects of a replay with other matching items from our memory. The extended replay user interface is shown in figure 11. Imagine once again that Mr. A is sitting at home after his shopping trip. When reflecting about his trip he suddenly notices that he has completely forgotten to compare his two hot candidate cameras in the store, although he had looked at each of them separately. With the extended replay functionality, he quickly recalls both cameras from his open personal memory and "replays" a comparison of both cameras on his PDA.

## CONCLUSION AND OUTLOOK

In this paper we presented the concept of open personal memories, which provide a simple to use platform for arbitrary applications to contribute interaction events to

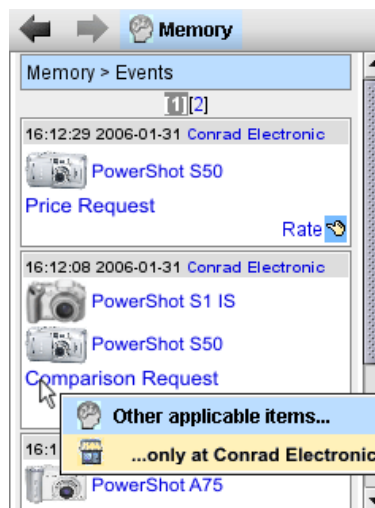


Figure 10. Searching the memory for related items.

a user's artificial memory. We described our proof-of-concept implementation, in which we extended an existing shopping assistant application to contribute to our open personal memory implementation based on the SPECTER system. Finally we proposed several novel functionalities that enable the user to exploit the recorded interaction history in order to enhance and augment his or her future interactions with an intelligent environment.

The practical utility of open personal memories depends on the additional benefit that users can draw from an exploitation of their past experiences. The prototype setup presented in this paper provides a test bed to implement and evaluate different exploitation approaches. In a first step, the basic exploitation

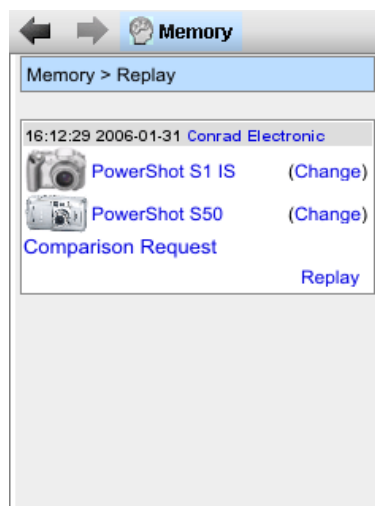


Figure 11. Performing extended replay.

techniques proposed in this paper need to be evaluated in a user study. Then, more complex exploitation approaches can be realized based on the existing services.

## ACKNOWLEDGMENTS

This work was funded by the German Federal Ministry for Education and Research (BMBF) under the contract numbers 524-40001-01 IW C03 (SPECTER) and 01 IN C02 (COLLATE II).

## REFERENCES

1. Baldes, S., Kröner, A., and Bauer, M., 2005. Configuration and Introspection of Situated User Support. In: Proceedings of LWA 2005, Lernen Wissensentdeckung Adaptivität, pp. 3–7.
2. Bauer, M., and Baldes, S., 2005. An Ontology-Based Interface for Machine Learning. In: Proceedings of the International Conference on Intelligent User Interfaces (IUI 2005), pp. 314–316.
3. Bauer, M., and Deru, M., 2005. Motion-Based Adaptation of Information Services for Mobile Users. In: Proceedings of the 10<sup>th</sup> International Conference on User Modeling (UM05).
4. Gemmell, J., Williams, L., Wood, K., Lueder, R., and Bell, G., 2004. Passive Capture and Ensuing Issues for a Personal Lifetime Store. In: Proceedings of The First ACM Workshop on Continuous Archival and Retrieval of Personal Experiences (CARPE '04), pp. 48–55.
5. Heckmann, D., 2005. Ubiquitous User Modeling. PhD thesis, Computer Science Department, Saarland University, Germany.
6. Plate, C., Basselin, N., Kröner, A., Schneider, M., Baldes, S., Dimitrova, V., and Jameson, A., 2006. Recommendation: New Functions for Augmented Memories. To appear in: Adaptive hypermedia and adaptive web-based systems: Proceedings of AH 2006.
7. Kröner, A., Heckmann, D., and Wahlster, W., 2006. SPECTER: Building, Exploiting, and Sharing Augmented Memories. In: Proceedings of the Workshop on Knowledge Sharing for Everyday Life (KSEL06), pp. 9–16.
8. Kuwahara, N., Noma, H., Kogure, K., Hagita, N., Tetsutani, N., and Iseki, H., 2003. Wearable Auto-Event-Recording of Medical Nursing. In: Proceedings of the 9<sup>th</sup> IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003).
9. Lamming, M., and Flynn, M., 1994. Forget-Me-Not: Intimate Computing in Support of Human Memory. In: Proceedings of FRIEND21, the 1994 International Symposium on Next Generation Human Interface.
10. Quan, D., Huynh, D., Karger, D.R., 2003. Haystack: A Platform for Authoring End User Semantic Web Applications. In: Proceedings of the 2<sup>nd</sup> International Semantic Web Conference (ISWC2003), pp. 738–753.
11. Schwarz, S., 2005. A Context Model for Personal Knowledge Management. In: Proceedings of the IJCAI Workshop on Modeling and Retrieval of Context (MRC '05).
12. Stathis, K., de Bruijn, O., and Macedo, S., 2002. Living memory: agent-based information management for connected local communities. In: Interacting with Computers, **14-6**, pp. 663–688.
13. Sumi, Y., and Mase, K., 2002. Supporting Awareness of Shared Interests and Experiences in Community. In: International Journal of Human-Computer Studies, **56-1**, pp. 127–146.
14. Wasinger, R., Krüger, A., and Jacobs, O., 2005. Integrating Intra and Extra Gestures into a Mobile and Multimodal Shopping Assistant. In: Proceedings of the 3<sup>rd</sup> International Conference on Pervasive Computing (Pervasive2005), pp. 297–314.
15. Wasinger, R., and Wahlster, W., 2006. The Anthropomorphized Product Shelf: Symmetric Multimodal Interaction with Instrumented Environments. In E. Aarts & J. L. Encarnac, ~ao (Eds.), Chapter in: True Visions: The Emergence of Ambient Intelligence.
16. West D., Quigley A., and Kay J., 2006. MEMENTO: A Digital Physical Scrapbook for Memory Sharing, In: Journal of Personal and Ubiquitous Computing Special Issue on Memory and sharing of experiences.