

Using an Extended Episodic Memory Within a Mobile Companion

Alexander Kröner, Stephan Baldes, Anthony Jameson, and Mathias Bauer

DFKI, German Research Center for Artificial Intelligence

Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany

<first name>.<last name>@dfki.de

ABSTRACT

We discuss and illustrate design principles that have emerged in our ongoing work on a context-aware, user-adaptive mobile personal assistant in which an extended episodic memory—the *personal journal*—plays a central role. The prototype system SPECTER keeps track of its user’s actions and affective states, and it collaborates with the user to create a personal journal and to learn a persistent user model. These sources of information in turn allow SPECTER to help the user with the planning and execution of actions, in particular in instrumented environments. Three principles appear to offer useful guidance in the design of this and similar systems: 1. an emphasis on user-controlled collaboration as opposed to autonomous system initiatives; 2. provision of diverse, multiple benefits to the user as a reward for the effort that the user must inevitably invest in collaboration with the system; and 3. support for diverse forms of collaboration that are well suited to different settings and situations. We illustrate the way in which these principles are guiding the design of SPECTER by discussing two aspects of the system that are currently being implemented and tested: (a) The provision of multiple, qualitatively different ways of interacting with the personal journal allows the user to contribute to its construction in various ways, depending on the user’s current situation—and also to derive multiple benefits from the stored information. (b) SPECTER’s collaborative methods for learning a user model give the user different ways in which to contribute essential knowledge to the learning process and to control the content of the learned model.

INTRODUCTION

There is growing agreement, reflected in the very existence of this workshop, that an extended episodic memory can constitute a valuable component of systems that serve as personal companions. But there remain numerous open

questions about how such a memory can be acquired and exploited. How much work will the user have to do to ensure that the extended memory is sufficiently complete and accurate; what form should this work take; and how can the user be motivated to do it? How can the system analyze the contents of the episodic memory so as to learn useful regularities that can in turn be exploited for the assistance of the user?

In this contribution, we discuss and illustrate three of the principles that we have found useful in our ongoing work on the relevant prototype system SPECTER. After sketching SPECTER’s functionality and comparing it with that of some representative related systems, we formulate and briefly justify three principles which appear to constitute a useful approach to addressing these requirements. Then, two aspects of SPECTER are discussed which illustrate how these principles can serve as a guide to the many inter-related design decisions that need to be made with systems that feature extended episodic memories.

BRIEF OVERVIEW OF SPECTER

Basic Functionality

SPECTER is a mobile personal assistant that is being developed and tested for three mutually complementary scenarios, involving shopping, company visits, and interaction at trade fairs. SPECTER exhibits the following characteristic set of interrelated functions: It extends its user’s perception by acquiring information from objects in instrumented environments and by recording (to the extent that is feasible) information about the user’s actions and affective states. It builds up a *personal journal* that stores this information. It uses the personal journal as a basis for the learning of a *user model*, which represents more general assumptions about the user (e.g., the user’s preferred ways of performing particular tasks). SPECTER refers to the information in the personal journal and user model when helping the user (a) to create plans for future actions and (b) to adapt and execute these plans when the time comes.

Relationships to Previous Work

The idea of building up a personal journal figured prominently in the early system FORGET-ME-NOT ([4]), though at that time the technology for communicating with objects in instrumented environments and for sensing the user's affective states was much less well developed than it is now. The much more recent project MYLIFEBITS ([3]) has similarly explored the possibility of maintaining an extensive record of a user's experience, but here the emphasis is more on managing recordings of various sorts (e.g., videos) than on storing more abstract representations. The idea of having a personal assistant learn a persistent user model can be found to some extent in many systems, such as the early CALENDAR APPRENTICE ([6]); but these systems have not used multifaceted personal journals as a basis for the learning, and there has been little emphasis in involving the user in the learning process. The idea of providing proactive, context-dependent assistance is reflected in many context-aware systems, such as shopping assistants and tourist guides; but there is much less emphasis on basing such assistance on a rich user model or on active collaboration by the user. The idea of collaboration between system and user has been emphasized in several projects; in particular, the COLLAGEN framework (see, e.g., [8], [9]) is being used explicitly within SPECTER.

DESIGN PRINCIPLES

In this section, we present and briefly justify three design principles which have proven useful in the design of SPECTER and which should be applicable to some extent to related systems.

User-System Collaboration as the Basic Interaction Model

In the foreseeable future, no computing device will be able to perform the functions listed above without receiving a significant amount of help from the user at least some of the time. For example, a system cannot in general record all actions of the user that do not involve an electronic device; and the user's affective reactions and evaluations are even more likely to be unrecognizable. Therefore, the user will have to provide some explicit input if a useful personal journal is to be built up. More generally, it is realistic to see each of the general functions served by the system as involving collaboration between system and user, although the exact division of labor can vary greatly from one case to the next.

A different justification for an emphasis on collaboration is the assumption that, even in cases where help by the user is not required, users will often want to be involved in the the system's processing to some extent, so as to be able to exert some control over it.

Provision of Multiple Functions

The necessary collaboration effort by users implied by the previous principle will not in general be invested by users unless they see the effort as (indirectly) leading to benefits that clearly justify the investment. Designers of a system that requires such collaboration should therefore try to ensure that the system provides multiple benefits as a reward for the user's investment. Even if only one or two particular types of benefit constituted the original motivation for the system's design, it may be possible and worthwhile to look for additional functions that take advantage of the same user input.

Flexible Scheduling and Realization of Collaboration

In addition to the user's motivation, another obstacle to obtaining adequate collaboration from the user is created by situational restrictions. For example, when the user is performing attention-demanding activities and/or interacting with SPECTER via a limited-bandwidth device, she may be able to provide little or no input.

A strategy for overcoming this problem is to look for ways of shifting the work required by the user to a setting where the user will have more attentional and computational resources available. For example, if the system can help the user to plan a shopping trip in advance, she may be able to use a high-bandwidth device and to supply information that she would not have time to supply while actually doing the shopping. Similarly, if SPECTER makes it worthwhile for the user to look back reflectively at the shopping trip after completing it, the user may be able to fill in some of the gaps in the record that the system has built up about the trip.

INTERACTION WITH THE PERSONAL JOURNAL

In accordance with the principle of multi-functionality, the data collection represented by the personal journal should be exploited in various ways. By providing methods for information retrieval, the journal may serve as extension of the user's personal memory for individual events and objects. A quite different type of application may use these data to provide feedback on how the user is spending her time, suggesting how she could adjust her time allocation in order to achieve her goals more effectively. Yet another way of exploiting the personal journal, discussed in the final major section below, is for SPECTER to mine its data in order to learn regularities that can serve as a basis for assistive actions.

The basis of these high-level interactions are so-called *journal entries*, which are created based upon signal input retrieved from an instrumented environment, or by means of abstraction. In the firmer case, fine-grained symbolic data are taken as input from sensors, and are directly stored in the journal. An exemplary setup of such an environment



Figure 1: An environment for testing SPECTER with RFID input: an instrumented shelf with RFID-enriched products. On the right-hand side, a laptop which performs shop communication, and provides SPECTER with input. End-user display and interaction are performed via a PDA.

is shown in Figure 1, where SPECTER has been connected with the RFID infrastructure created in the project REAL ([10]). The recorded signals serve as input for abstraction methods, which may range from syntactical, hard-coded translation to machine learning techniques.

Journal entries are usually created automatically, which leads to several requirements to SPECTER's user interface. Firstly, the kind of the recorded data and potentially the way how they have been retrieved should be transparent in order to strengthen the user's trust in the system. Hence the user needs a facility for inspecting the journal. Furthermore, content incorporated through entries may contain errors: measurement errors and wrong abstractions may occur, and the user herself might change with time her opinion about the correctness of previously created entries. Accordingly SPECTER requires a user interface, which enables the modification of journal content. Finally, with respect to the goal of a flexible scheduling of collaboration these requirements are completed by the need for an interface, which is adaptable to varying application scenarios.



Figure 2: The SPECTER browser, with a viewer for listing journal entries. In the upper area controls for navigation, in the lower area the viewer display.

Interface Approach

That need for flexibility is taken into account by a journal browser, which enables accessing the personal journal via so-called *viewers*. These realize varying data views of the information stored in the journal, a popular approach known from systems such as [3], [7], and [11]. An example of SPECTER's journal browser and a viewer for displaying lists of journal entries is shown in Figure 2.

In this framework, the browser as well as the viewers may be exchanged with respect to the given platform and interaction task. The browser is a central component that serves content requests from viewers, provides a repository of resources shared by several viewers. The latter ones include shared data such as display preferences, and shared user interface elements, such as access to common navigation facilities, and the viewer selection. That selection is in general performed automatically by the browser with respect to the display request, but may also be performed manually by the user if the viewer has registered itself within the browser's user interface.

Due to their varying functions, viewers may differ in their interaction not only with the user but also with the system itself. For instance, when displaying a list of journal entries, a viewer may be updated automatically when new entries (e.g., from sensors) arrive. That behavior might be

confusing if the user is just entering data using a form-like viewer. Therefore the browser relies on a feature mechanism to configure itself with respect to the viewer's preferences: a viewer's configuration includes a list of feature triggers, which may be applied by SPECTER components such as the browser in order to adapt their behavior to the given viewer. Following our previous example, this way the form editor may indicate that display updates are not granted while the viewer is active.

Navigation and Annotation

Navigation in the personal journal relies in the first place on *requests*, which provide a similar functionality as hyperlinks. Instead of Web addresses, they make reference to particular SPECTER components, optionally further described using form parameters. Additionally, they may carry a complex value encoded in XML that is submitted to the requested component. The user may apply these requests to browse the journal similarly as the Web, and may organize frequently used requests in a list of bookmarks.

An alternative way of navigating the journal is provided by the so-called *reminder points*. This specific kind of journal entry is created by the user during interaction with the environment with only one click (see the "!" button in the upper right corner of Figure 2). The rationale of these points is that the user might be too busy or distracted to provide detailed feedback. Nevertheless she might notice the need to adjust the system's behavior, and this need can be expressed via a reminder point. Later on at a more appropriate time and location for introspection, she may inspect the recorded reminder points and perform in collaboration with SPECTER the required adjustments.

Another way of dealing with journal entries is annotation. It provides a means of associating information with entries quite similar to the approach applied in [3]. In SPECTER, annotations serve in the first place as storage for information about how an entry performs with respect to selected aspects of the user model. Accordingly annotations include free text, references to other journal entries or Web pages, content categories, and ratings. Here content categories represent predefined content descriptions, provided by SPECTER for quick (and less precise) description of the kind of content. A rating expresses the performance of an entry with respect to a rating dimension selected from a predefined set (e.g., importance or evaluation).

Annotations are further described by a fixed set of meta data. These capture information about the annotation such as a privacy level, and the source of the annotation. The latter one is of particular importance, since the user has to stay informed about *who* has created an annotation - she herself, or SPECTER. An editor for entry annotations is shown in Figure 3. The form-like viewer provides feedback about the annotations associated with an entry, and

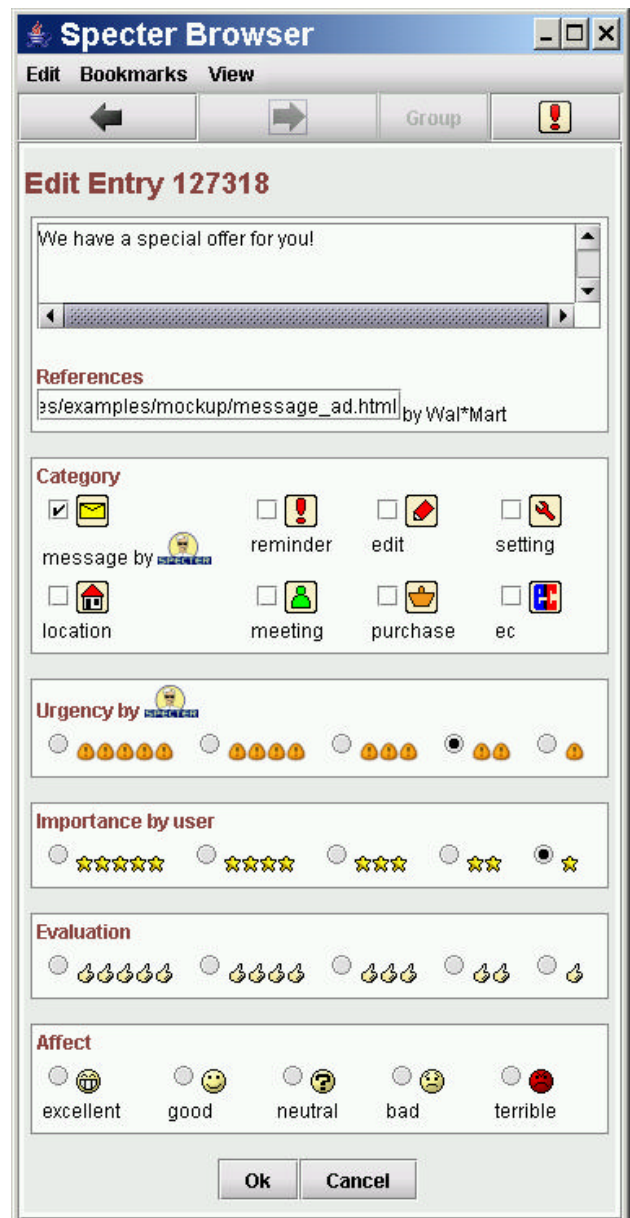


Figure 3: A form-like viewer that enables annotating a journal entry: a field for entering a free text comment, check boxes for content category selection, and select boxes for performing ratings. The selected values are marked with their sources (see "by").

enables editing them in part.

COLLABORATIVE LEARNING OF THE USER MODEL

As was mentioned above, one of the benefits offered by the personal journal is the ability of the system to learn a user model that can in turn serve as a knowledge source for intelligent assistance. Just as the acquisition of data for the personal journal is best viewed as a collaborative process,

the same is true of the process of learning the user model. The system brings to this process (a) a large amount of data from the personal journal, (b) a repertoire of learning techniques, and (c) a large amount of computing capacity (and unlimited “patience”) that can be applied to the learning task. But the user’s input will in general also be necessary: Her common-sense knowledge and introspective abilities can help to filter out spurious additions to the user model that would simply reflect chance regularities in her behavior (cf. [1]). Moreover, when there are several possible learned models that are equally well supported by the data, the user may reasonably prefer the model that makes the most intuitive sense to her. In short, the basic conception of SPECTER gives rise to a novel interface design challenge: How can a user who has no technical knowledge of machine learning or user modeling be allowed to collaborate in the process of learning a user model from data?

We will look at this problem in connection with one particular function of SPECTER’s user model: that of triggering the offering of services to the user.

SPECTER offers several types of service to the user. Some of these make use of external resources (e.g., technical devices such as printers), while others make use of internal functions of the system (e.g., retrieval of facts from the personal journal). If SPECTER simply waited for the user to request each possible service explicitly, many opportunities would be lost, simply because the user is not in general aware of all currently available and relevant services. Therefore, SPECTER tries to learn about regularities in the user’s behavior that will allow it to offer services at appropriate times. For example, if the learned user model indicates that the user is likely to want to perform a particular action within the next few minutes, SPECTER may offer to activate a service that will facilitate that action.

While there exist a number of approaches for collaborative learning that involve a human in the process of constructing a classification model (e.g., a decision tree), these approaches focus on supporting data analysts as opposed to essentially naive users (see, e.g., [2]). We are currently investigating the use of machine learning tools like TAR2 ([5]) that apply heuristics to produce imperfect, but easily understandable—and thus, modifiable—classification rules. Here we present an assistant component, the trigger editor, which gives the user intelligent suggestions for creating and modifying trigger rules for services.

Example: The EC Card Purchase Service

Our discussion will refer to the following example. Suppose that the user sometimes pays in stores with an EC card.¹ At one occasion the cashier rejects her card telling

¹For non-European readers: An EC card is like a credit card except that the funds are transferred to the recipient

her that her bank account provides insufficient funds to pay for her shopping. In order to prevent this embarrassing experience in the future, the user sets a reminder point, thus marking the current situation—and the resulting entry in the personal journal—to be dealt with later.

The rationale of this is that the user decided to create an automated service that triggers a status check of her bank account—a basic functionality provided by SPECTER—whenever an EC card payment is likely to occur. In order to do so she will create an abstract model of this particular type of situations using SPECTER’s machine-learning capabilities. Whenever a new shopping situation occurs, SPECTER will use this model to classify the situation and trigger the bank account check in case this classification indicates a high probability of the user using her EC card.

Identifying Training Examples

In a first step, SPECTER’s machine-learning component needs a number of *training examples*—previous shopping episodes stored in the personal journal that can be used to distinguish EC payments from “non-EC payments”. The system displays the entry marked with the reminder point and asks the user to indicate what is special about it. The user indicates the use of the EC card (“MeansOfPayment = EC card” in the personal journal) whereupon SPECTER looks for previous entries of the same category (shopping) with identical and differing values for MeansOfPayment and classifies these examples according to this value (“positive” for EC card, “negative” for all other values).

Learning a Decision Tree

Once the training data have been identified, SPECTER applies a machine-learning algorithm to create an appropriate classifier. One useful learning technique in this context is decision tree learning, which yields a relatively comprehensible type of model (see Figure 4). Even though users would seldom be willing or able to define a reasonably accurate decision tree entirely by hand, critiquing a decision tree proposed by the system may be a reasonably easy—and perhaps even enlightening—activity, if the user interface is designed well.

Figure 4 shows two decision trees that the user might deal with in connection with the EC Card Purchase service. Each node of a tree is labeled with an attribute, and each edge specifies a possible value (or range of values) for the attribute. Each leaf of the tree is labeled as positive or negative, indicating the decision that results if a path is traversed through the tree that leads to this leaf. In the case of service triggering, a positive result means that SPECTER should establish the goal of invoking the service. (Whether or not the service is actually invoked can depend on other

directly from the purchaser’s bank account.

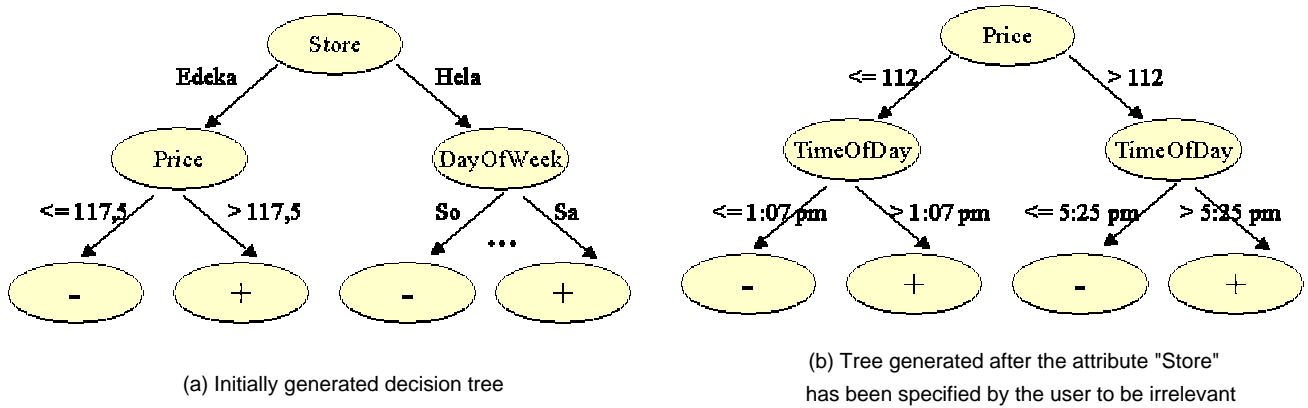


Figure 4: Two examples of decision trees that arise during the collaborative specification of a rule for triggering the service *EC card purchase*.

factors, such as the existence of competing goals.)²

When the system presents a learned tree such as the one in Figure 4 (a), the user can critique it in any of several ways, including: eliminating irrelevant attributes, selecting paths from the tree, and modifying split decisions. The question of what interface designs are best suited for this type of critiquing requires further exploration and user testing; the next subsection describes the critiquing interface currently being tested in SPECTER.

Critiquing of Decision Trees

Figure 5 shows the two main dialog boxes of the current decision tree editor, which is implemented as a viewer that runs within the browser. The interface allows the user to critique the current decision tree for a given type of decision step by step until she is satisfied with the result.

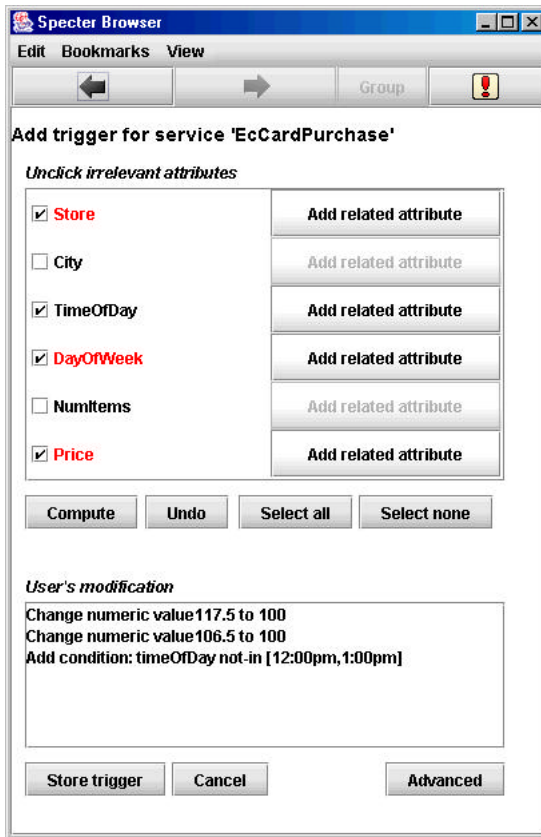
The standard interface hides the potential complexity of a decision tree as depicted in Figure 4 by merely listing the set of attributes used by the machine-learning component (see Figure 5 (a)). Depending on regularities occurring in the training data, some of these attributes—although well-suited to discriminate positive and negative examples in the decision tree—might make little sense from the user’s perspective. For example, if the user happened to use her EC card only in the morning in all shopping episodes recorded by SPECTER, then the attribute *TimeOfDay* will almost inevitably be used in the decision tree. The user’s background knowledge, however, enables her to easily identify such “meaningless” aspects and remove this attribute altogether, thus preventing its use for classification purposes.

²Attributes other than the ones shown in this example may also be relevant—e.g., the total cost of the other items that are still on the user’s shopping list and which therefore remain to be purchased on the same day.

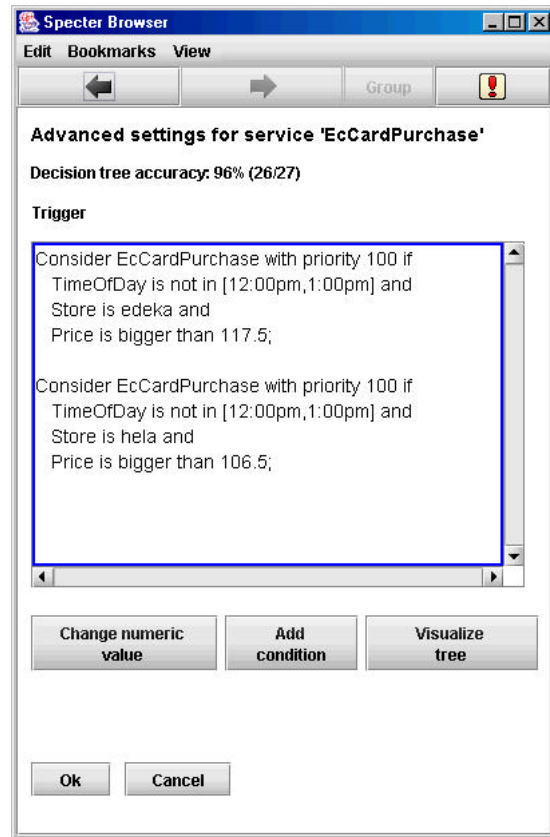
Another way of critiquing the decision tree is to replace an attribute by another one which is semantically related. To this end, whenever the user presses the “Add related attribute” button, SPECTER will identify concepts in the domain ontology that are in close proximity to the one represented by the attribute under consideration and generate appropriate attributes to be used in the decision tree. This way, the system’s capabilities to deal with regularities and *statistical* relationships among the training data is complemented by the user’s ability to deal with *semantic* interrelations.

Advanced users have even more options to influence the machine-learning component of SPECTER. She can directly inspect the classification model (either depicted as a decision tree, a set of rules (see Figure 5 (b)), or visualized in some other way yet to be investigated) and change the split criteria, i.e. the attribute values tested in a rule or tree (e.g. change *Price* from 117.50 to 100 as depicted in Figure 5). Doing so will of course affect the classification accuracy, i.e. the percentage of correct classifications of episodes from the personal journal. The user is informed about the current quality of the classification model and can bias the system to produce “false positives” rather than “false negatives” (which would mean that the bank account is checked even in some situations when the user will not use her EC card) or vice versa, depending on which error is more serious for the user.

One of the many design issues that we are exploring in connection with this interface is the question of whether (a) to present to the user a graphical depiction of the decision tree (as shown in Figure 4), (b) to stick to dialog boxes such as those in Figure 5, or (c) to offer a selection of interfaces. The principle of flexible scheduling and realization of collaboration suggests providing several different (though fundamentally consistent) views of a decision



(a) Simple critiquing options



(b) Advanced critiquing options

Figure 5: The basic (left) and advanced (right) dialog boxes in the current version of the SPECTER decision tree editor.

tree that are appropriate for different usage situations (e.g., a quick check on a rather unimportant decision tree vs. in-depth analysis and editing of a highly important one). The two different dialog boxes shown in Figure 5 represent a step in this direction.

CONCLUSION AND FUTURE WORK

In this contribution we described in part our ongoing work on SPECTER, a system that aims at assisting users in instrumented environments by means of an episodic memory. First results include a set of design principles, which have already proven their value as guides through a potentially immense design space. We believe they may be found helpful by designers of other systems featuring advanced personal memories. These days, a commercial product (Nokia LifeBlog³) was released that allows the user to create a simple version of what we call a personal journal. This provides a hint that this kind of functionality may sooner or later enter our daily lives.

³www.Nokia.com/lifeblog

We have illustrated how these design principles have been applied during the development of one of SPECTER's most important components: the *personal journal*. In the sequel we concentrated on the interface approach consisting of a journal browser and varying viewers. Here the browser allows navigating the journal contents using a hyperlink-like approach, and the viewers provide varying data views. An application of this interface is the decision tree editor. By means of our prototype implementation, we illustrated how the end user may construct triggers for services provided by SPECTER. This construction is basically an iterative process, where SPECTER is creating candidate decision trees that might serve as triggers, and the user is critiquing these trees. This process is supported by the editor in various ways including attribute selection, biasing the learning component with the aim to minimize consequences resulting from classification errors, and manual modification of the generated models.

Our next steps will include the extension of the view-based interface. For instance, we have to acquire information

about which kinds of viewers are actually required by the end user, and we have to evaluate implemented viewers. Additionally, the machine-learning component will need an interface that makes its complicated inferences and the resulting models accessible to even naive users.

ACKNOWLEDGMENTS

This research was supported by the German Ministry of Education and Research (BMB+F) under grant 524-40001-01 IW C03 (project SPECTER). Furthermore, we like to thank the REAL team for the valuable advice and support.

1. Gediminas Adomavicius and Alexander Tuzhilin. User profiling in personalization applications through rule discovery and validation. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 377–381, San Diego, CA, 1999.
2. M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: An interactive approach to decision tree construction. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 392–396, 1999.
3. J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. MyLifeBits: Fulfilling the Memex vision. *ACM Multimedia*, pages 235–238, 2002.
4. Mik Lamming and Mike Flynn. “Forget-me-not”: Intimate computing in support of human memory. In *Proceedings of FRIEND21, the 1994 International Symposium on Next Generation Human Interface, Meguro Gajoen, Japan*, Meguro Gajoen, Japan, 1994.
5. T. Menzies, E. Chiang, M. Feather, Y. Hu, and J. D. Kiper. Condensing uncertainty via incremental treatment learning. *Annals of Software Engineering, Special issue on Computational Intelligence*, 2002.
6. Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, 1994.
7. D. Quan, D. Huynh, and D. R. Karger. Haystack: A platform for authoring end user semantic web applications. In *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, pages 738–753, Sanibel Island, Florida, USA, 2003.
8. Charles Rich and Candace L. Sidner. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8:315–350, 1998.
9. Charles Rich, Candace L. Sidner, and Neal Lesh. COLLAGEN: Applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4):15–25, 2001.
10. M. Schneider. Towards a transparent proactive user interface for a shopping assistant. In A. Butz, C. Kray, A. Krüger, and A. Schmidt, editors, *Proceedings of the Workshop on Multi-User and Ubiquitous User Interfaces (MU3I 2004) SFB 378, Memo Nr. 83*, 2004.
11. C. Shen, B. Moghaddam, N. Lesh, and P. Beardsley. Personal Digital Historian: User interface design. In *Extended Abstracts of the 2001 Conference on Human Factors in Computing Systems*, 2001.