Musette: Modeling USEs and Tasks for Tracing Experience

Pierre-Antoine Champin, Yannick Prié, and Alain Mille

Lyon Research Center for Images and Intelligent Information Systems
LIRIS FRE 2672 CNRS — Lyon 1 University, France
{champin|yprie|amille}@liris.univ-lyon1.fr
http://liris.univ-lyon1.fr/

Abstract. In this article we present a new approach for modeling a user's experience in using a computer system. In this perspective, we developed the Musette approach. It relies on the building of a use trace conforming to a general use model, describing the objects and relations handled by the user of the computer system. This primitive trace can then be analyzed with respect to explained task signatures, in order to locate reusable episodes.

1 Introduction

It is a triteness to say that computers are widely used, for more and more various and numerous tasks like information organization, storage, communication, retrieval and sharing. Computing environments are increasingly customizable, in order to get closer to the user's practices, usages and, more generally, to their needs. To cope with this strong trend, it is necessary to take into account user's computer mediated tasks in order to be able to interpret in their context the traces left by the use of the computer environment "in contex".

Indeed, in many situations of user assistance, there can be a wide variety of questions that can be formulated, depending on the context of use. Let us stress the fact that this context has nothing to do with what is commonly addressed in so-called "contextual help": the latter is exclusively considering the computer environment context (e.g., selected item, current menu) while we are focusing on the user's context, in particular the task he is willing to perform. Case-Based Reasoning is widely used for Help Desk systems. Those need cases to be described through forms [6] or during a "conversation" [1] (which assumes that questions can be structured in order to fit the cases of a library). Therefore, case structure has to be defined in advance and case libraries are built according to it. Experience stored in them thus becomes scarcely exploitable for unanticipated questions. We claim instead that the challenge resides in being able to trace concrete experience in such a way that it could be re-exploited to provide useful answers to questions which were not fully defined from the start.

Our group has been attempting to propose solutions for this challenge across several research works [8, 5, 4, 3], which lead us to elaborate a general framework

for representing concrete experience in relation with its context of use. The MUSETTE global approach is presented in section 2, use model and traces are described with an example in section 3, the way to split a trace in episodes as potential reusable cases is detailed in section 4, and section 5 draws the opened perspectives through some current research projects.

2 Global Approach

The figure 1 presents the general framework of our approach, leading from the observation level to the experience reuse level, with two levels of experience modeling.

A *user* interacts with a *system*, leading to changes in the overall computing system (events, files...). An *observer agent*, observing these changes according to an *observation model*, generates a *primitive trace*, which conforms to a general *use model*. Then, a *generic trace analyzer* extracts significant *episodes* from the primitive trace, according to *explained task signatures*. These episodes can be (re-)used by *assistant agents*, which can assist the user either as agents clearly distinguished from the system (direct assistance), or by modification of the system (system mediated assistance).

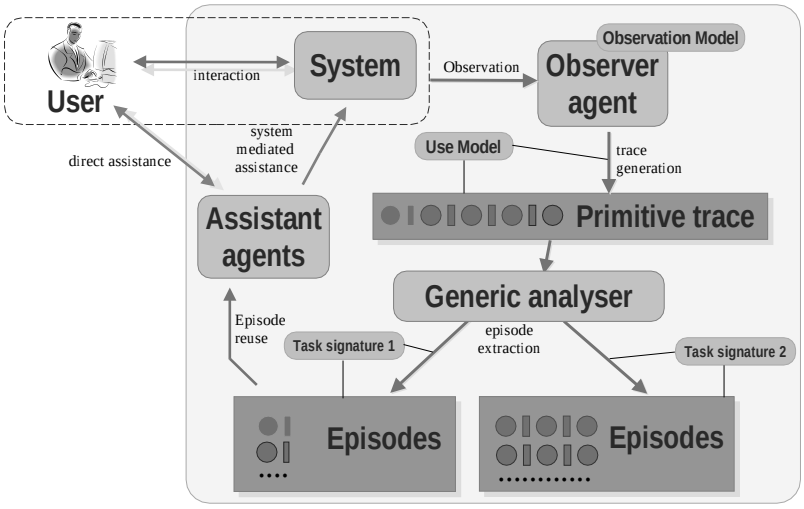

Fig. 1. *Introducing our approach and vocabulary.*

The interest of the MUSETTE approach relies in assisting the user on past episodes, reused to facilitate his current task. Let us give a possible scenario: the user is currently exploiting a computer environment for a particular task. He asks for the help of an assistant agent to retrieve a way to complete a part of his task with the environment, on the basis of a target episode elaborated from the current trace, in accordance with a particular task signature. He is

then proposed the most adaptable episodes resulting from the analysis of the available use traces, and whose adaptation is guided by the assessment of the differences between the target episode and the most adaptable episode reminded.

3 Use Model and Traces

3.1 What do we observe and how?

Since the MUSETTE approach requires a representation, as a primitive trace, of the interactions between the user and the system, the first step in applying that approach is to *decide* what the trace will be made of, and how it will actually be achieved. Those questions must be answered respectively by the use model and the observation model, in order to build the *observer agent* in charge of producing the trace (cf. fig. 1).

¿From a general point of view, the trace will be composed of *objects of interest* (OI). Those can be of one of the three following categories: entities, events and relations. Entities can be characterized as objects *being present* to the user in their interaction with the system, while events can be characterized as objects *happening* during the interaction. Relations are binary, and can imply either entities or events.

The use model for a particular system describes what kind of entities, events and relations will actually be observable to produce the primitive trace. Additional constraints, including ones over the internal structure of the OIs, can also be part of the use model. However, they are specified by the language used to implement the MUSETTE approach, and are not detailed here. Let us emphasize on the fact that the objects of interest used to describe a system, as the word 'interest' implies, depend on the particular focus, however general, of the use model. Deciding how the user-system interaction will be observed is bounded to be biased by the use model designer's goals; the ideal aim of the use model is however to be as general and "task-neutral" as possible.

Describing the components of the to-be-produced trace is not sufficient to build an observer, though. The observation model still has to be described, as a set of means to access relevant data in the system, as well as rules constraining the process of producing the trace —*e.g.*, which relevant subset of all the observable entities must be written to the trace at a given moment?

Unlike the use model, the observatiothat model is not specified by the MUSETTE approach for the moment. Indeed, producing a trace conforming with a given use model, by observing the interactions between the user and the system highly depends on the system itself, on which we deliberately made no assumptions. We envision that software systems will be more and more self-explicative, and more and more expandable, therefore generic observers, controlled by a set of system-dedicated formal observation rules, may eventually become a reality. However in the current state of the art, an *ad-hoc* observer has to be built for every system and with a particular use model in mind, and the observation model has to be hard-coded manually in such an observer.

3.2 Modeling traces

Once the observer agent has been specified by the use model and the (possibly hard-coded) observation model, it can produce traces from the observation of the interactions between the user and the system.

The structure of the trace is not limited to a continuous stream of entities and events, possibly in relations with one another. Indeed, entities are used to represent the *state* of the system at a given moment (or during a period *considered* to be an instant in the context of the use model). On the other hand, events happen in the *transitory* period between two states. Hence the grouping, in the trace, of OIs according to their category, into an alternate sequence of states and transitions.

It is worth noting that states and transitions in the MUSETTE approach merely have a temporal role. They are not intended to convey any predefined causal meaning; in particular, the events in a transition are *not* bound to logically *imply* the following state. A transition in the trace can indeed contain one or several events, which can be related or not (see transition 6 in figure 2). A transition could even contain no event at all —but be required by the model in order to separate two consecutive states. Of course, a given use model *may* provide such causal semantics by means of specific kinds of entities, events and relations that it defines.

3.3 Example

Figure 2 gives an example of a rather simplistic use model of a web browser, together with a fragment of a primitive trace conforming to that use model.

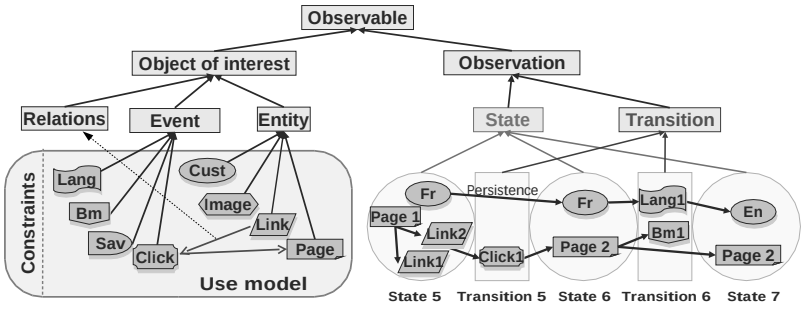

Fig. 2. *A web navigation trace.*

Entities of the use models are web pages (Page), hyperlinks (Link), images (Image) and customization features (Cust). Events are the user clicking on a hyperlink (Click), their saving a page or an image locally (Save), their setting a bookmark on a page (Bm) and their changing the preferred language of the browser (Lang). In the following, we will consider that pages also have an internal attribute url containing their address.

For the sake of readability, all the possible relations are not represented in the use model, but the relation from a link entity to a click event, and from a click event to a page entity —obviously meaning that the corresponding link was clicked, leading to the corresponding page to be displayed. Other relations appear in the trace only in the figure, but would in principle be defined in the use model as well: from page to link ("the page contains the link"), from page to bookmarking ("the bookmarking applies to the page"), from page to page ("the page is reloaded") from customization feature to language change, then to another customization feature (where the customization features represent the previous and new language settings). The Persistence relation is also worth commenting: it allows, in this particular use model, to express the fact that entities of different states actually represent the *same* object from the system. The identity of objects may not be necessary in any use model, so it is not a part of the core MUSETTE model. However, the Persistence relation is one possible mean of expressing it in a particular use model, if needed.

4 Extasi and Episodes

Assuming that the use model enables an appropriate description of the interactions of the user with the system, the user's experience is potentially retrievable from the primitive trace. More precisely, we call an *episode* any part of the trace corresponding to a specific experience in performing a specific task, and which can be reused in a similar situation.

4.1 Explained Task Signatures

We need a mean to locate episodes in the primitive trace. Episodes related to a particular task usually share some common features: involving the same kind of entities or events, co-occurring or occurring in a given order, etc. More generally, we consider that these common features can be expressed by a) a pattern of the graph constituted by the objects of interest and their relations, b) constraints on the relative positions of OIs in the trace, c) language-dependent constraints on the internal structure of OIs.

Once these common features have been identified for a particular task, they can be considered as a *signature* of this task. Indeed, their instantiation in the trace can be interpreted as an evidence of the user performing that task in the corresponding period.

Episodes are not limited to parts of the trace instantiating a task signature, though: once identified the task performed by the user, one can improve its interpretation of the trace. The roles that the OIs play in that trace may be easier to understand; additional relations, not captured by the observer agent, may be inferred; etc. Therefore, the episode can be *explained*, by a number of information coming from the fact that it has been recognized as an occurrence of a particular task. These explanations are *annotations* of one or several elements of the signature, ranging from free text (to be used by users) to formally defined symbols (to be used by specialized software assistants).

A number of *explained task signatures* (Extasi) thus allow the generic analyzer (cf. figure 1) to produce, from the primitive trace, episodes instantiating the signature part of Extasis and annotated by their explanation part.

4.2 Episodes and cases

Episodes are not strictly speaking cases in the CBR sense, since the primitive trace is not a case base. It can rather be considered as a *potential* case base for every Extasi allowing to extract reusable episodes from it. In other words, the primitive trace has no *a priori* structure constraining the kind of cases it can contain, hence the kind of problems it can help to solve. Cases are elaborated differently depending on the identified tasks to assist and their corresponding Extasis, of which there can be as many as required.

Of course, there is a trade-off between the versatility and the efficiency of such a system, and it highly relies on the use model. Should it be too general, different tasks will be indistinguishable from one another, and episodes will be to vague to be efficiently reused. Should it be too specific, some tasks outside its focus will become plainly undetectable.

4.3 Example

We show here how two common web browsing tasks can be identified in our example use model, and how their signature can be explained by annotations on the matching OIS.

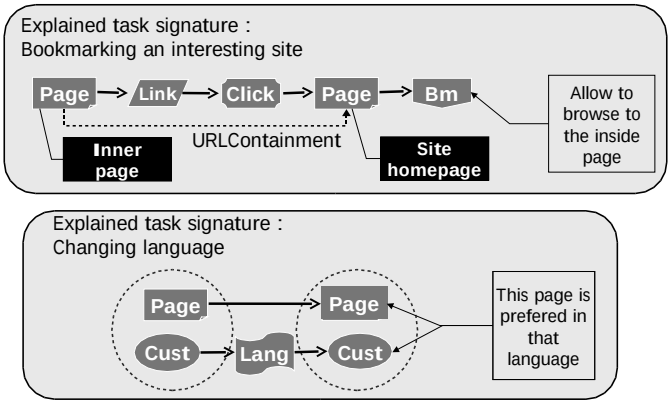

Fig. 3. *Explained Task Signatures and episodes.*

When discovering an interesting web page, we often want to put a bookmark on the *site* containing this page, since it probably contains other interestingly related pages. Such tasks can easily be spotted in the primitive trace: from a first Page, a second one is accessed whose url is a prefix of the former one's url, then a Bm (bookmark) event occurs in relation with the latter Page. The upper

EXTASI in figure 3 has such a signature (note the dotted arrow representing a language-dependent constraint over url attributes). Both pages in the signature are annotated by 'Inside page' and 'Site first page'. The Bm event is also annotated with textual explanation.

Another task consists in changing the web browser language setting in order to view different versions of a web page available in different languages. Our use model allows to detect such a task, since changes in the language settings are represented by a Lang event, as well as the corresponding page change (without clicking on a link). The lower EXTASI in figure 3 has such a signature, with dotted circles representing a co-occurrence constraint of Page OIs with Cust OIs representing the currently set language. Here again, a textual annotation provides explanations about the OIs.

One will remark that both EXTASIs presented here have a matching in the example trace of figure 2: from state 5 to transition 6 is an episode of "bookmarking an interesting site", and from state 6 to state 7 is an episode of "changing language".

5 Discussion

5.1 Related works

Task modeling has been widely studied by the knowledge engineering community. This modeling can be performed in the context designing knowledge based systems [10] and is increasingly used for knowledge management purpose [7]. In the same way, ontology design is more and more explicitly task driven [9]. Knowledge engineering proposes models, methods and tools, often implying great efforts to elicit tasks from users' actual behaviors. On the other hand, computer assistants helping users to perform a task, only need it to be expressed in terms of relevant resources for the assistance, rather than full fledged task modeling.

Nevertheless, in an experience reuse approach, it is necessary to spot use traces pieces which would refer to some user's task. Hence, we proposed EXTASIs as simplified views of task models. Such signatures can be designed according to classical knowledge engineering approaches in order to be integrated in a computer environment, before the system is ever used; but they can also be designed on demand at run-time by the user himself. This approach assumes that end users are co-designers of their assistance environment. As it has been discussed in section 4.2, this can be considered as a customization of case elaboration knowledge in the CBR cycle.

5.2 Perspectives

We are currently applying the MUSETTE approach to the design of several assistant agents concerned with various tools:

- a tool for cartographical organizing and P2P searching of documentary information[1]. We will reuse significant use episodes for presentation adaptation and ranking adaptation.

[1] See http://www.human-links.com/

– a tool for digital simulation design and results exchange among groups of experts. On the first level, we consider that each user is helped by a digital *alter-ego* using the MUSETTE approach to trace his experience in using the system, and can help him along his tasks. On the second level, we consider societies of *alter-egos*, exchanging experience and constructing shared use experience. The new model is called MAZETTE for Multi-agent MUSETTE.
– a basic navigation tool, so as to study how we could annotate web resources by their uses, and reuse these annotations [2] for user adaptation and help in the context of the Semantic Web.

The application of the MUSETTE model to an interaction needs a prototyping phase: in many cases, observer agents will have to be coded from scratch, and therefore need significant development efforts. To go further than plain pen-paper modeling, we need a tool for rapidly prototyping use model, primitive trace and first task signatures. We are currently building such a tool, based on PROTÉGÉ[2]. Indeed, we hope that our efforts will lead to the development of a real *methodology* for experience reuse according to MUSETTE principles.

References

1. D.-W. Aha, L. Breslow, and H. Muñoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14(1):9–32, 2001.
2. P-A. Champin and Y. Prié. *MUSETTE: uses-based annotation for the Semantic Web*. IOS Press. to appear.
3. P-A. Champin. *Modéliser l'expérience pour en assister la réutilisation : de la Conception Assistée par Ordinateur au Web Sémantique*. Thèse de doctorat en informatique, Université Claude Bernard - Lyon 1, Villeurbanne (FR), 2003.
4. F. Corvaisier, A. Mille, and J.-M. Pinon. Information retrieval on the world wide web using a decision making system. In *RIAO 1997*, pages 284–295, Jun 1997.
5. E. Egyed-Zsigmond, Y. Prié, A. Mille, and J.-M. Pinon. A graph-based audiovisual document annotation and browsing system. In *RIAO 2000*, volume 2, pages 1381–1389, Apr 2000.
6. O. Herbeaux and A. Mille. Accelere: a case-based design assistant for closed cell rubber industry. *Knowledge-Based Systems*, 12:231–238, 1999.
7. H. Holz, A. Könnecker, and F. Maurer. Task-specific knowledge management in a process-centred see. In K.-D. Althoff, R.L. Feldmann, and W. Müller, editors, *Advances in Learning Software Organizations, Proc. of LSO 2001*, number 2176 in LNCS, Sep 2001.
8. Y. Prié and A. Mille. Reuse of knowledge containers: a local semantics approach. In M. Minor, editor, *Workshop on Flexible Strategies for Maintaining Knowledge Containers, ECAI 2000*, number 33, pages 38–45, Aug 2000.
9. C. Reynaud and F Tort. Using explicit ontologies to create problem-solving methods. *International Journal of Human-Computer Studies*, 46:339–364, 1997.
10. G. Schreiber, A. Hakkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Welde, and B. Wielinga. *Knowledge Engineering and Management: The CommonKADS methodology*. The MIT Press, 1999.

[2] See http://protege.stanford.edu/