MUSETTE: a Framework for Knowledge Capture from Experience

Pierre Antoine Champin, Yannick Prié, Alain Mille

LIRIS - Bât Nautibus - UFR Informatique
Université Claude Bernard Lyon 1 / F-69622 Villeurbanne Cedex
prenom.nom@liris.cnrs.fr
http://liris.cnrs.fr/prenom.nom

Résumé. Nous présentons dans cet article une nouvelle approche de modélisation de l'expérience d'utilisation d'un système informatique, avec pour objectif de réutiliser cette expérience en contexte pour assister l'utilisateur à effectuer sa tâche. Le modèle se base sur la capture d'une trace d'utilisation conforme à un modèle d'utilisation général, lequel décrit les objets et les relations manipulés par l'utilisateur du système informatique visé. Cette trace primitive peut être considérée comme une base de connaissances neutre par rapport à la tâche, qui peut être analysée *a posteriori* à l'aide de signatures de tâche expliquées permettant d'y localiser des épisodes signifiants qui pourront être réutilisés par des assistants logiciels comme connaissances contextualisées. Quatre scénarios illustrent cette approche.

1. Introduction

It is a triteness to say that computers are widely used, for more and more various and numerous tasks, which mainly rely on "information management": information organization, storage, communication, retrieval, sharing… Furthermore, information management environments get increasingly customisable, in order to get closer to users' practices, usages and, more generally, to their needs of handling information whatever its form. For example, we could consider that an environment composed of a/ the web as a resource provider (documents, data) and b/ a tool for viewing and editing HTML documents, is adapted to any task involving information gathering and publication. The spectrum of computer-mediated tasks becomes wider, and tools for performing these tasks become more versatile and customisable. Since, on the other hand, they have more and more (often inexperienced) users, there is a increasing need for assisting the latter in their tasks while using tools or sets of tools. As a consequence, there is a need to design software agents as *assistants*, which would take advantage of knowledge describing the task at hand. Indeed it becomes necessary to take into account user's *tasks* in order to be able to interpret, in their context, the traces left by the use of the computer environment. Of course, 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 notion of "context" has nothing to do with what is commonly addressed in the 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 or she is willing to perform.

More precisely, we do consider *two* kinds of tasks. First, we consider tasks that are well identified, for which assistance would rely on knowledge described in carefully designed ontologies. But it is also important to consider a second kind of tasks, which are hard to

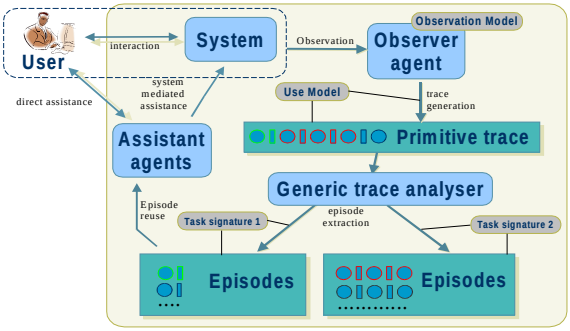

FIG. 1 - *Introducing our approach and vocabulary*

anticipate, and should be recognized from their manifestations and defined on the fly, considering actual experience of use of the system. We would like to address these tasks, so our main question is: how is it possible to model and capture experience in using a system, so that it can be reused as *knowledge* for user assistance? Moreover, how could it be that the assistance itself could evolve with concrete experience?

Our group, which comes from the Case-Based Reasoning (CBR) research field as a mean of tracking and reusing experience, has been attempting to propose solutions for this challenge across previous research works [Prié and Mille, 2000; Champin and Prié, 2003]. This led us to elaborate Musette (Modelling USEs and Tasks for Tracing Experience), a general framework for representing concrete experience in relation with its context of use.

Next section presents the Musette global approach. The two following sections present the notions of *use model* and *traces*, then of *explained task signature* as a mean to *split* a trace in *episodes*, or potential reusable cases. The fifth section deals with scenarios demonstrating different kinds of Musette assistants in action, while the sixth section is devoted to discussing related works, principally in the field of CBR and task modelling.

2. Global approach

Figure 1 presents the general framework of our approach, leading (clockwise) from the observation level (upper-right corner) to the experience reuse level (lower-left), with two levels of experience modelling. A *user* interacts with a *syste*m, leading to changes in this computing system (events, files...). An *observer agen*t, observing these changes according to an *observation model*, generates a *primitive trac*e, which conforms to a general *use model*. Then, a *generic trace analyser* extracts significant *episodes* from the primitive trace, according to *explained task signature*s. These episodes can be (re)used by *assistant agent*s, which can assist the user either as agents clearly distinct from the system (direct assistance), or by modification of the system (system mediated assistance). In the latter case, the assistance interaction can in turn be observed by the observer agent, allowing assistance episodes to be also reused. Let us emphasize the fact that Musette is only a general framework. It can be implemented using various languages or representation formalisms, provided that those enable the representation of every component of the Musette approach.

3. Use model and traces

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 constructed. 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). Basically, the trace will be composed of *Objects of Interest* (OI). Those can belong to 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. Objects make sense for both the computer application and for the user. 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. Let us emphasize 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. 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 observation model is not specified by the Musette approach for the moment, and 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.

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 relation 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, turns the trace 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. Of course, a given use model can add such causal semantics to specific kinds of entities, events and relations it defines.

4. Extasis 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. Pieces of the trace are recognized as episodes related to a particular task thanks to *EXplained TAsk SIgnatures* (Extasis).

We need a mean to locate episodes in the primitive trace. More generally, we consider that common features can be expressed by: 1/ a pattern of the graph constituted by the objects of interest (e.g. events and entities) and their relations; 2/ constraints on the relative positions of OIs in the trace (e.g. co-occurrences in observations – States or Transitions –, distance between observations in the trace); 3/ language-dependent constraints on the internal structure of OIs (e.g. attribute values).

Once these common features have been identified for a particular task, they can be considered 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 their 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, and not present in the use model, may be inferred; etc. Therefore, the episode can be annotated, or *explaine*d, by a number of information coming from the fact that it has been recognized as an occurrence of a particular task. Explanations may take the form of free text annotation (that a human could interpret), or of formal knowledge annotation (aimed at an automated agent interpretation).

5. Scenarios

Let us demonstrate how different kinds of assistants can be designed on top of the Musette approach, considering a web browsing scenario. A specific assistant can take into account one given task by reusing the episode instantiating the corresponding Extasi. For example, we could build an assistant that would notify the user whenever she is browsing a page in an interesting site —that is, whenever it would find an episode matching an Extasi "Bookmarking an interesting site" whose site is the current site. Such a specific assistant can apply case based reasoning mechanisms [Aamodt and Plaza, 1994] to reuse relevant episodes in a given context. However, an advantage of the Musette approach is that the same knowledge base (the primitive traces) can be shared by many different assistants extracting different episodes from it. Moreover, a new assistant can be added without needing to build a completely new knowledge base: only a new Extasi has to be provided in order to extract new episodes from existing traces.

The second scenario involves a *generic* assistant capable of handling any number of tasks, via their Extasi, in a regular way: exploring the current trace with respect to a particular task. For instance, the user could browse all the interesting sites she has already visited, or all the pages she preferred to read in French. He could also be suggested to put a bookmark on a page, or change the language setting, when he has already done so with a similar page. Again, the assistant could even perform those actions automatically. The advantage of Musette here is that every particular task is reified and elicited by the corresponding Extasi, and that the explanations provided in the Extasi can be used to guide the assistant as well as to make it understandable by the user.

In the third scenario, the user explicitly asks the assistant for help. The latter then has to identify the task being performed by the user in order to select appropriate episodes for reuse. Task identification can range from explicit selection of an Extasi by the user, to fully automatic detection in the current trace according to the available Extasis. An intermediate solution is for the system to propose various interpretations of the user's activity, based on the partially instantiated Extasis and corresponding explanations. The user may then accept or reject those interpretations until a consensus is reached.

The fourth scenario goes a step further than the previous one. Assume none of the proposed Extasis satisfy the user. The assistant could provide a mean for him to specify even more precisely what task he is willing to perform. For example, our user does neither want to ask someone, nor to refine his query by adding more keywords, but would rather submit the same query to a more specialized search engine (e.g., field specific or language specific). What the user is actually doing is describing a *new* task with its own Extasi —without

specifying it completely, of course, since the user wants help to find the appropriate specialized search engine. Even though this Extasi was not known before, this task might have been already performed, and therefore some traces might contain episodes matching this newly created Extasi. The user can indeed describe a previously unknown task on the fly, and still obtain assistance about it. We see here how the separation in Musette, between the general use model and specific task signatures, leverages user assistanceMerci de respecter les styles suivants pour les listes.

6. Related works

Long before digital computers and the Internet, [Bush, 1945] was dreaming of what he called the "MEMEX", a device that would capture everything a scientist looked at or commented about, creating a trail of information which he or other scientists could later retrieve. In the 90's, [Hill et al., 1992] saw cumulative interactions of users with digital objects as a form of wear and tear, comparable to the torn pages in a manual. More recently, [Wexelblat and Maes, 1997] have developed a tool enabling users to visualize the paths the others have taken through a site. These preliminary works focused on what could be presented directly to the user as footprints to follow for her (unformulated) request. They aimed to work as generic assistants. Some works focused on the user task to put in context such assistance [Farell et al., 2000; Francisco-Revilla and Breimer, 2000] while others tried to capture general web navigation episodes on static signatures [Corvaisier et al., 1997; Jaczinski and Trousse, 1998] or, like [Takano et al., 2000], used past procedure cases to help their use in specific applications. These last works rely on the Case-Based Reasoning paradigm, which is widely used for Help Desk systems. Those need cases to be described through forms [Herbeaux, 1999] or during a "conversation" (which assumes that questions can be structured in order to fit the cases of a library) [Aha et al., 2001]. Therefore, case structure has to be defined in advance and case libraries are built according to it. The experience stored in them thus becomes scarcely exploitable for unanticipated questions. If cases have to be relaxed as use episodes dynamically found in the trace, it is nevertheless necessary to take into account the user tasks context. Task modelling has been widely studied by the knowledge engineering community. This modelling can be performed in the context designing knowledge-based systems [Schreiber et al., 1999] and is increasingly used for KM purpose [Holz et al., 2001]. In the same way, ontology design is more and more explicitly task driven [Reynaud, 1997]. Knowledge engineering proposes models, methods and tools, often implying great efforts to elicit tasks from users' actual behaviours. 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 modelling. Nevertheless, in an experience reuse approach, it is necessary to spot pieces of the use traces that 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.

7. Conclusion

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 modelling, we need a graphical tool for rapidly prototyping use model, primitive traces and first task signatures. We developed such a prototype as a plug-in for PROTÉGÉ (http://protege.stanford.edu/), using OKBC as an implementation language for MUSETTE. Apart from the systems we have already built, upon whose creation experience we designed MUSETTE, and therefore which – to some extend – implemented parts of the MUSETTE approach, we are now more specifically applying this approach to the design of several assistant agents concerned with various tools.

References

[Aamodt and Plaza, 1994] A. Aamodt, E. Plaza. Case-based reasoning; Foundational issues, methodological variations, and system approaches. AI Communications, Vol.7, No.1

[Aha et al., 2001] D.-W. Aha, L. Breslow, and H. Munoz-Avila. Conversational case-based reasoning. Applied Intelligence, 14(1):9–32, 2001.

[Bush, 1945] V. Bush, As we may think, Atlantic Monthly, 176(1), 101-108, 1945

[Champin and Prié, 2003] P-A. Champin and Y. Prié. Musette: uses-based annotation for the Semantic Web, In *Annotation for the Semantic We*b, IOS Press, Amsterdam (NL) , 2003.

[Corvaisier et al., 1997] F. Corvaisier, A. Mille, and J.-M. Pinon. Information retrieval on the WWW using a decision making system. In RIAO 1997, pages 284–295, Jun 1997.

[Farell et al., 2000] R. Farrell, P. Fairweather and E. Brumer. A task based architecture for application aware adjuncts, ACM International Conference on Intelligent User Interfaces, New Orleans, LA USA, pages 82-85, 2000

[Francisco-Revilla and Breimer, 2000] L. Francisco-Revilla and E. Breimer. Adaptive medical information delivery combining user, task and situation models, Internation Conference on Intelligent Interfaces, New Orleans, LA USA, pages 94-97, 2000

[Hill et al., 1992] W.C. Hill, J.D. Hollan, D. Wroblewski and T. McCandless. Edit Wear and Read Wear. In Proc. of ACM Conference on Human Factors in Computing Systems, CHI'92, pages 3-9, New York ACM Press, 1992

[Holz et al., 2001] H. Holz, A. Könnecker, and F. Maurer. Task-specific knowledge management in a process-centred see. In Althoff, Feldmann and Müller, editors, Advances in Learning Software Organizations, Proc. of LSO 2001, LNCS 2176, 2001.

[Jaczinski and Trousse, 1998] M. Jaczinski, B. Trousse. WWW Assisted Browsing by Reusing Past Navigations of a group of users, in Advances in Case-Based Reasoning, 4th EWCBR, Dublin, Ireland, 1998

[Prié and Mille, 2000] 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.

[Reynaud, 1997] C. Reynaud and F Tort. Using explicit ontologies to create problem-solving methods. Inter-national Journal of Human-Computer Studies, 46:339–364, 1997.

[Schreiber et al., 1999] 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.

[Takano et al., 2000] A Takano, Y. Yurugi and A. Kaenaegami. Procedure Based Help Desk System, ACM IUI 2000, New Orleans, LA USA, pages 264-272, 2000

[Wexelblat and Maes, 1997] A. Wexelblat and P. Maes. Footprints: History-rich web browsing, In RIAO'97, pages 75-84, 1997.