Trace-based framework for Experience Management and Engineering

Julien Laflaquière[2], Lotfi S. Settouti[1], Yannick Prié[1], Alain Mille[1]

[1] LIRIS laboratory, UMR 5205 CNRS, Bat. Nautibus, Université Claude Bernard Lyon 1, 69622 Villeurbanne CEDEX, France

[2] Institut Charles Delaunay, University of Technology of Troyes, FRE CNRS 2848, Tech-CICO laboratory, 10010 Troyes CEDEX, France
PhD research supported in part by Champage-Ardenne district grant

{yprie, lsettout, jlaflaqu, amille}@liris.cnrs.fr

Abstract. The paper deals with experience management in computer-mediated environments. It particularly focuses on complex tasks whose support relies more on *experience* than on *knowledge*. The presented approach is based on the "use traces" concept to investigate the "activity reflexivity" as a first step in experience management, and experience sharing and reusing as possible applications. The paper also outlines the framework supporting Trace-Based Systems creation. Traces, trace models and trace life cycle are formally defined. The main parts of the framework architecture: collecting, transformation, visualization, and query systems are also detailed.

1 Introduction

It's a common to say that computers are widely used, for more and more various and numerous tasks, usually with a strong documentary dimension. Hence, more and more activities are performed in *computer-mediated environments*, to achieve at least in part these tasks. Such environments are composed of both digital contents *and* the tools that allow manipulating them. Among all the challenges that knowledge management (KM) has to take up, we focus in this paper on *complex mediated tasks support*. As we will show, complex tasks have characteristics that make them hold more onto *experience* than onto simple *knowledge* (in the sense of Sun [1]). Taking into account these characteristics and adopting a specific approach to manage them is a stake for the emerging young experience management and engineering field [1]. In particular, we will focus in this paper on the notion of *interaction trace* for experience management and experience engineering.

This paper has two main parts. Section 2 outlines why and how one can consider *traces of environment use* as a tool to tackle some experience management (EM) objectives, from the activity reflexivity to the experience reuse. Section 3 describes necessary theoretical concepts towards a generic *framework for traces-based systems*.

2 Traces for interaction experience management and engineering: rationale and scenarios

We define a complex task in a computer-mediated environment, as a *high level*, *dynamic*, *open-ended*, and strongly *context-dependent* task. Economic Intelligence (EI) on the Web is a good example of this type of task [2]: EI related tasks associate difficulties from process-based classical approach of EI and those from complex Web search tools manipulation (high-level); the tasks are defined step by step during its realization (dynamic); there is neither exact goal, nor identified means to reach this goal (open-ended); and overall, the realization of the tasks depends on the context, here the information found (or not found) during the process (context-dependent). Because of these characteristics, complex tasks are difficult to formalize and to model, and consequently, to integrate in a classical KM process.

As a hint towards a possible solution, we investigate the concept of *activity reflexivity* as a means of making users conscious of their own activity and in a sense conscious of the experience they are living. We consider that an activity is reflexive if its realization provides some information about itself, i.e. leaves an accounted for representation of itself. Several research works underline the role played by activity reflexivity for supporting and enhancing the realization of complex tasks [3]. For instance in Computer Supported Collaborative Learning, reflexivity allows a deeper learning [4] while in Information Retrieval, it is considered as allowing a better activity structuration, avoiding user's disorientation [5].

2.1 Traces for reflexivity in computer mediated activity

To obtain activity reflexivity in complex mediated tasks, one has to be able to propose a kind of activity representation to the user. This representation has to be constructed from the observation of the mediating environment. Furthermore two opposite general kinds of approaches can be distinguished. *Quantitative* approaches are based on log-files. These log-files, obtained by passive observation, are used to calculate some statistical insights about the user's exploitation of his environment, and/or to profile the user himself [6]. *Qualitative* approaches are mostly proposed in ethnographic and ergonomics research studies [7]. These approaches consist in careful observation, *in situ*, often completed by audio and video recordings, allowing auto-confrontation for observed users. Both quantitative and qualitative approaches remain unsatisfactory to deal with reflexivity: the first are disconnected from miscellaneous contexts in which activity takes place, while these contexts are fundamental elements of the experience; the second are time-expensive and are concerned with a very short period of time. We claim that an intermediate solution is possible, based on *use traces* of computer-mediated environment. This is what our *Trace-Based Systems* aim to do (section 3).

Our main goal is to deal with use traces that "make sense" for the considered user. The structure of a trace and its abstraction level have to be pertinent, accordingly to the tasks the user realizes. In other words, the description power of a trace has to correspond to his reflexivity needs.

With this objective in mind, our approach can be broadly decomposed as follows: a) a modeling of the use is performed, *with* the user participation and *in the context* of a particular task. This allows underlining some salient interaction components called *Objects of interest* (entities, events and relations), which are relevant for him, in the context of *his use* of the environment in the context of his task (there is neither causal nor hierarchical *a priori* constraints and objects of interest definitions belong to a *use model*) and b) an observer agent is built that can accordingly generate traces from the interaction between the user and his environment. Such traces are a graph of objects of interest, where entities and events are linked with relations (see Fig. 1.).

One advantage of this kind of trace is to be "readable" by the user, with an adapted graph visualization tool (which has a critical and complex role in this context). The traces outline the activity structure and its global shape. They can give a continuous representation of the user's activity in the computer-mediated environment as a historic classic view, but it is not necessarily useful. In fact, a trace can be usefully exploited to *contextualize* its proper components. Each object of interest, embedded in the trace, can be contextualized by the context of its use, i.e. by the role it takes relatively to the other components. To be able to replace each object of interest in its context is one dimension of activity reflexivity in the computer-mediated environment.

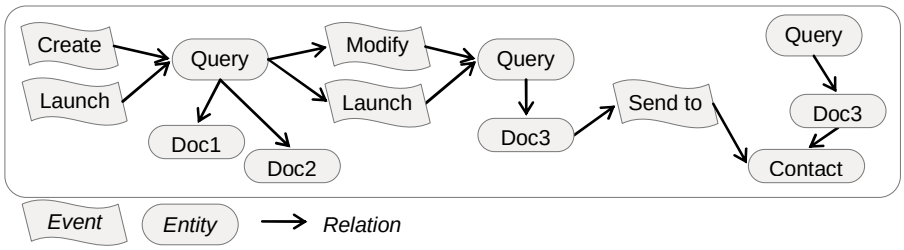

Fig. 1. A part of use trace, as a graph of objects of interest: in this simplistic example, a user of Human-Links® (EI tool), *launches* a *query* with some keywords. He has to *modify* the query to obtain a pertinent result, so he *sends* the *document* to a *contact* (other user of the software). For detailed presentation see [2].

2.2 Traces for experience sharing and reusing

Some applications of our approach have shown that such traces can "make sense" for users as well as for activity analysts [8]. So, one can examine the question of traces exploitation towards the initial issue of activity reflexivity. Two problems seem particularly interesting because they fit with EM purposes: *experience share* and *experience reuse*. Since use traces as the ones we described above can hold a kind of activity reflexivity (e.g. the interpretation of his own trace by a user), it is possible for a user to exploit the same traces as a means to *share* his use/interaction experience. We do not mean that the trace, or part of it, *is* the experience, and that we can exchange it directly (it will probably never be the case). We rather consider the trace as a means, a tool or a *support* for experience sharing.

It can be seen as a "boundary object" [9], useful in several well known situations in KM, when a person has to explain his work face to face with some material supports (analysis of problem situation in a design project or memory project constitution for example). Making such use of a trace needs tackling the trace visualization issue. We do not think as Morse [10] that visualization is the only challenge, but it remains a real and central problem to deal with real trace-based experience sharing.

In KM and EM fields, another recurrent problem is *experience reuse*. In our case we are interested in reusing experience of environment use in a particular task. There are several ways of dealing with this problem, and we want to underline that we are just considering one of them. We can extend the approach detailed in section 2.1: in addition to the objects of interest, we define *tasks signatures* as patterns in the trace graph which characterizes a particular task realization. The goal here is not to describe precisely one pattern for one task, but to be able to know when a particular task occurs. Thanks to tasks signatures, we can extract some significant *episodes* in the trace. One episode at least corresponds to one task signature (a web search for example), and can be considered as reusable piece of use experience. How do we reuse it?

Suppose the user of a computer-mediated environment has a problem and asks his system for help. The system tries to identify in current use the beginning of a known signature. Then it can find and present to the user the past episodes that correspond to this known signature. These episodes, seen as pieces of past use experience, can indicate to the user how in the past he coped with that problem (or not), in a similar task, maybe in a different context. We stress the fact that this past experience reuse holds on the user himself: presenting past episodes does not necessarily give a solution; it can just act as a mnesic help or an inspiration source, a means to facilitate the task at hand. If no signature is identified the user can define a new one by himself, for the system to parse the trace and find useful episodes. Here the challenge lies in providing traces that can be reused to cope with problems that were not determined when the tracing system was constructed [11].

Through tree examples at different levels (activity reflexivity, experience sharing, and experience reuse), we have shown *how* a traces-based system could be useful to support user activity with an EM point of view, and *why* it opens new perspectives to instrument EM approaches for complex computer-mediated tasks. The first part of this paper also gave some clues that suggest the underlying complexity of such systems, and we therefore have to lean on a rigorous theoretical approach and methodological tools. This is why we developed the *traces-based systems framework*. In the following section, we will describe traces-based systems and sketch the idea of how they could be used in order to support the trace exploitation process.

3 Towards a Trace-Based System Framework for EME

In order to tackle the traces management challenge, we present a trace-based systems framework. This framework allows the definition of trace-based systems (TBS), which holds on traces exploitation, e.g. the production, the transformation and the visualization of traces. We will first define our notions of trace and trace lifecycle, before setting up a formal framework for traces-based system.

We introduce traces as *temporal sequences* of *observed items*. "Temporal sequence" indicates the existence of an order-relation that organizes trace data relatively to a time base. "Observed items" indicates that trace data result from an observation. This definition allows us to argue that a trace is a numeric document which reveals temporally located data resulting from an observation. As shown in section 2, we are mainly concerned with use traces of computer-mediated environment. In this context, a trace is considered as a recording of a computer-mediated activity that is potentially constructed from a variety of sources (log-files, videos, transcripts, etc.).

We divide the trace lifecycle into different stages. Firstly, *collecting* deals with deciding what to collect and how to collect it into a basic trace. For example, a basic trace would gather all the events occurring during an interaction as objects of interest. A basic trace is a TBS entry-level trace. Secondly, *transformation* deals with automatically or manually filtering, rearranging or adding information to the basic traces in order to meet the description needs of the user. In our example basic trace, the collected data can be irrelevant or too detailed: *selection transformation* can be used for separating data and sequences of interest from the "noise", while *abstraction transformation* (data aggregation), can be used to provide higher-level objects of interest to the user. Thirdly, *presentation* deals with traces visualization and involves choosing what to present and how to present it to the user. It is important that each of these stages be considered separately to avoid any confusion.

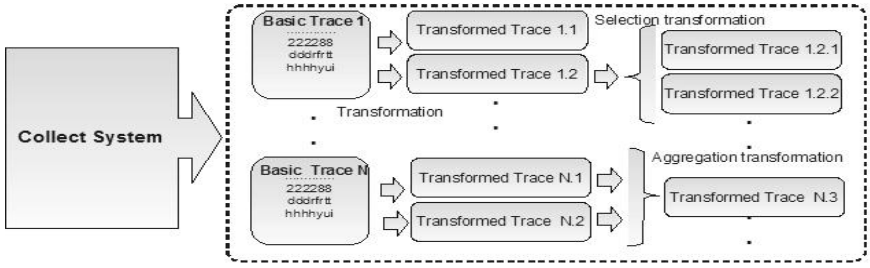

Fig. 2. Traces are collected, transformed and visualized when needed.

3.1 Traces and traces-models

The trace lifecycle will be supported by traces-based systems. Before describing their various components, it is necessary to present the notion of trace model. In a TBS, each trace must always be associated to an explicit *trace model*, which allows its interpretation. A trace model can be considered as a model describing the vocabulary and the structure of the traces it is associated with. For example, trace log-files often rely on the *Common Logfile Format (CLF)*, traces XML documents are described by DTD trace models or schemas, *etc.* In this work, we will use the ontological model as a general trace model compatible with models that will actually be instantiated in a TBS (simple log models, schema-based models, *etc.*). Formally:

Definition 1: *A Trace Model is an ontology* $M_T = (C; \leq_C ; \leq_R; R; T; A; \sigma_A ; \sigma_R)$ *consisting of a set of concepts C organized in a hierarchy with an order relation* $\leq_C$*, a set of relations R organized with* $\leq R$*, a relation signature* $\sigma_R : R \rightarrow C \times C$*, a set of data types T , a set of attributes A, and a attribute signature* $\sigma_A : A \rightarrow C \times T$.

The following definition concerns the general formal definition of a trace:

Definition 2*: A trace is a quintuplet* $(M_T, D_p, O_{tr}, R_t, R_s)$ *where* M_T *is the associated trace model;* D_p *is a temporal domain (T, <) with T a set of time instants and < an order on T;* O_{tr} *is a set of objects O,* $O_{tr}=\{O_0, O_1, \ldots, O_n\}$ *such as* $\forall O_i \in O_{tr}$, $f(O_i) \in C$, *with f a labelling function* $f: O_{tr} \rightarrow C$. $R_t \subseteq D_p \times D_p \times O_{tr}$ *is a relation representing the temporal links between objects and time intervals.* $R_s \subseteq O_{tr} \times O_{tr}$ *is a relation representing the structural links between objects, and* $\forall R_{si} \in R_s$, $g(R_{si}) \in C$, *with g a labelling function g:* $R_s \rightarrow C$.

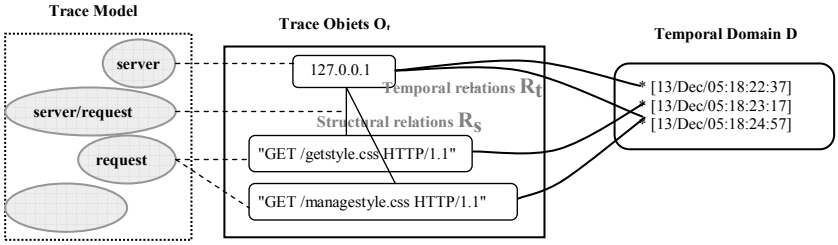

Fig. 3. In this example, the trace model is a set of concepts (server, request, username). Trace objects (one server and two requests) are related to the temporal domain D_p through R_t (note the server is related to a time interval). Traces objects have structural relations through R_s.

3.2 Traces-based system architecture

The different parts of a TBS are described in Figure 2. The *TBS Kernel* is the core of the system. It is composed of a *transformation system* and a *traces base.* The traces base is a set of traces and allows permanent storage and access to the traces. A real traces base can be for example a set of XML files or a temporal database.

As seen above, a main part of trace exploitation holds on their transformation. *The transformation system* performs transformations τ during the modification and handling of the traces. The transformation system allows (1) modifying the trace by enriching or filtering its data, (2) modifying the model of trace, (3) updating the traces base or making automatic transformations by using transformation models (sets of formal rules expressed in a rule-based language).

The *query system* allows formulating and resolving queries on the traces base. Queries can be made on traces considering the models they are associated with, their transformations, their objects in their interrelations and time inscription. For instance one can consider all the traces of an individual at a certain transformation level, and query for episodes involving objects in structural and temporal relations (for example two web pages related to the same server, visited in a same hour).

The *collecting system* is a set of processes allowing the conversion of several *tracing sources* into basic traces by appropriate tools. Tracing sources are files or data streams in an unspecified explicit format, from which basic traces will be constructed. The simplest collecting system would do simple data integration and time-based synchronization of the various sources. A more complicated system would incorporate different tracing sources in an intrinsic iterative process for instanced by mixing video annotation and log files use, *etc*.

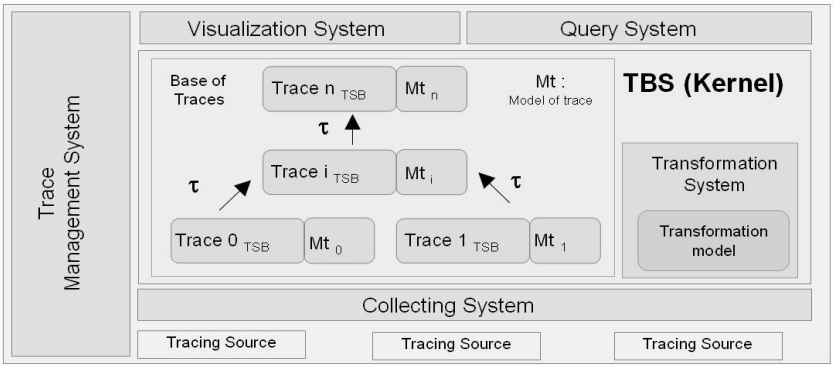

Fig. 2. Trace-based framework architecture.

The *visualization system* allows visualizing the traces and thus facilitates analysis and interpretation. The visualization system is a set of techniques for presenting traces (basic traces or transformed traces) in a visual form allowing human's direct exploitation. A well-known visualization involves using a timeline that describes the temporal sequence of data. If video is implied as a tracing source, hence in the temporal domain, visualization can link video and traces. For example, play a video synchronously with trace visualization for better interpretation (as in [8]).

Lastly, the *trace management system* (TMS) allows managing the various models involved in a TBS: trace models, transformation models, queries, *etc*. It allows the archiving of the traces, the granting of rights on traces base and offers also an integrated management of trace lifecycle. Indeed, TMS holds a traceability of transformations and operations performing in TBS. This functionality allows the management of the experience use of TBS and facilitates the trace management and engineering.

4 Conclusion

This paper has presented our approach which suggests to consider use traces, constructed from the interaction between a user and his computer-mediated environment, as a tool for EM approach. In the first part, we have outlined that activity reflexivity is a key step to deal with interaction or use experience of a user. After explanations about the kind of traces involved in our approach, we have proposed examples of use trace exploitation to investigate experience sharing and experience reusing.

In the second part of this paper, we have detailed the framework developed to support use traces exploitation with Trace Based Systems (TBS). We have formally defined the notions of trace and trace model, and have given some details on the different parts of a TBS that are needed to support trace processing.

The general approach we proposed in this short paper is backed by different domain-oriented studies we carried out in our research team. Future work implies developing the general approach both on the theoretical and practical sides. The theoretical work will focus on traces, reflexivity and experience management for individuals and collectives, and formal trace modeling, transforming and querying. The Practical work will be devoted to the implementation of the general framework and its instantiation into various real trace-based systems (especially on collaborative systems), so as to validate our assumptions, and to conduct complete experiments in EME.

5 References

1. Sun, Z., Finnie, G. Experience Management in Knowledge Management. In Proc. 9th International Conference, Part I, KES 2005, Melbourne, Australia, (2005) p.979.
2. Laflaquière, J., Champin, P.A., Prié, Y., Mille, A. Approche de modélisation de l'expérience d'utilisation de systèmes complexes pour l'assistance aux tâches de veille informatiquement médiées, In ISKO-France'2005, INIST/CNRS, Nancy, France, (2005) 209-230 (in french)
3. Laflaquière, J., Ciaccia, A. : Facilitation de tâches informatiquement médiées : une approche centrée sur la réflexivité de l'utilisation, 6th workshop Colloque des jeunes chercheurs en sciences cognitives 2005, Bordeaux, France (2005), 217(in french)
4. Weerasinghe, A., Mitrovic, A. : Facilitating deep learnig through self-explanation in an open-ended domain, International journal of Knowledge-based and Intelligent Engineering systems, IOS Press, 10 (2006) 3-19
5. Castelli C., Colasso L., Molinari A., Getting lost in Hyperspace: Lessons Learned and Future Directions, in *ED-MEDIA 96/ED-TELECOM 96,* Boston, (1996), 124-130
6. Mobasher, B., Cooley, R., Srivastava, J. 2000a. Automatic personalization based on web usage mining. Commun. ACM, 43 8 (August), 142–151.
7. Dekker S., Nyce, J.M., How can ergonomics influence design? Moving from research findings to future systems. Ergonomics, vol. 47, N. 15 (2004), 1624 - 1639.
8. Georgeon, O., Bellet, T., Mille, A., Letisserand, D., Martine, R.: Driver behaviour modelling and cognitive engineering tools development in order to assess driver situation awareness. Workshop on Modelling Driver Behaviour in Automotive Environments. (2005), Ispra.
9. Star, S. L. 1989. The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. M. Huhns and L. Gasser, eds. *Readings in Distributed Artificial Intelligence.* Morgan Kaufman, Menlo Park, CA.
10. Morse, E. and Steves, M. CollabLogger: A Tool for Visualizing Groups at Work. Proceedings of WETICE 2000, Workshops on Enabling Technologies:Infrastructure for Collaborative Enterprises, IEEE Computer Society, (2000), 104-109.
11. Champin, P.-A., Prié, Y., & Mille, A. : MUSETTE: a framework for knowledge capture from experience. In Proceedings: Egc'04, (2004), Clermont Ferrand, France (in french)
12. Luotonen A. "The Common Log file Format." http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html

Description of modifications

- *Grammar corrections have been completed.*
- *A short definition of "activity reflexivity" has been added earlier in the paper (section 2).*