IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITB12592

This paper appears in the book, *Emerging Trends and Challenges in Information Technology Management, Volume 1 and Volume 2* edited by Mehdi Khosrow-Pour © 2006, Idea Group Inc.

OUML: A Language to Map Observed Usages to BPML

Jean-Mathias Heraud, Laure France, & Joseph Heili
Graduate School of Chambery, 12 avenue d'Annecy, 73181 LeBourget du Lac, France, {jm_heraud, l_france, j_heili}@esc-chambery.fr

INTRODUCTION

Business Process Management (BPM) has widely spread out recently. More and more companies use systems based on BPM [7]. One of the main reasons of this enthusiasm for the BPM is the possibility to improve a process [6] by recording the process progress in order to perform optimizations in the process definition/implementation [8]. However, the process progress recording is unfortunately an approach too limited to help the process reengineering. We thus propose an approach that consists of separating the usage collect and the process progress first, and then, of linking these elements thanks to a description language of usages: OUML (Observed Usage Modeling Language). This approach, coupled to a matching system, allows us to go beyond these limits.

The paper is organized as follows: We show in section 2 the limits of the process progress recording to help the process reengineering. Then, we propose in section 3 our approach of the collect, and then, in section 4, we present OUML (Observed Usage Modeling Language), our description language of usages. In section 5, we discuss this approach. Finally, we show in the discussion the potentialities of our approach for the process reengineering.

BPMS: LIMITED POSSIBILITIES FOR PROCESS REENGINEERING

"BPM refers to a set of activities which organizations can perform to either optimize their business processes or adapt them to new organizational needs" [11] and BPMS refers to the software systems which support BPM. Usually, BPM is described by three phases: design, execution and monitoring. Currently, most of the BPM Systems enable the observation of the activated process progress [4] for the monitoring phase. This observation is performed by recording the different states of the activities composing these processes. The transitions and duration are then deduced from the previous observations on a common timeline. These possibilities of observation help to determine which processes have been performed totally or partially, which ones have been successful or failed. Indeed, this type of observation permits a comparative analysis between recommended process and performed process. This comparison allows us to check whether the foreseen sequences are fulfilled or some activities are omitted or permuted.

However, this type of observation is limited due to the fact that we can observe only what was recommended. Any activity not foreseen in the process will not be observed and thus will not be detected. Consequently, the reengineering help possibilities are restricted to adjustment, suppressions or permutations of process activities. Indeed, the process quality manager will not be able to rely on this observation to determine which activity must be added to the process to meet a need. To raise this type of limits, we thus propose another type of observation that we present in the next section.

OBSERVING THE CLIENT COMPUTERS

Keyloggers are small applications that are executed on a computer client. These programs are usually executed *without the knowledge* of the computer users, since their purpose is to spy the users. The basic functioning principle is to record every keystrokes in a text file. In our research framework, we use this type of software to trace during experiments the user activities by recording all these keystrokes, the list of the computer processes that are executed, the titles of the dialog boxes displayed on the screen and their content. In order to lead experiments, we developed a software agent capable of logging the user activity, and of exchanging the collected information with other agents of the same kind (observer agents) or agents specialized in the collect and the processing of this information type (collector agents). However, we can note that this multi-agent architecture is justified by the work we undertake on the collaborative issues, beyond the scope of this paper (see [2] for more details).

The use of keyloggers may be opposed to anonymity which constitutes one of the aspects of privacy. This assertion has to be moderated according to the legislations of the various countries where such systems are used. Within the framework of our experiments, we retained the four following principles inspired of the Information and Telecommunication Access Principles Position Statement [11]:

- No data should be collected from people not aware
- Individuals should have the choice to decide how their information is used and distributed
- Individuals should have the right to view any data files about them and the right to contest the completeness and accuracy of those files
- Organizations must ensure that the data are secure from unauthorized access

The data logged by the observer agents are stored in a generic log format (GLF) [9]. Indeed, to be able to obtain the log sources that store the information in other formats such as apache logs, we defined rewriting rules enabling the transformation of the main log formats into GLF. The collector agent uses these rules to be interfaced with multiple log sources active on the computer (other than the user activity, for instance, the logs Apache on a web server).

The agents can communicate to each other via messages in GLF format, but these messages are at a low level of abstraction. Therefore, we need a higher level, namely the trace described in OUML format. We introduce OUML in the following section.

DESCRIPTION OF USAGES: OUML

We defined a language that allows us to represent the performed usages in a more abstract way than just a set of raw logs. The idea is to define an intermediate level between logs and the process modeled by BPML [1], by using a vocabulary close to BPML, but with the own logic of the logs. Indeed, there is a gap between raw logs and the process while considering two major "things", activities and roles. Therefore, we need to make individual activities emerge from raw logs. For instance, we use the notions of activity and sequence defined in BPML; however, we did not take into account the choice operator, since only one user path is observed in the logs. Nevertheless, this operator could be reintroduced later on at another level, when we will need to combine several traces or define a trace query language. We present here the general lines of the OUML language.

BASIC STRUCTURE OF AN OBSERVED USAGE

Any sequence of observed log can be expressed by an object "trace". A trace can be composed of complex activities in ways described in the next subsection. These activities are constituted by combining simple or complex activities, where simple activities can be split into two categories: the user activities and the system activities. The user activities are:

- "User's action": Represents the task of the user (name, role, resource, etc.). This activity can have the status "started", "completed" or "aborted".
- "User's inactivity": Expresses a lap of time with no user's activity observed. This activity can have the status "started" or "completed".

The five other types of simple activities rely on the System activity:

- "System Assign": Assigns a new value to a property.
- "System Call": Instantiates a process and waits for it to complete.
- "System Delay": Expresses the passage of time.
- "System Spawn": Instantiates a process without waiting for it to complete.
- "System Synch": Synchronizes on a signal.

COMPOSITION OF ACTIVITIES

Several activities can be performed at the same time. In order to better understand the logical sequencing of the activities, we propose the following operators:

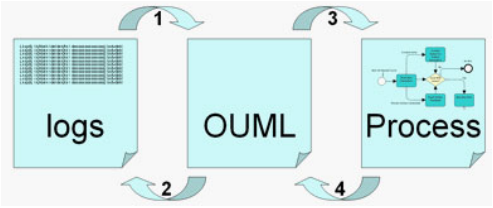
- All(A,B) : the operator tells that the complex activities A and B are executed in parallel.
- Sequence(A,B) : the operator tells that the complex activities A and B are executed sequentially (B after A).

MAPPING BETWEEN LOGS, OUML AND BPM

Our objective is to define the four mappings shown in Figure 1. To reach this goal, we chose to follow an incremental approach based on an experiment that we done in 2004 [5]. We are currently concerned with the first step (1) and we plan to follow the three others in the order used in the figure.

- **Generate OUML traces from raw logs:** In a first stage, we try to generate OUML traces from the logs of this experiment. This stage allows us to define a mapping (1) between raw logs and an OUML trace. The method we retain to generate OUML traces is based on a set of rules and patterns.
- **OUML as a raw log query language:** In a second stage, we use the obtained OUML traces to test a mapping (2) towards other sources of logs observed during the same experiment. This stage allows us to test OUML as a raw log query language, since several sets of raw logs correspond to one OUML trace.
- **Mapping between OUML and BPML:** With the same set of OUML traces, we then test the mapping (3) between OUML and the BPML definition of the process recommended during the experiment.
- **BPML as an OUML query language:** When this stage of modeling all the usages observed during the experiment is completed, we can work on mapping (4) OUML traces with the recommended process. This last stage allows us to define a query language for OUML trace.

Figure 1. Mapping between logs, OUML and BPM



CONCLUSION

In this paper, we proposed a language to represent observed usages. This language is particularly useful when observations are captured from numerous sources. This work is still in progress, and thus mapping definitions between raw logs, OUML and BPML, lead to an evolution of the OUML language.

The prospects at the end of this experiment are numerous. By creating a relationship between raw logs and a process modeling language such as BPML, OUML makes it possible to envisage the detection of the process execution in the observed usages. This detection may be handled a posteriori but also in real time. The applications of such a system could be, for instance, fraud detection, awareness, or knowledge management.

In a more general way, increasing the richness of the observation level and clarifying the sequencing of a process are paramount stages. We already proposed some visualization tools to represent the traces in [3]. All this work will make it possible to improve the quality of the process implemented and, thereafter, to allow us to improve their level of maturity.

REFERENCES

[1] Arkin A. (2002) "Business Process Modeling Language" From Business Process Management Initiative (BPMI). Contributions by: Blondeau D., Ghalimi I., Jekeli W., Pogliani S., Pryor M., Riemer K., Smith H., Trickovic I., White S.A., 98 pages.

[2] Carron T., Heraud J.-M., (2005) "A SMA architecture based on usages observation to support collaborative tasks", Working Paper, Syscom, university of Savoie.

[3] France L., Heraud J.-M., Marty J.-C., Carron T. (2005) "Help through visualization to compare learners' activities to recommended learning scenarios.", The Vth IEEE International Conference on Advanced Learning Technologies, Taipei, Taiwan.

[4] Grigoria D., Casatib F., Castellanosb M., Dayalb U., Sayalb M., (2004), "Business Process Intelligence", Computer in industry, Volume 53, pp.321-343.

[5] Heraud J.-M., Marty J.-C., France L., Carron T. (2005) "Helping the Interpretation of Web Logs: Application to Learning Scenario Improvement." Workshop Usage analysis in learning systems, AIED2005: XIIth International Conference on Artificial Intelligence in Education, Amsterdam, Netherland

[6] Marty J.-C., Heraud J.-M., Carron T., France L., (2004) "A quality approach for collaborative learning scenarios", Learning Technology Newsletter, publication of IEEE Computer Society Technical Committee on Learning Technology, Volume 6, Issue 4, pp 46-48.

[7] MacVittie L. (2005) "IT Detours On the Road to BPM", Intelligent Enterprise Magazine, August 2005.

[8] Paulk M.C., Curtis B., Chrissis M.B., Weber C.V., "Capability Maturity Model - Version 1.1." IEEE Software, pp. 18-27, 1993.

[9] Rambaud S., Armanet L., Jost C., (2005) "Generic Log Format (GLF) : un langage pivot pour combiner plusieurs sources de logs", Master thesis, university of Savoie, in french.

[10] http://en.wikipedia.org/wiki/Business_Process_Management

[11] http://www.cla.ca/about/access.htm