Facilitating Workflow Evolution in an Advanced Object Environment

Dickson K.W. CHIU[1], Qing LI[2] AND Kamalakar KARLAPALEM[3]
[1]*Department of Computer Science, University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*
[2]*Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong*
[3] *Indian Institute of Information Technology, Gachibowli, Hyderabad, INDIA*
email: kwchiu@cs.ust.hk, csqli@cityu.edu.hk, kamal@iiit.net.

Workflow is automation of a business process. A Workflow Management System (WFMS) is a system that assists in defining, managing and executing workflows. To support flexible enactment and adaptive features, such as on-line workflow evolution and exception handling, a WFMS requires advanced modeling functionality. As workflow evolution requires the modification of workflow definitions or adding ECA rules to the system during work in progress, an advanced schema evolution capability is required at run-time. It should be noted that the resolutions based on schema evolution are general-purpose ones, which can help reduce the occurrence of additional exceptions. As such, we have developed ADOME-WFMS based on an Advanced Object Modeling Environment (ADOME [4] -- an active OODBMS with role and dynamic schema support), with a novel exception centric approach. The contribution and objectives of our research with respect to workflow evolution are as follows. The meta-modeling approach to activity modeling, which facilitates flexible workflow enactment and evolution, is novel. Moreover, our work offers a coherent practical solution from workflow modeling, enactment, to evolution. While much other work has concentrated on theoretic aspects of workflow evolution, our mechanisms for workflow evolution and various workflow evolution policies are well established with detailed but simple algorithms, based on our expressive activity meta-model. Even though there has been a lot of work in workflow evolution and workflow exception, our augmented solution to exception handling based on workflow evolution is a new one. The use of ADOME-WFMS in supporting flexible workflow enactment and workflow evolution through effective web interface facilities has been shown as a trendy and promising direction. Furthermore, as our underlying ADOME environment is essentially an active OODBMS, many techniques we have developed for ADOME-WFMS should be applicable to WFMSs based on similar environments.

In ADOME-WFMS, we are advocating a three-level meta-modeling approach wherein workflows, capabilities, exceptions, and handlers are defined at a meta-level, in addition to full object orientation. This approach enables full inheritance of activity properties (such as rules and re-execution mode) down the composition hierarchy and applies to each of the activity / task instances. From this meta-level schema, users can define all other classes for the WFMS, including activity schemas, agents, roles, exceptions and handlers. Further, from these schemas WFMS objects (in particular, activity instances) can be instantiated. This contributes a substantial improvement to WFMS modeling using relational models. Extensive reuse is also facilitated as discussed in [2].

ADOME-WFMS has the required facilities (cf. [3]) for supporting various types of workflow evolution. In particular, besides conventional exception handling resolutions, the *Human Intervention Manager* sub-module also accepts update of workflow on-line. In contrast, there are currently few WFMSs having such facilities for supporting the whole spectrum of exception-handling resolutions, especially those relating to workflow evolution. We have also designed a web-based interface, upon which the user can access a workflow evolution menu page and choose any of the suggested resolutions to be persistent, or enter schema evolution operations, update of workflow and/or enter new ECA rules (subject to enforcement strategies).

[1] proposed a complete, minimal, and consistent set of workflow evolution primitives. These primitives can be used for static and dynamic workflow evolution, including for migration of individual workflow instances. They are divided into *declaration primitives* and *flow primitives. Declarative primitives* modify the declaration of workflow variables and their default values. These are well supported in ADOME-WFMS, since the underlying ADOME layer actually has the same corresponding primitives for schema evolution.

Declarative primitives include *Addvar* (*a: Activity, v: Variable_Name, t: Type, d: DefaultValue*) for adding an attribute *v* of type *t* with initial value to the schema declaration of an activity / task *a* if the attribute does not exist in the activity schema; and *RemoveVar* (*a: Activity, v: Variable_Name*) for removing the attribute *v* from the schema declaration of an activity / task *a*. In particular, changing

Proceedings of the Seventh International Conference on Database Systems for Advanced Applications (DASFAA '01)
0-7695-0996-7/01 $10.00 © 2001 IEEE

default values of *workflow system variables* have important semantics, such as change of task capability requirement, default rule binding. *Flow primitives* modify the flow structure of the activity schema. With reference to the meta-activity schema declaration, all these primitives, in essence, update the default value of the initial value of the workflow system variable *Activity_Graph*. On the other hand, if we want to apply flow primitives to evolve an activity instance, we update the *Activity_Graph* attribute of the instance instead. Flow primitives include: *AppendNode* (*n, a: Activity*) for inserting a new sub-activity named *n* in the activity graph, as a successor of sub-activity named *a*[1]; *SplitNode* (*n, a: Activity; b: Boolean*) for adding an existing task *n* among the successors of *a*, (i.e., adds a transition from *a* to *n* in the activity graph), with transition condition *b*, so that *a* becomes an OR-split node and *n* becomes an OR-join node; *JoinNode* (*a: Activity; w: set of Activity*) for makeing all activities in *w* become predecessors of *a* in AND-join mode; and *ModifyCondition*(*s, t: Activity; b: Boolean*) for changing the transition condition of the arc from *s* to *t* to *b*.

Based on these primitives, other common operators can be derived (cf. [2] for detailed algorithms). From a user's perspective, these primitives may be too low-level. Some typical workflow evolution methods for ADOME-WFMS at a higher-level are specified with the above-mentioned workflow primitives so that users can easily choose their desired workflow evolution. Furthermore, these workflow evolution actions are typical solutions for avoiding some currently anticipated exceptions.

Workflow evolution policies refer to how workflow instances adopt newly evolved workflow schema, while they are executing. Because ADOME-WFMS uses activity decomposition, upon workflow evolution (i.e., modification of a certain sub-activity class definition), the side effects of affecting other activities containing this sub-activity are very much confined. At the time of the workflow evolution, only those activities having the same sub-activity currently executing are affected. Other activities having the same sub-activity but not currently executing are unaffected since the sub-activity is encapsulated, and behaves as a black box to activities at a higher level.

To allow further control over the semantics and implications of workflow evolution, users can choose the various evolution policies, such as *temporary change* (for the current instance only), *delayed change* (other running instances continue with the old definition), *abrupt change* (abort all executing instances and restart with new definition), *immediate change* (all instances suddenly use the new definition), and *progressive change* (allow different instances to take different decisions on whether to hand over to the new definition).

For progressive change, a mechanism called hand-over policy is used. As motivated in [5] hand-over rules (in the form of ECA rules) can be used to specify how individual affected workflow instances adopt the new schema for execution. The main issue considered is the position of a running instance, i.e., the current task of that instance being executed (specified with the "ON" clause). The condition part can be any user-defined condition, including those referring to past execution history. Because ADOME-WFMS Recovery Manager (cf. [2]) keeps track of the sequence of executing tasks and sub-activities for backtracking, not only the current position but also any pattern specification on the execution history can be specified.

We consider further research issues on interfacing and interoperability important for extending the applicability of an advanced WFMS engine, including graphical workflow evolution tools, and inter-operation with other WFMS. We are interested in the application of ADOME-WFMS in various advanced real-life e-commerce environments, such as procurement, finance, stock trading and insurance. We believe such an interface environment is particularly suitable for our target of advanced e-commerce applications.

[1] The former successors of *a* will become successors of *n* in the new schema.

References

[1] F. Casati. Models, Semantics, and Formal Methods for the Design of Workflows and their Exceptions. PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy (1998)

[2] D.K.W. Chiu. Exception Handling in an Object-Oriented Workflow Management System. Ph.D. Thesis, Computer Science Dept., Hong Kong University of Science and Technology (2000)

[3] D.K.W. Chiu, Q. Li and K. Karlapalem. A Meta Modeling Approach for Workflow Management Systems Supporting Exception Handling. *Information Systems*, 24(2):159-184 (1999).

[4] Q. Li and F. H. Lochovsky. ADOME: an Advanced Object Modeling Environment. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):255-276 (1998).

[5] C. Liu, M. Orlowska, H., Li. Automating handover in dynamic workflow environments. In Proceedings of the 10th International Conference on Advanced Information Systems Engineering CAiSE'98, Pisa, Italy (1998)159-172.

Proceedings of the Seventh International Conference on Database Systems for Advanced Applications (DASFAA '01)
0-7695-0996-7/01 $10.00 © 2001 IEEE