2001

Methodological Considerations for Time Modeling inWorkflows

Olivera Marjanovic
The University of New South Wales, o.marjanovic@unsw.edu.au

Methodological Considerations for Time Modeling in Workflows

Olivera Marjanovic

School of Information Systems, Technology and Management
The University of New South Wales, Sydney, Australia
o.marjanovic@unsw.edu.au

Abstract

This paper offers a critical analysis of the popular control-flow-oriented workflow modeling paradigm and argues that time is not just an additional property specified on the top of control flows as it is commonly done in commercial and research workflow models. It also explains why this way of time modeling can easily lead to an inability to properly specify temporal constraints and verify their consistency. The paper proposes a two-level workflow modeling where the control-flow level is used for modeling of control flows, monitoring of workflow execution and reasoning about the order of tasks while the operational level is used for more precise time modelling and verification of temporal constraints.

Keywords

Conceptual modeling, workflows, time management, business processes

INTRODUCTION

Workflow technology has been widely recognized as one of the most influential business technologies next only to the Internet (Workflow Management Coalition 1999). Workflows are process-oriented business information systems that offer the right task at the right point of time to the right people along with the resources needed to perform these tasks (Dadam 2000). Workflows have been used for modeling, automation and reengineering of business processes in numerous application domains such as finance and banking, office automation, health care, software engineering and telecommunications in small, as well as large, organisational settings (Worah and Sheth, 1996).

The problem of time management has only recently attracted substantial attention in the workflow research community. This is mainly due to the fact, that in spite of the importance of time for coordination and execution of business processes, the existing time management support in commercial workflows is rather rudimentary (Dadam 2000), (Eder et al. 1999), (Geppert et al. 1998). The requirements for time modeling and visualisation far exceed capabilities provided by today's commercial workflow systems. However, if the workflow management systems are to support a broader range of business processes, they have to include better time management (Marjanovic & Orlowska 1999), (Dadam et al. 2000). At the same time, none of the currently available time modeling techniques such as those found in project management, temporal databases or artificial intelligence are suitable for, and directly applicable to, time management in workflows due to the specifics of a typical business process.

This paper analyses the popular control-flow-oriented workflow modeling paradigm from a time modeling perspective and argues that time is not just an additional property that can be added to an existing workflow model. This is because when modeling control flows, temporal properties have to be considered i.e. temporal properties can be used to determine control flows. Thus, specification of various temporal constraints "on the top" of a control-flow oriented workflow model as it is usually done in current workflow research and practice is not adequate as it can lead to inability to properly specify temporal constraints and verify their consistency both during modeling time and run-time.

The main contributions of this paper are as follows:

- Critical evaluation of the existing control-flow oriented modeling paradigm and its suitability for time modeling in production workflows

- Introduction of the two-level conceptual modeling of production workflows that comprises the high-level (or control flow oriented model) and the operational-level model. The high-level model is suitable for modeling of control flows, monitoring of workflow execution and reasoning about the order of individual tasks. The operational level model is suitable for more precise time modeling including modeling of temporal constraints and verification of their temporal consistency. Workflow models at the control-flow and operational levels, though syntactically different, are constructed to be semantically equivalent.

- Methodological considerations for time modeling in workflows
- Analysis of the possible automation of workflow modeling at the operational level

The paper is organised as follows: The next section describes the related work in the area of time management. Then a control-flow oriented workflow modeling paradigm is presented, followed by analysis of the most important reasons why this modeling paradigm is not suitable for time modeling in workflows. The paper then introduces the two-level workflow modeling and presents the most important methodological considerations that have to be taken into account for proper time modeling. Finally, the analysis of possible automation of workflow modeling at two different levels is described followed by conclusions and directions for the future work.

RELATED WORK

It is very well known that the state-of-the-art of the workflow systems is largely dictated by product vendors. Currently available commercial products offer very limited support for time representation and management. This appears mainly in the form of "calendars" and "to-do" lists that specify deadlines for various tasks and generate alarms when deadlines are going to be missed. However, just generating alarms is not sufficient for effective time management since a workflow user has no support for effective time modeling (including specification and verification of various temporal constraints) and time-related decision making including detection and possible prevention of time-related problems and their management.

At the same time, very few workflow research projects take into consideration the general problem of time specification and management. For example (Eder et al. 1999) uses an extension of the PERT diagram for time modeling in workflows. The ADEPT project (Dadam et al. 2000) investigates modeling of real-time deadline constraints and the consequences of missing deadlines in the case of structural changes of a workflow during its execution (e.g. introduction of a new task in a workflow instance). The HOST (Healthcare Open Systems and Trials) project (Haimowitz et. al. 1996) introduces a temporal reasoner that is used for time management in health-care processes.

All of these research projects provide important contributions to the time management in workflows. However, they all follow the same control-flow oriented modeling paradigm where control flows are specified first and then various temporal properties (duration, constraints) are specified on the top of them. As this paper will illustrate, this particular approach is quite limited, because when modeling control flows time has to be taken into consideration. Thus, temporal properties can determine control flows in a model.

Problems of time representation and management have been researched in other disciplines for many years. Project management is probably the most influential area, because of its well understood and accepted time representation and scheduling features such as the PERT chart, Gantt chart and CPM (Critical Path Method). Although workflows and projects share many characteristics e.g., they are time and resource constrained, planned, executed and controlled, workflows significantly differ from projects and they require "workflow-specific" time management (Marjanovic and Orlowska 1999).

Other relevant time-related research disciplines include *job-shop scheduling* (with the emphasis on finding the right schedule for completing a set of tasks while satisfying temporal and resource constraints), *artificial intelligence* (with the emphasis on time representation and reasoning), *temporal data bases* (with the emphasis on storage and retrieval of temporal data), and in *real-time software engineering* (with the emphasis on specification of properties of real-time systems). However, none of temporal models used in these disciplines is completely suitable for the time management in workflows. For more detailed analysis of the main differences see (Marjanovic and Orlowska 1999).

CONTROL-FLOW ORIENTED WORKFLOW MODELING

The purpose of this section is to describe the control flow oriented modeling paradigm in workflows and to discuss why this way of modeling may not be suitable for the modeling of time as it may lead to temporal imprecision and an inability to accurately specify and consequently verify temporal constraints.

Workflow modeling

A workflow model is a description of tasks ordering, data, time, resources and other aspects of business processes. For example, a workflow "Processing of postgraduate applications" can be implemented to process a postgraduate admission at the university. Different alternative executions within a workflow model (introduced by decision nodes) are called *process instance types* or *instance types* for short. Workflow instances are particular occurrences of the process, for example a particular student's application for admission.

Currently, the most common workflow models are organised around control flows. That means, when modeling

workflows we are guided by the order of individual tasks – e.g. we look at what comes first, second, third etc. This way of modeling may have its origin in the earlier versions of workflows where a typical business process was organised around "movement of documents" or objects and where other aspects of the model such as data flow, time and transaction properties have been ignored.

This control flow paradigm also determines our understanding of the meaning of the typical workflow structures such as: concurrent, alternative, exclusive or-join and synchronizer (see Figure 1). For example, when an alternative structure is used, it means that based on a decision made a group of instance types is split into two or more groups where each group has different execution pattern i.e. for each group a different set of tasks is executed. An *exclusive or-join* structure is used to merge different instance types just because after a certain point in workflow execution all these different instance types start to follow the same execution pattern i.e. the same tasks are executed for all instance types. In this case, we treat the subsequent tasks as the "same" for all instance types just because they are the same from the control flow perspective.

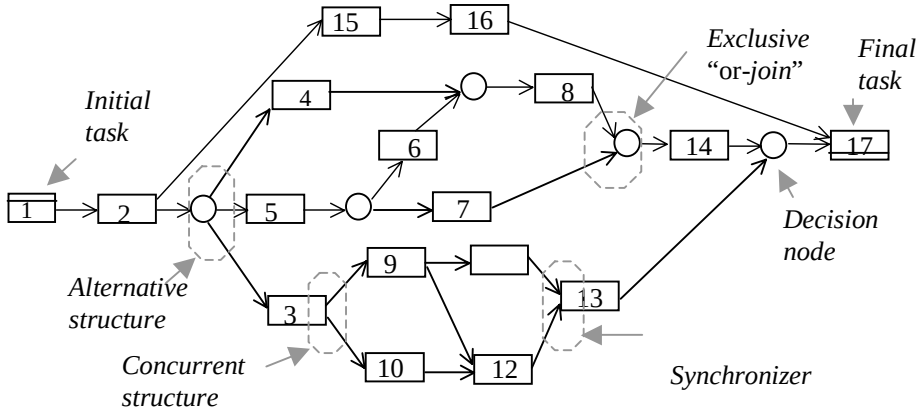

Figure 1: An example of a simple workflow graph

When a workflow is modeled from a control flow perspective, the resulting models are compact, where redundancy (i.e. repetition) of individual tasks is completely avoided. A direct consequence of this approach is that we tend to use a bottom-up approach where several instance types are "glued" or combined together just because they follow the same control flows and execute the same tasks in the same order. Control-flow oriented modeling has an obvious and indisputable advantage: workflow models are much easier to understand due to their reduced complexity. That is the main reason why control-flow oriented models are especially promoted by *Business Process Reengineering*. Furthermore, control-flow oriented workflow models are best used for monitoring purposes (to determine the current state of an execution of an instance), to reason about the order of individual tasks and to understand flows of data.

An example of a simplified, control flow oriented, workflow model of a process "Processing of a postgraduate application" is depicted in Figure 2. For the sake of simplicity it is assumed that durations of individual tasks are expressed in days (d), though different time granularity may be easily adopted.

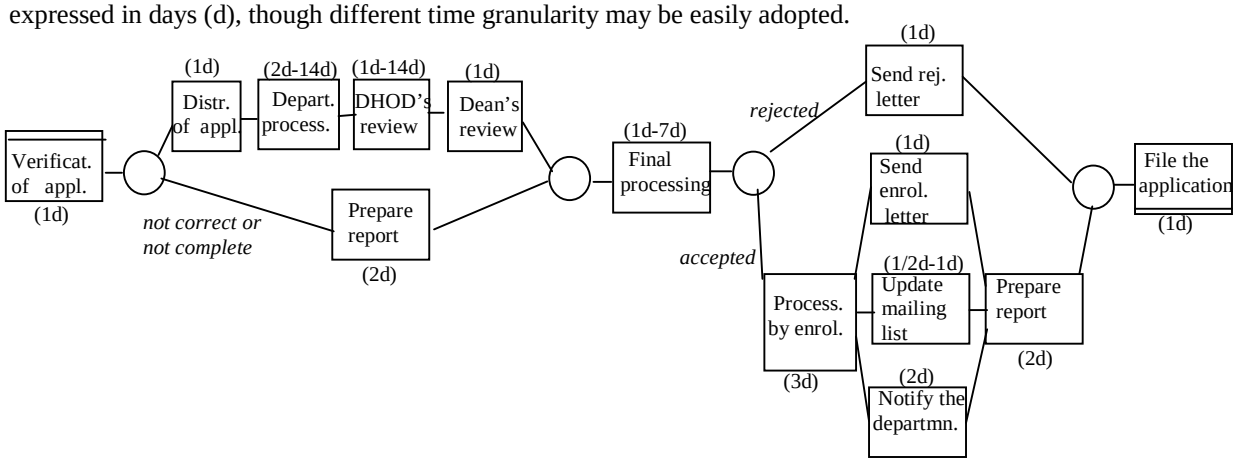

Figure 2. "Processing of a postgraduate application" – a simplified control-flow oriented workflow model

This example can be interpreted as follows. The admission process starts when the admission office receives a postgraduate application. The first task is to verify the completeness and correctness of the applications. Complete and correct applications are distributed to the respective department for processing. After departmental processing all applications are reviewed by the deputy head of department for postgraduate studies and after that

forwarded to the respective dean. After the dean reviews the application, the application is sent back to the admission office for final processing. Note that, if the application is initially incomplete or incorrect, it is sent directly to the administrative officer for final processing. If the application is accepted, it is forwarded to the enrolment section for further processing. Three different tasks are then initiated concurrently. An enrolment letter is sent to the applicant, the applicant's details are put on the enrollment mailing list, and the department is notified again. After all three tasks are completed, the enrollment officer signs the report, and forwards the application to the central registry. If the application is rejected, the rejection letter is sent to the applicant and the application is forwarded to the central registry. Both successful and unsuccessful applications are filed in the central registry.

Hence, according to the control-flow modeling paradigm, even if a task belongs to several different instance types at the same time, we tend to assume that it is always the same for every instance type if it is always preceded or succeeded by the same tasks. Again, to determine whether a task belongs to one or more instance types at the same time, we look mainly at the control flows.

Modeling of time in control-flow oriented workflow model

From the business perspective, temporal constraints are defined by laws and regulations, business policies or formal corporate rules, common practices as well as mutual agreements and expectations related to efficiency/productivity of business practice. A workflow model must capture these constraints if they are to be enforced in the process execution. Specification of temporal constraints may involve both *relative* and *absolute* time. For example, to model the expected duration of individual tasks or instance types, relative time values of a certain granularity are used, e.g. two hours, 10 minutes. During workflow execution (run-time) absolute (or real) time values of certain granularity are used, e.g. a task started at 2.00 p.m. and finished at 3.00 p.m. on the 1st of July 2001.

The following are the three main temporal constraints used in workflow modeling:

- A *duration* constraint, which is used for modeling the expected duration of a workflow task. This is usually specified either precisely by using a single relative time value as an interval of two relative time values (e.g. task takes between 20 and 30 minutes) representing its expected minimum and maximum duration (e.g. a task takes 1 hour).

- A *deadline* constraint, which can be specified in terms of absolute time limits when a task should start or end during workflow execution (e.g. the deadline for a grant application is 23. Jun 2001 at 5 p.m.).

- Finally, an *interdependent* temporal constraint limits when a task should start/finish relative to the start/finish of another task. The time distance can be represented as a relative time value. For example: "a project proposal must be ready for review no later than 7 days before the committee meeting.

Another important concept is that of *temporal consistency*. A temporal constraint is consistent with a given workflow model if and only if it can be satisfied based on syntax of the workflow model and expected minimum and maximum duration of relevant workflow tasks (Marjanovic and Orlowska, 1999). It is important to point out that consistency of temporal constraints has to be verified several times during the workflow lifetime: initially during workflow modeling (at the built time) and then again during execution (at the run time) and at several control points (usually after each decision node) to make sure that tasks are executing as planned.

When time is modeled in a typical "control-flow" oriented workflow model, the common procedure is to determine control-flows first and then to add time "on the top" of such model i.e. to assign temporal attributes such as duration or beginning and end time to individual tasks which order of execution has been predetermined by control flows. Majority of currently available workflow models that incorporate temporal aspects follow that modeling paradigm. Examples can be found in our own work (see Marjanovic and Orlowska, 1999) as well as in other workflow research project such as (Dadam et al. 2000), (Eder et al. 1999), (Haimowitz 1996).

After analysis of a number of workflow models and business processes, it is possible to observe, that if a workflow model is designed mainly on the bases of the control flows then modeling of time in such a model for some workflow tasks can be rather imprecise. In the previous example (depicted by Figure 2), one can observe that the task "Verification for completeness and correctness" takes 2 hours while DHOD's review can take between ½ hour and 2 weeks. Unfortunately, due to the nature of this task, it is not possible to be more precise during design of a control-flow oriented model. On the other hand, precision during workflow modeling is very important due to the fact that violation of some temporal constraints can be detected even before workflow start executing. In order words, imprecise information is likely to limit the accuracy of verification of temporal constraints and prediction of possible violation.

Based on our experience, it was possible to observe that large (i.e. imprecise) intervals usually correspond to

tasks that belong to more than one instance type and majority of these tasks follow the exclusive "or-join" structures. For example, task "DHOD's review" is such an imprecise task because specific PhD applications may take longer to review (e.g. international versus local applications). Both international and local applications follow the same execution pattern and in the majority of the cases they are processed in the same way. Furthermore, some international applications may take longer to review and so forth. Hence, by assigning the same duration interval to all different cases to describe the task "DHOD's review", the resulting interval ends up being quite large. Similarly, the task "Final processing" takes longer to execute for course work than research postgraduate applications and so forth. Additional problems occur when we try to specify different temporal constraints for different instance types that follow the same execution pattern. For example we may want to specify different deadline constraints for different types of postgraduate applications e.g. the deadline for international students is 10 of October while the deadline for local applicants is 31 of October 2001.

Hence, the main problem with time modeling in control-flow oriented workflow models is that temporal properties of different instance types can be quite different (e.g. different duration of individual tasks or different temporal constraints) however, these instance types are "glued" together just because they follow the same execution pattern.

One could argue that, a possible solution would be to keep the control flow oriented model but to assign two or more different duration intervals to a single task that have different temporal properties for different instance types. However, in addition to increased complexity, the problem is also that in some cases, based only on the control flows different instance types cannot be determined before that a task is completed. See for example a model depicted in Figure 3. Though task 4 belongs to four different instance types, type of the workflow instance that is currently in progress, can be determined only after task 4 is completed. Therefore, it would be incorrect to assign four different duration intervals to task 4 as we cannot reason about them during run-time until task 4 is completed.

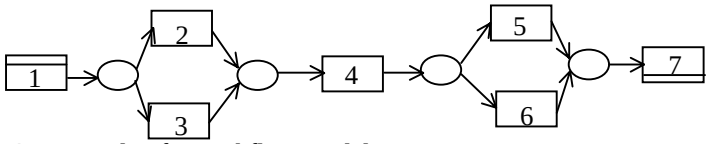

Figure 3: An example of a workflow model

Another possible solution to this problem would be to take the "bottom-up" approach and "unglue" all instance types that have been put together by control-flow oriented modeling approach and model them as a separate execution paths in a workflow model. Again, this solution has a number of problems. The resulting model is very complex, especially for the workflow models with large number of alternative and exclusive "or-join" structures. Even though some tasks are exactly the same for different instance types they are represented a number of times in the same model. Another, even more serious problem is that during workflow modeling it is impossible to predict all different cases of a single task in advance (e.g. all possible cases of PhD applications).

So when considering options for workflow modeling, on one side we have a case of simple, non-redundant control-flow oriented workflow model and on the other hand, a workflow model with a large number of workflow instance types modeled separately and therefore a large number of redundant tasks. However, none of these two approaches is suitable for proper time modeling in workflows, as the following two problems have to be solved. Firstly, it is necessary to determine tasks in the control-flow oriented model that may behave differently for different instance types as well as tasks that behave the same way for all instance types they belong to. Another problem is how to design a workflow model where design of temporal constraints is not guided by control flows.

This paper proposes the third approach i.e. to use a two-level workflow model. A high-level control flow oriented workflow model is used for monitoring, reasoning about control flows and task dependencies while an operation-level workflow model is used to distinguish tasks that behave differently for different instance types and for the modeling of time. The following two sections describe these models in more details.

TWO-LEVELS OF WORKFLOW MODELING

Identification of imprecise tasks

Suppose that the *expected* duration of a workflow task i, d(i) is represented as:

$$d(i) = [m(i), M(i)]$$

where *i* is an unique identifier of a task, and m(i) and M(i) are two relative time values representing respectively the expected minimum and maximum duration of a task i.

The same task usually belongs to many different instances i.e. it is executed a number of times, each time with different beginning and end time. Suppose that *Inst* represents a set of all workflow instance identifiers. We define execution of task i for a particular instance $\Im k$ as follows:

$$E(i,\Im k) = [b(i)_k, e(i)_k] \text{ where } \Im k \in Inst \text{ and } i \in N$$

$b(i)_k$ and $e(i)_k$ represent two absolute time values which correspond to beginning and the end of task i, respectively and *N* is a set of workflow tasks.

The real duration of task i for given instance $\Im k$ can be calculated only after task's completion as follows:

$$r(i)_k = dist\ (b(i)_k, e(i)_k)$$

where function *dist* returns the relative time value that corresponds to the time distance between $b(i)_k$ and $e(i)_k$.

Workflow log is used to store various data on workflow execution. However for the purpose of time modeling we are interested in the absolute times when a task starts and finishes executing for all workflow instances that task belongs to. Therefore, the accumulated experience on execution of task i can be represented as:

$$Exp(i) = \{E(i,\Im_1), E(i,\Im_2), \ldots, E(i,\Im_m)\} \text{ where } \Im_1,\Im_2,\ldots,\Im_m \in Inst \text{ and } i \in N$$

Based on the absolute times that correspond to the beginning/end of task i, it is possible to analyse the real duration of a task as a function of all workflow instances it belongs to.

For example, if the real duration of task i is in most cases outside of the expected duration interval (e.g. in most cases longer than the maximum expected duration), refinement of the original workflow model is required. Note that the problem of determining the most appropriate duration is domain-dependent and cannot be generalised. For example for more time-critical systems the requirement could be that more than 95% of all tasks belong to the estimated domain while in some other cases if 80% of all tasks belong to the interval {m(i), M(i)) the expected duration is still acceptable.

We are particularly interested to analyse a workflow log to detect cases where duration of a task is a function of an instance type. Figure 4 depicts such a case where ti and tj are two instance types and time line R is used to represent real duration (depicted as "dot") versus expected duration (depicted as two values m(i) and M(i)). Thus task i takes longer to execute for instance type t_j than for instance type t_k.

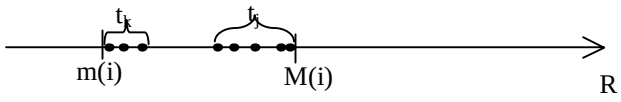

Figure 5: Real duration of task i as function of instance types

Note that in many cases this situation cannot be detected in advance during workflow modeling but only after analysis of the accumulated experience stored in a workflow log.

So in this case we can model the expected duration of task i as:

Although it is clear that task duration is a function of instance types t_k and t_j, in a control-flow oriented workflow

$$d(i) = \begin{cases} [m'(i), M'(i)] & for \quad i \in t_k \\ [m''(i), M''(i)] & for \quad i \in t_j \end{cases} \quad where \quad m(i) \le m'(i) \le M'(i) \le m''(i) \le M''(i) \le M(i)$$

model these two instance types are "glued" together causing temporal imprecision as two different duration intervals cannot be assigned to task i.

Recall that not every task that belongs to two or more instance types has duration that is the function of an instance type i.e. it may take approximately the same time to execute for different cases. So in this case workflow instances can remain "glued" together.

When workflow experience is analysed and all such tasks are detected and their different "cases" are identified, the next step is to design a model at the operational level.

Workflow modeling at the operational level

To accommodate modeling of imprecise tasks and still preserve simplicity and compactness of the control-flow oriented workflow model – we propose the introduction of the second level workflow model; the so called operational level workflow. This level takes into account not only control flows but also temporal properties of

individual tasks and temporal constraints. The underlying assumption for construction of this model is that temporal properties will determine modeling of control flows at this level.

To illustrate workflow modeling at the operational level we use the same example of the workflow "Postgraduate Application Processing". Recall that the control-flow model of this business process was depicted by Figure 2. Figure 7 illustrates the same workflow at the operational level.

Note that the task "Departmental processing" when represented at the control-flow level takes between 2 and 14 days. Hence, the associated duration interval is rather imprecise. The same duration constraint when represented at the operational level is more precise: departmental processing of course-work postgraduate applications takes 2 days while research postgraduate applications takes 21 days. Also the task representing the review of an application by the deputy head of department "DHOD's review" when specified at the control level has rather long duration interval – it takes between 1 and 14 days. When specified at the operational level it is more precise: DHOD's review for course work applications takes between 1 and 2 days, DHOD's review for local research applications takes 5 to 7 days while review of international postgraduate applications takes 14 days. Final processing at the control-flow level takes 1 to 7 days while at the operational level it takes 7 days for correct and complete applications while processing of incorrect and/or incomplete applications, takes 1 day.

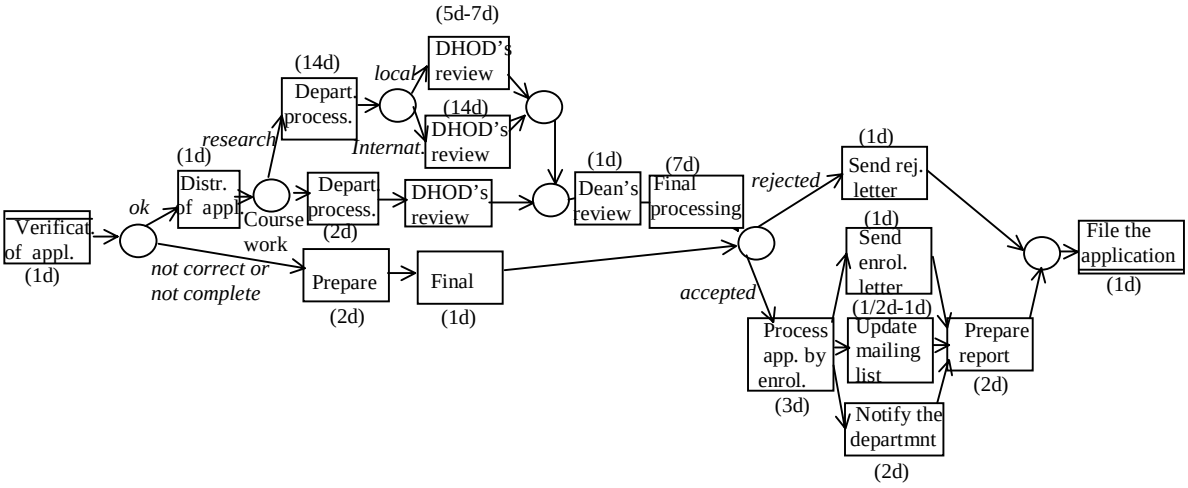

Figure 6: An operational level workflow model of "Postgraduate Applications Processing"

METHODOLOGICAL CONSIDERATIONS FOR TIME MODELING IN PRODUCTION WORKFLOWS

Construction of the operational-level workflow model is not an automatic process. After the experience on workflow execution is accumulated and analysed as described in the previous section, a workflow analyst has to decide how to model workflow at the operational level. Note that in this paper, we limit our consideration to the temporal aspect of a workflow model. The following modeling recommendations should be taken into account:

- For workflow tasks that belong to more than one instance type, if temporal properties of a task cannot be distinguished for different instance types (e.g. a task has the same complexity for different instance types) then we use a single task with a single duration interval. In other words, if task is always of the same complexity no matter what instance type it belongs to, this task will be the same at both modeling levels.

- Two or more instance types can be joined together by exclusive or-join structure at the operational level *only* if the tasks that follow the decision node have the same complexity (duration) for all individual instance types.

- If a task has different duration for different instance types then it is necessary to distinguish these different cases at the operational level by introducing decision nodes.

- Consistency of the control flows has to be preserved at the operational level. That means that all decision nodes are exclusive (including the newly introduced decision nodes that exist only at the operational level), and there are no deadlocks with synchronization constructs.

- The order of the individual tasks determined at the control flow level must be preserved at the operational level. In other words if task i is executed before task j at the control level, then at the operational level task i will be executed before all cases of task j (e.g. j' and j"). That is why we say that workflow models at two different levels are syntactically different but semantically equivalent.

- When an interdependent temporal constraint is specified at the control-level for a task then that constraint applies to all instance types that the task belongs to. If different temporal constraints apply to different instance types that the task belongs to, then they should be modeled at the operational level. For example, suppose that at the control-flow level an interdependent temporal constraint defines that task j must start no later than d time after task i is completed (as depicted by Figure 7a). The same temporal constraint can be represented at the operational level in a number of ways (as depicted by Figure 7 b, c and d) where j' and j" represent different cases of task j.

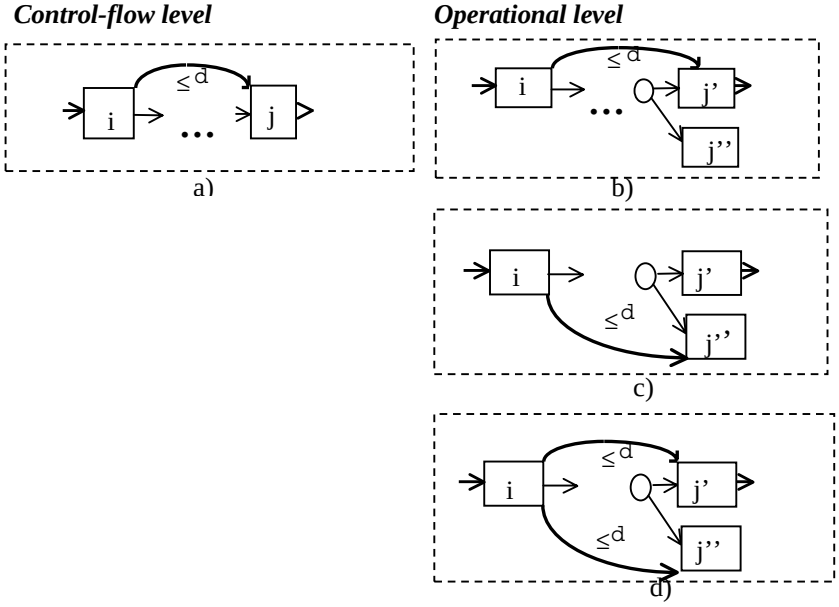

Figure 7: An interdependent temporal constraint at the control-flow and operational levels

Therefore, it is possible to specify an interdependent temporal constraint only for some type instances or for all instance types to which task j belongs. In this way it is possible to detect cases where the same temporal constraint is consistent for some instance types and inconsistent for others. Furthermore, it is also possible to define different interdependent temporal constraints for different instance types as depicted in Figure 8 and verify them independently at the operational level. Note that this particular case cannot be represented at the control-flow level.

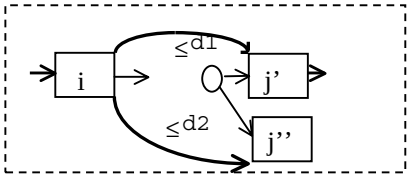

Figure 8: Different interdependent constraints defined for different instance types at the operational level

- A deadline constraint defined for task i at the control-level such that task i must be completed by *Date1* can be represented in three different ways at the operational level (as depicted by Figure 9).

In this way, at the operational level, it is possible to assign deadline constraints to individual instance types rather than all instance types to which a task belongs. For example, it is possible to specify a deadline for international PhD students rather than all postgraduate students. Furthermore, it is possible to verify the temporal consistency of constraints for an individual instance type rather than all instance types to which a task belongs. In this way, precision is improved because it is possible that a temporal constraint is consistent for some but not for all instance types. If the same constraint is verified at the control-flow level it will appear as inconsistent for all instance types.

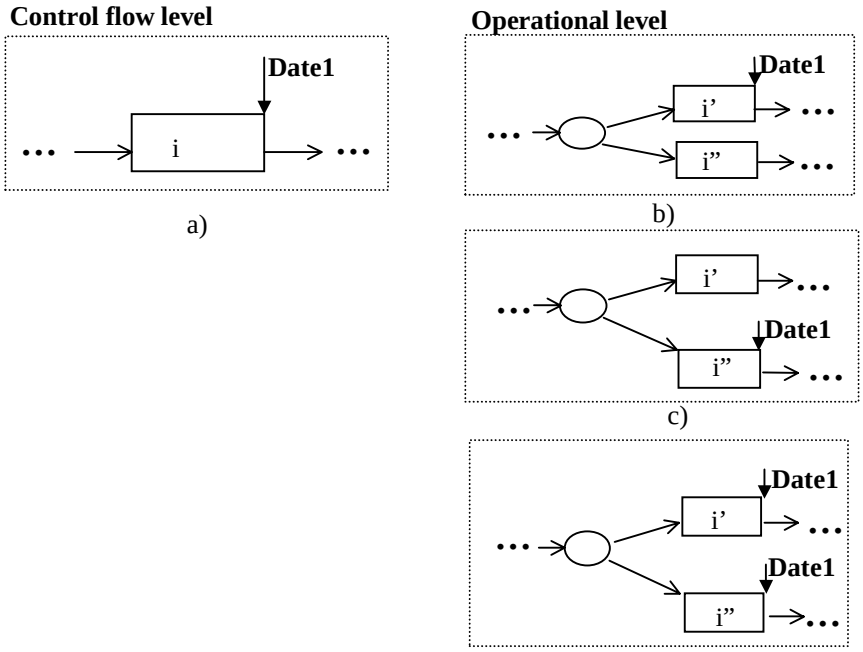

Figure 9: Modeling of deadline constraint at the control flow and operational levels

Similarly, it is possible to specify different deadline constraints at the operational level for different instance types to which task i belongs (as depicted by Figure 10). Note that this is not possible at the control-flow level.

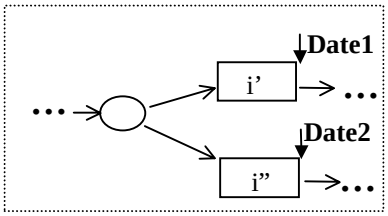

Figure 10: Specification of different deadline constraints for different instance types at the operational level

- Verification of the temporal constraints should be performed at the operational level. This makes the verification process much more accurate, as it is possible to detect cases when the temporal constraint is consistent for some instance types but not for others. Note that if the same temporal constraint was represented and verified only at the control-flow level it would be reported as inconsistent for all instance types. The algorithm for verification of temporal consistency in workflows as introduced in (Marjanovic and Orlowska, 1999a) is still applicable to the model at the operational level (it is out of the scope of this paper). Due to the improved precision in specification, the accuracy of verification will be improved. During workflow execution, it is possible to predict more accurately when a specific task is likely to occur as well as to dynamically verify temporal constraints based on the actual execution (i.e. the real duration) of individual tasks.

AUTOMATION OF TWO-LEVEL WORKFLOW MODELING

As we already pointed out, design of the workflow model at the operational level is complex. However, automatic support can be provided to some extent for some aspects of modeling. Based on the theoretical concepts introduced in this paper the following design activities can be effectively supported by technology:

- Identification of imprecise tasks that require remodeling or refinement.

- Analysis of temporal patterns for individual tasks (e.g. whether task duration has changed over time) and comparison of temporal patterns for different workflow tasks

- Analysis of the real duration for imprecise tasks to determine the most appropriate expected duration. This may include simulation as well as automatic verification of temporal consistency to determine whether the chosen duration is consistent with all temporal constraints.

- For imprecise tasks, analysis of task dependency from the instance type

- Construction of the operational-level workflow model – including modeling of temporal constraints and

verification of temporal consistency as well as verification of control flows.

Although the construction of the operational-level model cannot be fully automated it is necessary to find the right balance between precision of the operational level model and its increased complexity. This is an iterative process where operational level workflow model including constraint specification is constantly refined based on the accumulated experience.

CONCLUSIONS

In spite of the importance of time for coordination and execution of business processes, the existing time management support in workflows is rather rudimentary. The requirements for time modeling and visualisation far exceed the capabilities provided by today's commercial workflow systems.

The paper critically analyses the popular control-flow-oriented workflow modeling paradigm and argues that specification of the temporal constraints "on the top" of such a model is not adequate as it can lead to inability to specify and verify temporal constraints. The paper proposes workflow modeling at two different levels and illustrates how, on the basis of the accumulated experience, the models at these two conceptual levels can be designed to be syntactically different but semantically equivalent. Finally, the paper offers some recommendations for design of the workflow model at the operational level and in particular, its temporal aspect.

The main conclusion of this work is that time is not just an additional property that is added on top of the control flows. Therefore, in order to specify control flows it is necessary to take into consideration the temporal aspect. This is an important conclusion that could be even extended to other process-oriented models in information systems (such as for example Petri-nets).

Although this paper is limited to the temporal aspect of the workflow model, the operational level can be also used for more precise modeling of data flows, resources as well as for more precise specification of compensation techniques used in dynamic workflows. In all these cases, the operational level workflow model is likely to be further refined. Our future work in this field includes investigation of the temporal aspect of interorganisational workflows used in e-commerce applications.

REFERENCES

Dadam, P. at all. (2000), *ADEPT- Next Generation Workflow Management System*, ADEPT Project Dept. DBIS, (http://www.informatik.uni-ulm.de/dbis/F%261/forschung/workflow/ftext-adept_e.html)

Eder J., Panagos E., Pozewaunig H. and Rabinovich M. (1999), "Time Management in Workflow Systems" in: W. Abramowicz, M.E. Orlowska (eds.), *BIS'99 3rd International Conference on Business Information Systems*, Springer Verlag London Berlin, Heidelberg,1999, pp 265-280.

Geppert, A., Kradolfer, M. and Tombros, D. (1998), "Market-Based Workflow Management", in Lamersdof and Merz (eds.), *Trends in Distributed Systems for Electronic Commerce*, Proceedings of the International IFIP/G1 Working Conference, TREC'98, Hamburg, Germany, June.

Haimowitz, I, Farley, J, Fields, G.S, Stillman, J. and Vaiver, B (1996), "Temporal Reasoning for Automated Workflow in Health Care Enterprises", in Adam and Yesha (eds.) *Electronic Commerce: Current Research Issues and Applications*, Lecture Notes in Computer Science, no. 1028, Springer-Verlag.

Marjanovic, O. and Orlowska, M.E. (1999) "On modeling and verification of temporal constraints in production workflows", *Knowledge and Information Systems*, Vol. 1, No. 2. Springer-Verlag.

Worah, D. and Sheth, A., (1996), "What do Advanced Transaction Models have to offer for Workflows?", *Proceedings of the International Workshop on Advanced Transaction Models and Architectures*, Goa, India.

Workflow Management Coalition (1999), *The Workflow Management Coalition Specifications - Terminology and Glossary*.

COPYRIGHT