Predictive Workflow Management

Euthimios Panagos and **Michael Rabinovich**
AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 07932
{thimios, misha}@research.att.com

Abstract

In this paper, we propose a new area for research in workflow management systems: the use of predictions in managing workflows. Predictions can be beneficial in making various decisions during workflow management such as: when to escalate? How much execution time to allow for each process step? What is a reasonable time for the whole process to complete? These questions are currently addressed by the business analyst in a static way for all process executions. We argue that by using predictions, workflow management systems can provide dynamic answers to these questions based on the current system state and past execution history. This should result in reduced operational costs and better system utilization, among other benefits.

Keywords: Deadlines, Escalations, Escalation Cost, Prediction, Workflow, Workflow Management.

1 Introduction

Today, organizations use workflow management systems (WFMSs) to streamline, automate, and manage business processes that depend on information systems and human resources (e.g., provisioning telephone services, processing insurance claims, and handling bank loan applications). WFMSs provide tools to support the modeling of business processes at a conceptual level, coordinate the execution of the component activities according to the model, monitor the execution progress, and report various statistics about the business processes and the resources involved in their enactment [2, 5].

A workflow is an abstraction of a business process. It consists of *activities*, which correspond to individual process steps, and *agents*, which execute these activities. An agent may be an information system (e.g., a database system), a human (e.g., a customer representative) or a combination of both (e.g., a human using a software program). The workflow specification may also specify execution durations, referred to as *deadlines* here, for both the activities and the process itself. When an activity misses its deadline, special actions, referred to as *escalations*, may be triggered. Escalations increase the operational cost of a business process since, otherwise, they would have been specified as part of the normal execution path.

In this paper, we propose the exploitation of state information in WFMSs to predict the timing properties of workflow executions, and to use these predictions to manage workflows better. We also outline two specific techniques we have started working on. These techniques use predictions to improve the expressiveness of the workflow model as well as reduce the operational costs of running workflow instances that result in escalations.

To motivate the model enrichment, consider the telephone service provisioning workflow in a telecommunications company. Currently, a business analyst statically decides the maximum execution time for all instances of the provisioning workflow based on the worst-case scenario for the load level of the resources involved in this workflow. Therefore, when a customer orders a service, the customer representative cites this execution time as the deadline. However, if the customer called during a quiet period, when the system is not busy, the company could have a competitive advantage by promising a shorter deadline. On the other hand, if the system is swamped with service orders (e.g., in a better than expected response to a promotion or a new service offer as it happened to AT&T WorldNet introduction), it would be better to promise a slightly longer execution time than to give an unrealistic promise that will most certainly result in an escalation.

Regarding the reduction of operational costs, prediction can be used to estimate the likelihood that active workflow process instances will escalate due to conditions at the agents that will execute some of the

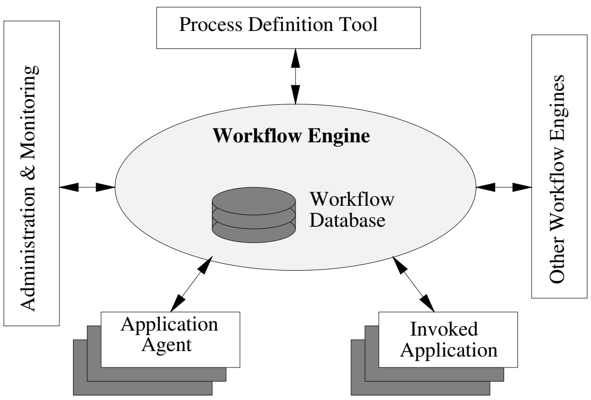

Figure 1: WfMC: reference architecture

process' activities *in the future*. Depending on this likelihood, it may be beneficial to escalate during the execution of an *earlier* activity, which has not missed its deadline yet. Indeed, an earlier escalation leaves more time to remedy the situation, which often translates into reduced operational costs. Also, in many cases an escalation involves rolling back finished activities and, hence, invoking it early entails less work and fewer wasted resources.

There are many other ways in which using prediction can be beneficial, such as scheduling of activities among agents or manipulating activity deadlines to reduce the number of instances resulting in escalations (we outlined some preliminary ideas with respect to the latter in [7]). Our work complements the work of [8] that describes challenges raised by applying workflow management technology in information systems.

The remainder of the paper is organized as follows. Section 2 offers an introduction to workflow concepts. Section 3 describes how predictions can be used for assigning completion times to workflow instances. Section 4 presents our basic algorithm for invoking early escalations and, finally, Section 5 concludes our presentation.

2 Current State of WFMSs

According to the Workflow Management Coalition (WfMC) reference model [3], shown in Figure 1, a WFMS consists of an engine, application agents, invoked applications, a process definition tool, and administration and monitoring tools. The process definition tool is a visual editor which is used to define the specification, i.e., the *schema*, of a workflow process. The same schema can be used for creating multiple *instances* of the same business process at a later time. The workflow engine and the tools communicate with a workflow database to store and update workflow-relevant data, such as workflow schemas, statistical information, and control information required to execute and monitor the active process instances.

The schema specifies the activities that constitute the workflow and precedence relationships among them. Precedence relationships determine the execution sequence of the activities as well as the data flow between them. Activities can be executed sequentially or in parallel. Parallel executions may be unconditional, i.e., all activities are executed, or conditional, i.e., only activities that satisfy a condition are executed. In addition, activities may be executed repeatedly and the number of iterations may be determined during the execution.

Existing WFMSs allow the specification of completion times, i.e., deadlines, for workflow activities and the entire workflow process [6, 4, 1, 9]. For the entire process, the deadline is specified as the allowable execution time. When a process instance is initiated, this deadline is translated into an absolute time based on the starting time of the instance. For an individual activity, the deadline may be expressed as the allowable execution time, or the time (relative to the starting time of the process instance) by which the activity must complete. For instance, an activity may be required to take no more than two hours, or to complete within six hours since the start of the process instance. Finally, the schema may specify time-triggered activities, *escalations*, which are executed when the process or some of its activities miss their deadlines.

Existing WFMSs maintain audit logs that keep track of information about the status of the various system components, changes to the status of workflow processes, and various statistics about past process executions. This information can be used to provide real-time status reports about the state of the system and the active workflow process instances, as well as various statistical measurements such as the average completion time of an activity belonging to the particular process schema. In the next sections, we discuss how these statistical measurements can be used for predicting the timing characteristics of the active workflow instances.

3 Predictive Deadline Assignment

In existing WFMSs, expected completion times (i.e., deadlines) are assigned during schema specification. Therefore, all process instances derived from the same process schema have the same deadlines. This approach is very static and may result in an increased cost for an organization. For instance, consider the telephone service provisioning process for a new customer. If all instances of this process are assigned a deadline of one week, a customer will be dissatisfied if she does not have her phone working within a week, and the telephone company will have to offer compensation to keep the customer.

A better way of handling the above problem is to *predict* the completion time of a process, when the process is submitted for execution, by examining the current state of the system (load, resource availability, etc.) and using the statistics collected during the past executions of instances having the same schema. Then, deadlines can be assigned to both the process instance and its activities using the predicted completion time. Different techniques can be used to predict the completion time, varying in accuracy and the amount of information they require. As a starting point, we propose a simple procedure and follow it with a discussion of some issues that must be resolved on the way to more accurate techniques.

Let W be a workflow instance containing distinct activities $T_1, \ldots, T_m$, and $avg\_completion(T_i)$ be an estimate of the average completion time for T_i. This estimate corresponds to the time it takes the agent that will execute T_i to complete T_i, after T_i is submitted for execution, and it includes any queuing time at the agent's work queue – such estimates can be easily extracted from the audit logs maintained by existing WFMSs. Assigning deadlines to the process and the individual activities consists of the following steps.

1. For each T_i, let its predicted completion time, $pred\_completion(T_i)$, be $a_{T_i} * avg\_completion(T_i)$, where a_{T_i} is the *load factor* that depends on the current load of the system. In the simplest way, a_{T_i} can be calculated as the ratio $\frac{cur\_queue(T_i)}{avg\_queue(T_i)}$, where $cur\_queue(T_i)$ is the current queue length of the agents that can execute T_i averaged over all such agents, and $avg\_queue(T_i)$ is the average queue length of these agents. Intuitively, if the average queue length of all agents that can execute T_i is 3 and the current queue length of these agents is 6, we should expect that T_i will take twice as long, provided that the current load conditions remain the same by the time T_i becomes ready for execution.

2. For every activity T_i, let $deadline(T_i) = s * pred\_completion(T_i)$, where $s > 1$ is a statically defined system parameter that adds some slack to the predicted completion time.

3. Find the critical path in the workflow using the above computed deadlines. When the workflow schema contains loops, roll them out assuming, e.g., the average number of loop iterations that occurred in past executions of instances of the same workflow.

4. Use the time of the critical path as the overall process deadline, and $deadline(T_i)$ as the deadline for an individual activity T_i (recall that this time has some slack built into it).

This procedure assumes that the WFMS keeps statistics about the average number of loop iterations in the past process executions for each workflow schema. We argue that the business analyst herself, when deciding statically the process deadline in today's WFMSs, must in effect roll-out loops by assuming some number of iterations. She either uses her judgement regarding an expected number of iterations, or relies on some semantic knowledge of the domain to come up with this number.

In addition, in the case where conditional executions are part of the workflow schema, the above algorithm always takes into account the longest branch. Again, using statistics about the frequency of past branch executions could result in a more accurate prediction. For instance, if branch A takes a month (a new cable must be installed to provision the order), compared to a couple of days taken by branch B, but A happens only 1% of the time, we may want to use a shorter branch in our calculations, even though the critical path goes through branch A. Also, the process definition tool must allow the business analyst to specify the branches that have to be taken into account using semantic knowledge about the process.

4 Early Escalations

When workflow activities take longer to finish than their allowable execution times (i.e., they miss their deadlines), escalations may take place. Escalations can have either local or global scope. Local escalations affect the triggering activity (i.e., the one that triggered the escalation), and they may: (a) restart the triggering activity, (b) execute a new activity and resume the triggering activity, or (c) replace the triggering activity with a new activity. Global escalations affect the whole business process, and they may: (d) abort the whole business process and compensate all finished activities, or (e) stop the normal flow of execution, compensate some of the finished activities, and execute an alternative sequence of activities to complete the process.

In all cases, escalations increase the cost of business processes and, thus, it is desirable to reduce their occurrences. Alas, escalations cannot be always avoided e.g., when resources required for the enactment of an activity are not available for an extended time period, or when the system is heavily loaded. In such cases, if we could *predict* that an escalation is likely to occur in the future, we could use this information to reduce the cost associated with the escalation. Specifically, consider a workflow with global escalations. If at some point during the workflow execution we determine that a future activity is going to escalate, it is often cost-effective to invoke escalation *immediately*, regardless of whether or not the currently executed activity misses its deadline. Indeed, this would allow more time for remedial actions and would reduce the cost associated with the execution and then compensation of finished activities.

Based on the above observation, the early escalations mechanism attempts to predict whether a process is going to escalate at some future point and, then, decide whether and when to force escalations before they actually occur. However, deciding whether to escalate early poses several problems: how many of the remaining activities should we consider? How do we decide that a future activity will miss its deadline? How do we determine that it is more cost effective to escalate early? The more activities we consider the more benefits we may reap, but the more inaccurate our predictions can be, and, thus, the greater the risk of a wrong decision. Further, the greater the cost difference between an early and future escalation, the less confident we have to be in predicting future escalation to justify escalating early.

In the following, we assume that the business analyst estimates, for each activity T_i, the cost of escalation when it is invoked during T_i's execution, denoted $escal\_cost(T_i)$ below. Let $avg\_completion(T_i)$ be T_i's average completion time (including queuing time) from the past executions and $deadline(T_i)$ be the allowable execution time of T_i. If we examine k future activities to determine whether early escalation is beneficial, the following algorithm can be used.

1. Let T be the activity to be executed next, and $T_1 \ldots T_k$ be the future k activities we are going to examine.
2. Let a_{T_i} be the load factor described in Section 3.
3. For every T_i, let $pred\_completion(T_i) = a_{T_i} * avg\_completion(T_i)$.
4. If there exists T_i such that $deadline(T_i) < pred\_completion(T_i)$ then early escalation is beneficial if:

$$[1 - \frac{escal\_cost(T)}{escal\_cost(T_i)}] * pred\_completion(T_i) >$$

$$deadline(T_i)$$

The last inequality reflects the idea that the less difference in escalation costs the more confident we want to be that T_i will miss its deadline before invoking an early escalation. The confidence in our prediction is indirectly measured by how much the predicted completion time of T_i exceeds its deadline. For example, if $pred\_completion(T_i)$ exceeds $deadline(T_i)$ slightly, then even a small error in estimating $pred\_completion(T_i)$ will lead to the incorrect prediction that T_i will escalate. Then, if the escalation costs of T and T_i are close, we may not want to escalate early, to avoid an unnecessary escalation. However, if the escalation cost of T_i greatly exceeds that of T, we would want to escalate immediately because we have a reason to believe that T_i will escalate.

The above algorithm will work fine as long as the activities we examine are executed unconditionally. However, when some of these activities are executed conditionally, then an activity with high escalation cost that is executed rarely will force us to escalate early all the time, as in the example at the end of Section 3.

This problem can be rectified in several ways. First, we can limit the scope of our prediction, k, by the next closest branching point in the workflow schema. Second, we can invoke early escalation only if is it beneficial for *all* alternative activities. This, however, introduces the following problem: a rarely executed alternative activity, which does not warrant escalation, will always block early escalations. As a middle ground between invoking escalation when it is beneficial for any one of the alternative activities and doing so only if it is beneficial for all alternative activities, we can use conditional probabilities of activity executions. These probabilities can be extracted from the audit log that records the history of past executions. In particular, if $frequency(T_i)$ is the frequency of execution of a conditional activity T_i given the preceding branching point is reached, we can modify the last inequality of the above algorithm as follows:

$$[1 - \frac{escal\_cost(T)}{frequency(T_i) * escal\_cost(T_i)}] *$$

$$pred\_completion(T_i) > deadline(T_i)$$

5 Conclusions

Using predictions in workflow management is a new research area with high potential value. Predictions can be beneficial in making various decisions during workflow management such as: when to escalate? How much execution time to allow for each process step? What is a reasonable time for the whole process to complete? In this paper, we have presented our preliminary ideas on how to answer the above questions using current system state information and the audit logs maintained by existing WFMSs.

However, many issues remain to be solved. In particular, computing the values of a_{T_i} and $frequency(T_i)$ require further investigation. In addition, dynamic modifications of workflow schemas have to be taken into account when predicting the entire completion time of a workflow instance. Finally, runtime determination of the number of iterations in a repeated execution of activities as well as the conditional execution of some activities may affect the accuracy of the predictions.

References

[1] TeamWare Flow. Collaborative workflow system for the way people work. P.O. Box 780, FIN-00101, Helsinki, Finland.

[2] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[3] The Workflow Management Coalition Group. A workflow management coalition specification. Technical report, The Workflow Management Coalition, November 1994. http://www.aiai.ed.ac.uk/WfMC/.

[4] InConcert. Technical product overview. XSoft, a division of xerox. 3400 Hillview Avenue, Palo Alto, CA 94304. http://www.xsoft.com.

[5] N. Krishnakumar and A. Sheth. Managing heterogeneous multi-system tasks to support enterprise-wide operations. *Distributed and Parallel Databases*, 3(2):1–33, 1995.

[6] F. Leymann and D. Roller. Business process management with flowmark. In *Proceedings of the 39th IEEE Computer Society International Conference*, pages 230–233, San Francisco, California, February 1994. http://www.software.ibm.com/workgroup.

[7] E. Panagos and M. Rabinovich. Escalations in workflow management systems. In *DART Workshop*, Rockville, Maryland, November 1996.

[8] A. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, and A. Wolf. Report from the NSF workshop on workflow and process automation in information systems. *SIGMOD Record*, 25(4):55–67, December 1996.

[9] Ultimus. Workflow suite. Business workflow automation. 4915 Waters Edge Dr., Suite 135, Raleigh, NC 27606. http://www.ultimus1.com.