Advances in Computational & Experimental Engineering & Science

Extending Workflow Systems with QoS Management

Jorge Cardoso[1]

Summary

As organizations adopt new working models, such as e-commerce, new challenges arise for workflow management systems (WfMSs). One such challenge is that of quality of service (QoS) management. A good management of QoS directly impacts the success of organizations participating in e-commerce activities by better fulfilling customer expectations and achieving customer satisfaction. In this paper, we discuss the implementation of a workflow QoS model for the METEOR workflow management system. We describe the components that have been changed or added, and discuss how they interact to enable the specification, computation, and monitoring of QoS.

Introduction

Organizations operating in modern markets, such as e-commerce, involve a systematic design, planning, control, and management of business processes. One important requirement of these processes is the quality of service (QoS) management. This requirement is a new challenge for workflow systems. While QoS has been a major concern for networking, real-time applications, and middleware, few research groups have concentrated their efforts on enhancing workflow systems to support workflow quality of service (QoS) capabilities. Most of the research carried out to extend the functionality of workflow systems QoS has only been done in the time dimension, which is only one of the dimensions under the QoS umbrella. Furthermore, the solutions and technologies presented are still preliminary and limited [1].

This paper enumerates and describes the enhancements that need to be made to workflow management systems to support processes constrained by QoS requirements. Our work in this area started with the definition of a QoS model for workflows [2]. The implementation of our QoS model and methodologies has been carried out for the METEOR system to allow the specification, recording, and computation of QoS [3]. The support of QoS requires the modification and extension of several workflow system components, and the development of additional modules.

This paper is structured as follows. We start by briefly describing the METEOR workflow system and its main architecture. We then present our QoS model and describe the modifications that have been made to the workflow enactment service to hold the QoS model. Next, we analyze the implications of each QoS dimension (time, reliability, and cost) to the workflow system architecture. We describe the modification of existing

[1] Departamento de Matemática e Engenharias, Universidade da Madeira, 9000-390 Funchal, Portugal

Advances in Computational & Experimental Engineering & Science

components and the creation of new modules that have been developed to support the workflow QoS management. Finally, the last section presents our conclusions.

Workflow QoS Implementation

The QoS model that we have developed has been implemented for the METEOR workflow management system. The METEOR project is represented by both a research system [4], and a suite of commercial systems that provide an open system based, high-end workflow management solution, as well as an enterprise application integration infrastructure. The system has been used in prototyping and deploying workflow applications in various domains, such as bio-informatics, healthcare [5], telecommunications [6], and defense [7].

Figure 1 describes the components that make up the METEOR system and the components that have been modified, extended, and created to enable QoS management. The work discussed in this paper is part of the research system and is not part of any commercial product yet.

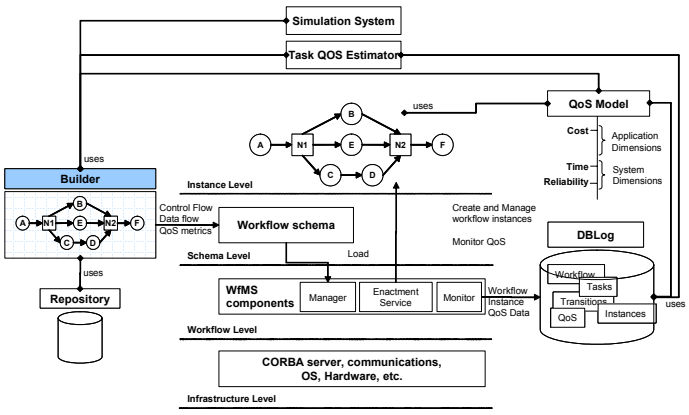

Figure 1 – QoS Management Architecture

QoS Model

Quality of service can be characterized along various dimensions. Based on previous studies and on our experience in the workflow domain, we have constructed a QoS model composed of three dimensions: time, cost, and reliability.

Time (T) is a common and universal measure of performance. For workflow systems, task response time can be defined as the total time needed by a task to transform a set of inputs into outputs. Cost (C) represents the cost associated with the execution of

Advances in Computational & Experimental Engineering & Science

workflow tasks. During workflow design, prior to workflow instantiation, and during workflow execution it is necessary to estimate the cost of the execution to guarantee that financial plans are followed. Reliability (R) corresponds to the likelihood that a task will perform when a user demands it; it is a function of the failure rate. The reliability dimension is a function of the number of times the success state is reached and the number of times the failure state is reached.

Enactment Service

In METEOR enactment service (ORBWork), task schedulers, task managers, and tasks are responsible for managing runtime QoS metrics. From the implementation point of view, we divide the management of the QoS dimensions into two classes: the system and the application class. The dimensions of the system class are managed by system components (e.g. a task scheduler), while the dimensions of the applications class are managed by components dynamically created to support a particular workflow application (e.g. a task implementation). In our system, the system class includes the time and reliability dimensions, while the application class includes the cost dimension.

Managing Time

Task response time (T) is composed of two major components: *delay time* (DT) and *process time* (PT). Delay time is further broken down into *queuing delay* (QD) and *setup delay* (SD). This makes the response time of a task t represented as followed:

$$T(t) = DT(t) + PT(t) = QD(t) + SD(t) + PT(t) \quad (1)$$

To efficiently manage the time dimension, workflow systems must register values for each of the functions involved in the calculation of task response time (T). The time dimension has its values set according to the task structure illustrated in Figure 2.

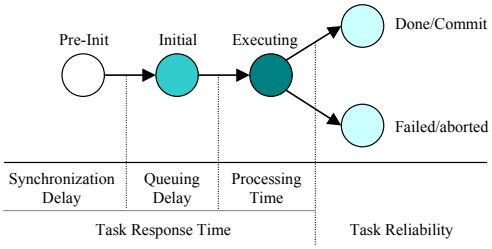

Figure 2 – Revised task structure (extended from [8])

Each state has been mapped to one of the functions that compose the time dimension. METEOR system follows this task structure to represent workflow task execution

Advances in Computational & Experimental Engineering & Science

behavior [8]. To more effectively support QoS management, the original structure has been extended, with the inclusion of the *Pre-Init* state, as shown in Figure 2.

The synchronization delay time is calculated based on the difference between the time registered when a task leaves the *pre-init* state and the time registered when it enters the state. A task *t* remains in the *pre-init* state as long as its task scheduler is waiting for another transition to be enabled in order to place the task into an initial state. This only happens with synchronization tasks, i.e. *and-join* tasks [9], since they need to wait until all their incoming transitions are enabled before continuing to the next state. For all other types of input and output logic (*xor-split*, *xor-join*, *and-split*) the synchronization delay time is set to zero.

As for the synchronization delay time, the queuing time is the difference between the time a task leaves and enters the *initial* state. A task in the *initial* state indicates that the task is in a queue waiting to be scheduled (by its task scheduler). Task schedulers treat their queues with a FIFO policy. One interesting queuing policy variation is associated with the scheduling of human-tasks. For a human-task instance, being in the initial state means that the task has been placed in a worklist for human processing. A user can select any human-task in a worklist, as long as the user role matches the task role. In this case, the queuing policy is SIRO (Serve In Random Order). Depending on the workflow system, other useful queuing policies can be used, such as priority queues. When a task instance enters a queue a time-stamp is attached to it. When the task is removed from the queue for scheduling, another time-stamp is attached to it so that the total queuing time can be calculated later. When a task is ready to be executed it transits to the executing state. As with the previous calculations, the time a task remains in this state corresponds to the processing time.

Managing Reliability

During a task realization, a number of undesirable events may occur. Depending on the successful or unsuccessful execution of a task, it can be placed in the done or fail state (for non-transactional tasks) and commit or abort (for transactional tasks). The former state indicates that the task execution was unsuccessful, while the latter state indicates that a task is executed successfully [8].

When an undesirable event occurs, an exception is generated. An exception is viewed as an occurrence of some abnormal event that the underlying workflow management system can detect and react to. If an exception occurs during the invocation of a task realization, its task enters the fail/abort state. In our implementation, it is the responsibility of task schedulers to identify the final state of a task execution in order to subsequently set the reliability dimension. Later this information is used to compute the failure rate, which is a function between the number of times the failed/aborted state is reached and the number of times state done/committed is reached. To describe task reliability we follow a discrete-time modeling approach. Discrete-time models are

Advances in Computational & Experimental Engineering & Science

adequate for systems that respond to occasional demands such as database systems. We use the stable reliability model proposed by Nelson [10], for which the reliability of a task *t* is,

$$R(t) = 1 - \text{failure rate} \quad (2)$$

Managing Cost

When a task is ready to execute, a task scheduler activates an associated task manager. The task manager oversees the execution of the task itself. Task managers are implemented as an object and are classified as transactional or non-transactional, depending on the task managed. Human tasks do not have an associated task manager.

Once activated, the task manager stays active until the task itself completes. Once the task has completed or terminated prematurely with a fault, the task manager notifies its task scheduler. The task manager is responsible for creating and initializing a QoS cost data structure from QoS specifications for the task overseen. When the supervised task starts its execution, the data structure is transferred to it. If the task is a non-transactional one (typically performed by a computer program), a set of methods is available to programmatically manage the initial QoS estimates. Once the task completes its execution, the QoS data structure is transferred back to the task manager, and later from the task manager to the task scheduler.

In the case of human tasks (performed directly by end-users), the QoS specifications for the cost dimension is included in interface page(s) (as HTML templates) presented to the end-user. When executing a human task, the user can directly set the cost dimension to values reflecting how the task was carried out. As mentioned previously, human-tasks do not have a task manager associated with them, and therefore a specific task scheduler is responsible for the task supervision. When the task completes its realization, the task scheduler parses the interface page(s) and retrieves the new QoS metrics that the user may have modified.

Conclusions

The use of workflow systems to manage, improve, and re-engineer business processes enables organizations to reduce costs and increase efficiency. While quality of service (QoS) management is of a high importance to organizations, current WfMSs and workflow applications do not provide full solutions to support QoS.

In this paper we explain how to implement a QoS model to a sophisticated workflow management system (the METEOR system) to enable the QoS management of processes. The support of QoS management requires the modification and extension of several workflow system components, including the enactment system, task schedulers, task managers, and the task model.

Advances in Computational & Experimental Engineering & Science

Reference

1. Eder, J., Panagos, E., Pozewaunig, H., and Rabinovich, M. (1999): Time Management in Workflow Systems. in *BIS'99 3rd International Conference on Business Information Systems*. Poznan, Poland: W. Abramowicz and M.E. Orlowska, Springer Verlag. p. 265-280

2. Cardoso, J., Sheth, A., and Miller, J. (2002): Workflow Quality of Service. in *International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02)*. Valencia, Spain, Kluwer Publishers

3. Cardoso, J. (2002): *Quality of Service and Semantic Composition of Workflows*, in *Department of Computer Science*. Ph.D. Dissertation, University of Georgia: Athens, GA. p 215.

4. METEOR (2004): *METEOR (Managing End-To-End OpeRations) Project Home Page*, LSDIS Lab. http://lsdis.cs.uga.edu/proj/meteor/meteor.html

5. Anyanwu, K., Sheth, A., Cardoso, J., Miller, J.A., and Kochut, K.J. (2003): Healthcare Enterprise Process Development and Integration. *Journal of Research and Practice in Information Technology, Special Issue in Health Knowledge Management*, **35**(2): p. 83-98.

6. Luo, Z. (2000): *Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes*, in *Department of Computer Science*. Ph.D. Dissertation, University of Georgia: Athens, GA. p 171.

7. Kang, M.H., Froscher, J.N., Sheth, A.P., Kochut, K.J., and Miller, J.A. (1999): A Multilevel Secure Workflow Management System. in *Proceedings of the 11th Conference on Advanced Information Systems Engineering*. Heidelberg, Germany: M. Jarke and A. Oberweis, Springer-Verlag. p. 271-285

8. Krishnakumar, N. and Sheth, A. (1995): Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations. *Distributed and Parallel Databases Journal*, **3**(2): p. 155-186.

9. Kochut, K.J. (1999): *METEOR Model version 3*. Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia: Athens, GA.

10. Nelson, E.C. (1973): *A Statistical Basis for Software Reliability*. TRW Software Series.