Towards Distributed Workflow Process Management

Johann Eder, Euthimios Panagos

AT&T Labs – Research

eder@acm.org, thimios@research.att.com

Abstract

Today, companies are driven by the rate at which technologies change and move towards global enterprises and virtual organizations in order to stay competitive. In global and virtual enterprises, business processes consist of multiple sub-processes that may span multiple time zones, organizational boundaries, and legal domains. Current workflow technology does not provide the necessary functionality to model, enact, and manage such processes due to its mostly centralized, coupled architecture and limited process management capabilities. In this paper, we present our on-going efforts to address some of the issues involved. In particular, we present an event-based infrastructure that supports distribution and decoupling, together with time-related constructs for addressing the time aspects of process management.

1. Introduction

Today, we experience an ever-increasing number of corporate merges, acquisitions, and strategic alliances. In this environment, the infrastructures already employed by the member companies and their business processes have to be integrated. However, integration can be a herculean task due to the heterogeneity of the information and communication systems involved and the fact that the organizational structure of the resulting entity may not follow the traditional, strictly hierarchical delegation model. The latter implies that a decoupled infrastructure, which preserves autonomy and supports dynamic changes, should be employed.

Traditionally, advances in Internet and distributed middleware technologies were not the driving factors in the design of workflow systems. Rather, a centralized, tightly coupled architecture was employed, together with proprietary components and tools. Today, the above shortcomings were identified, and advances in distributed middleware and Internet standards are influencing workflow technology. Nevertheless, several areas still need to be addressed in order to be able to support the dynamic, distributed environment that characterizes global and virtual organizations. Some are these areas are hyper-process modeling, distributed role and worklist management, interoperability, asynchronous and decoupled communication, and better process management tools.

While there have been some efforts by the Workflow Management Coalition (WfMC) [12] and the OMG Business Object Domain Task Force [6] to address the above shortcoming, these efforts have not been fully materialized yet. In addition, these efforts address only parts of the issues involved. In particular, they primarily focus on the selection, instantiation, and enactment of business processes. They do not address issues of distributed exception handling, nor do they address time management issues. Furthermore, the proposed standards address workflow systems that operate in a coupled mode, i.e., workflow engines have to know each other in order to communicate and jointly execute processes.

The contributions of this position paper include an event-based workflow infrastructure and modeling constructs for addressing the time aspects of process management. In particular, we discuss the functionality an event service should provide in order to address distributed, decoupled, and dynamic workflow executions. In addition, we briefly cover some of the additional components we believe they should be part of the overall infrastructure for supporting global and virtual companies as well as e-commerce service providers. Finally, we introduce timing constructs that are required for expressing the various time dependencies that may exist between the activities and sub-processes that constitute distributed processes.

2. Workflow Standards and Current State of WFMSs

According to the Workflow Management Coalition (WfMC) reference model [13], a Workflow Management System (WFMS) consists of an engine, application agents, invoked applications, a process definition tool, administration and monitoring tools, and an interoperability component that is used for selecting, instantiating, and executing remote processes [12]. The process definition tool is a visual editor that is used to define the schema of workflow processes (i.e., specify the activities that constitute the workflow and precedence relationships among them). The same schema can be used for creating multiple *instances* of the same process at a later time. The workflow engine and the various tools communicate with a workflow database to store and update workflow-relevant data, such as schemas, statistical information, and control information required for executing and monitoring the active process instances.

The OMG Business Object Domain Task Force has produced a workflow specification that utilizes events during workflow execution [6]. In particular, events are raised when process instances are created and terminated, when work items are created and terminated, when state changes occur and so on. However, the specified events and the proposed architecture cannot efficiently handle distributed, dynamic workflows. While the IDL-based specification can address interoperability, the existing OMG event and notification services [4,5] do not provide the necessary high-level constructs for handling distributed workflows. For example, there are no mechanisms for grouping together event producers in order to encapsulate the events that are raised within a particular sub-workflow. In addition, no time management functionality is discussed in the above proposal.

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 generate events and 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. Support for time management in existing WFMSs is limited to process simulation (to identify process bottlenecks, analyze execution duration of activities, *etc*.), assignment of activity deadlines, and triggering of process-specific exception handling activities, referred to as *escalations*, when deadlines are missed at run time.

3. Event-Based, Decoupled Workflow Infrastructure

In this section, we outline the distributed workflow infrastructure we are currently working on for addressing decoupled, distributed, and dynamic business processes. Figure 1 shows the high-level architecture, having an event-notification service as its key component module. In addition, we assume the availability of a distributed storage component that is used for exchanging process-specific data as well as workflow-specific data. Such a distributed storage component is based on WEB technology (similar to [2] and the WFMS Panta Rhei [1]). In particular, we assume that XML is used for workflow-specific data. For process-specific data, XML or HTML can be used. Furthermore, we assume that a security mechanism is employed to ensure authenticated and secure access to the various system components. Finally, a distributed LDAP-based directory is used for storing role information together with process information. This directory provides naming and lookup facilities.

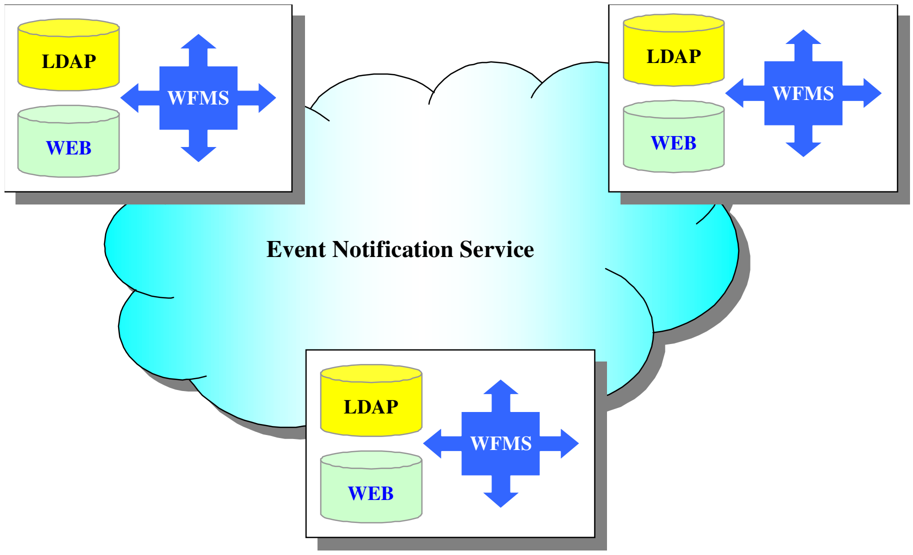

Figure 1: Decoupled, distributed workflow architecture

By using an event notification infrastructure that provides decoupled notifications of events (i.e., event suppliers need not be aware of the event consumers and event consumers need not be aware of event suppliers), the following benefits are realized:

- Workflow participants (e.g., workflow engines and agents) can subscribe to receive events that trigger the start of workflow activities and processes, and events that describe state changes in the workflow processes they are interested in. Therefore, workflow participants can be dynamically added or removed without requiring any modifications to the existing architecture;

- Workflow administrators can subscribe to events that correspond to state changes in existing workflow instances and, thus, monitor the lifecycle of processes;

- Exception handling can be driven by the appropriate events. For example, by subscribing to various alarm and resource availability and capacity events (e.g., agent availability and workload), timing and resource-related exceptions are handled.

An important feature of an event notification service for supporting distributed workflows is the ability to support multiple event domains. Domains may correspond to different organizations, departments within organizations, administrative domains, and so on. This functionality is required for being able to distinguish the various events that are raised within each domain, while maintaining the autonomy of the domain. Here the goal is to avoid imposing a new naming scheme on all the involved members of a global or virtual organization. Our event-notification service, READY [3], provides such support and, in addition, it uses domain routers for connecting event domains in a hierarchical or peer-to-peer topology. Domain routers can be used for encapsulating event mappings between domains, enforce access restrictions, and regulate the flow of events between domains.

READY clients interact with READY using *admin*, *supplier*, and *consumer* sessions. Admin sessions are used for creating/destroying supplier and consumer sessions and sessions groups and for other administrative operations. Supplier sessions are used to supply events to the service, and suppliers must declare the kinds of events that they will supply. Consumer sessions are used to register *specifications*, which describe both event patterns and actions to take when matches are found for these patterns. The most common action, *notify*, causes the consumer session to deliver a notification with the matched event(s). Notification behavior is controlled by session properties, including quality of service properties (QoS) that control the reliability of notification delivery, and delivery properties that control the delivery order. Suppliers, consumers, and specifications can be grouped together, enabling sharing of specifications, uniform control over QoS and delivery properties, and efficient suspend/resume operations of a large number of specification.

Another important feature of READY is its ability to support self-describing events. A READY event type definition specifies a set of required and optional fields, where each field contains a field name, a type identifier, and a value. READY types can have subtypes, and subtype declarations simply add additional required or optional field specifications to those of their parent types. Compound type identifiers such as **`WF.Process.Start`** can be used with the convention that the structure of the type corresponds to the type hierarchy. In our proposed workflow infrastructure, event types are registered with the LDAP directory using XML to describe their structure and semantics.

Event notifications can contain process-specific data as well as pointers to application-specific data. The process-specific data may correspond to results of the execution of a particular step in a workflow as well as specifications regarding the next steps that have to be followed. In addition, notifications can contain access control rights and *time-to-live* fields. Time-to-live fields are important in the case where the next step of a given workflow needs to be carried out within a given time period. Consequently, the appropriate agent or workflow engine should receive the notification before the expiration of this deadline otherwise a time exception is raised. Finally, since event notifications may be delivered in a variety of ways (e.g., email, paging, entry in a worklist, *etc.*), agents and workflow engines can choose the appropriate medium for receiving them. The medium can be selected based on properties of notifications, such as time-to-live, event source, priority, and workflow process instance id, and it can be modified dynamically.

4. Time Constraints

Time constraints are crucial in designing and managing distributed business processes and, therefore, time management should be part of the core management functionality provided by workflow systems to control the lifecycle of processes. At build-time, workflow modelers need means to represent time-related aspects of business processes (e.g., activity durations and time constraints between activities) and check their feasibility. At run-time, process managers need pro-active mechanisms for being notified of possible time constraint violations. Workflow participants need information about urgencies of the tasks assigned to them to manage their personal work lists. If a time constraint is violated, the workflow system should be able to trigger exception handling to regain a consistent state of the workflow instance. Business process re-engineers need information about the actual time consumption of workflow executions to improve business processes. Controllers and quality managers need information about when and how long activities of a workflow instance have been performed.

Time constraints belong to two categories: *structural* and *explicit*. Structural time constraints follow implicitly from control dependencies and activity durations of a workflow schema. They arise from the fact that an activity can only start when its predecessor activities have finished. Explicit time constraints are derived from organizational rules, laws, commitments, and so on. Examples of such constraints include: (1) an invitation for a meeting has to be mailed to the participants at least one week before the meeting; (2) after a hardware failure is reported, the service team has to be at the customer's site within 4 hours; (3) vacant positions can be announced at the first Wednesday of each month; (4) loans above USD 1M have to be approved at a regular meeting of the board of directors (i.e., such applications can be approved on dates where a regular meeting of the board of directors is scheduled).

Explicit time constraints are temporal dependencies between events or bindings of events to certain sets of calendar dates. In workflow systems these events correspond to the start and end of activities. We introduce the following explicit time constraints:

- **Lower-Bound Constraint**: The distance between two events must be greater than or equal to δ. We write `lbc(A,B,δ)` to express that δ is a lower bound for the time-interval between source event `A` and destination event `B`.

- **Upper-Bound Constraint**: The distance between two events must be smaller than or equal to δ. We write `ubc(A,B,δ)` to express that δ is an upper bound for the time-interval between source event `A` and destination event `B`.

- **Fixed-Date Constraint**: To express that an event `B` can only occur on fixed date(s), we write `fdc(B,T)`, where `T` is a *fixed-date object*. A fixed-date object is an abstraction that generalizes a, typically infinite, set of dates such as "every other Monday". This abstraction provides methods for returning valid before and after dates for a given date, maximum distance between two valid dates, and maximum distance between valid dates of two fixed-date objects.

An example of lower-bound constraint includes a chemical process control, where one reaction may be initiated only when certain time passes after the start of some other reaction. Upper-bound constraints are even more common, e.g., the requirement that a final patent filing is done within a certain time period after the preliminary filing, or time limits for responses to business letters.

We have already developed techniques for addressing some of these constraints at process build and instantiation times, and taking pre-emptive actions at run time when potential time exceptions may materialize [7,8,9,11]. Currently, we are in the process of extending our techniques to check the satisfiability of these constraints by annotating the workflow graph with time information based on the CMP or PERT techniques [10]. While preliminary checks can be performed at process build and instantiation, we are particularly interested in run time techniques that monitor (using our event-based infrastructure) the execution progress and changes in the state of the systems (e.g., agent availability) and enforce the specified time constraints.

5. Conclusions

In this position paper, we presented our approach to management of workflow processes in a distributed environment. The core or our approach is an event notification service that provides efficient, asynchronous, and decoupled notification of events. The main goals of the work are to provide an infrastructure that supports dynamic workflows and, in addition, to provide timing constructs so that time management is facilitated in such an environment. Currently, we have introduced timing constructs for expressing explicit time constraints, and we are developing techniques for testing the satisfiability of these constraints at process built and instantiation times and, also, enforcing them at run time.

Acknowledgments

We would like to thank Michael Rabinovich for his valuable comments and for helping us improve the presentation of this position paper.

References

1. Eder, J., Groiss, G., Liebhart, W. “*The Workflow Management System Panta Rhei*.” In Advances in Workflow Management Systems and Interoperability, Springer-Verlag, 1998.
2. Groiss, H., Eder, J. “*Interoperability with World Wide Workflows*.” In 1st Word Conference on Integrated Design and Process Technology, Austin, Texas, 1995.
3. Gruber, R., Krishnamurthy, B., Panagos, E. *“High-Level Constructs in the READY Event Notification System*.” In 8th ACM SIGOPS European Workshop on Support for Composing Distributed Applications, Sintra, Portugal, 1998.
4. Object Management Group. “*Event Service Specification*.” ftp://www.omg.org/pub/docs/formal/97-12-11.pdf.
5. Object Management Group. “*Notification Service Specification*.” ftp://www.omg.org/pub/docs/telecom/98-06-15.ps.
6. Object Management Group. *“Workflow Management Facility. Request for Proposal*.” ftp://ftp.omg.org/pub/docs/cf/97-05-06.pdf.
7. Panagos, E., Rabinovich, M. “*Escalations in Workflow Management Systems*.” In the DART Workshop, Rockville, Maryland, 1996.
8. Panagos, E., Rabinovich, M. “*Predictive Workflow Management*.” In Proceedings of the 3rd International Workshop on Next Generation Information Technologies and Systems." Neve Ilan, ISRAEL, 1997.
9. Panagos, E., Rabinovich, M. “*Reducing Escalation-Related Costs in WFMSs*.” In NATO advanced Study Institute on Workflow Management Systems and Interoperability, Istanbul, Turkey, 1997.
10. Philipose, S. "*Operations Research - A Practical Approach*." Tata McGraw-Hill, New Delhi, New York, 1986.
11. Pozewaunig, H., Eder, J., Liebhart, W. “*ePERT: Extending PERT for Workflow Management Systems*.” First East-European Symposium on Advances in Database and Information Systems, St. Petersburg, Russia, 1997.
12. Workflow Management Coalition. “*Workflow Standard - Interoperability Abstract Specification*.” http://www.aiim.org/wfmc/standards/docs/I4Mime1x.PDF.
13. Workflow Management Coalition. “The Workflow Reference Model.” http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf.