Research Report

Providing High Availability in Very Large Workflow Management Systems

M. Kamath G. Alonso R. Günthör C. Mohan

IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA
{*kamath, gustavoa, rgunther, mohan*}*@almaden.ibm.com*

IBM Research Report RJ9967, July 1995

Research Division
Yorktown Heights, New York • San Jose, California • Zurich, Switzerland

Providing High Availability in Very Large Workflow Management Systems *

M. Kamath† G. Alonso‡ R. Günthör C. Mohan

IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA
{*kamath, gustavoa, rgunther, mohan*}*@almaden.ibm.com*

IBM Research Report RJ9967, July 1995

ABSTRACT:

Workflow management systems (WFMSs) support the modeling, coordinated execution and monitoring of business processes within an organization. In particular, very large workflow management systems are used in organizations where the number of users may be in the tens of thousands, the number of process instances in the hundreds of thousands, and the number of sites in the thousands, all distributed over wide geographic areas. In these environments, failure of the WFMS or the underlying workflow database which stores the meta-information about the processes is not tolerable and hence continuous availability is a key aspect of the system. This paper addresses the problem of providing high availability in workflow management systems by proposing a backup technique which ensures that execution of a process instance can be resumed at any point in time in the event of failures. An essential characteristic of our backup scheme is that it allows the user to define different availability levels in order to avoid high costs for maintaining backups for all process instances. The backup scheme to support the different availability levels is implemented using the workflow semantics, which we believe will — (i) make it independent of the underlying workflow database, thus permitting the use of heterogeneous databases as primary and backup, (ii) reduce overheads, especially when compared to backup schemes provided by database systems.

---

*This work is partially supported by funds from IBM Hursley (Networking Software Division) and IBM Böblingen (Software Solutions Division). Even though we refer to specific IBM products in this paper, no conclusions should be drawn about future IBM product plans based on this paper's contents. The opinions expressed here are our own.

†This work was performed while the author was visiting IBM Almaden from the Computer Science Department at the University of Massachusetts, Amherst, MA 01003, USA.

‡This work was performed while the author was visiting IBM Almaden. He is now at the Institute for Information Systems, Database Group, ETH Zentrum, CH-8092 Zürich, Switzerland

1. Introduction

Workflow management systems, WFMS, enjoy an increasing popularity due to their ability to coordinate and streamline complex business processes within large organizations. As such, WFMS are not very different from other collaborative tools in that they try to provide an environment to foster cooperation between several agents. What makes WFMSs unique is the type of environment in which they are used. The goal of WFMSs is to orchestrate the execution of business processes, ultimately being responsible for the control of the organization's activities. Thus, a WFMS becomes not just a support tool but an integral part of the organization. This is substantiated by the fact that the most likely candidates to use a WFMS are large corporations such as banks, insurance companies and telecommunication companies which need to coordinate several instances of a variety of business processes. Current customer requirements indicate that the number of potential users can be in the tens of thousands. In some cases, the number of process instances has been estimated as high as 300,000 instances per month, while the number of sites connected to the WFMS can be in the thousands, distributed over a wide geographic area and based on heterogeneous systems. With these figures, continuous availability becomes a crucial aspect of any successful commercial WFMS. Surprisingly, this is an issue that has been ignored by workflow developers, and to our knowledge, has not yet been addressed by research in the area of workflow management.

This paper reports ongoing research in the availability of WFMSs, as part of the research efforts of the *Exotica* project [MAGK95]. The research has been centered around *FlowMark* [IBM95a, IBM95b, LA94, LR94], a WFMS from IBM, however our results can be easily generalized to any WFMS.

In this paper, our focus will be on WFMSs that use a centralized database (workflow database) to store meta-information about the business processes. They conform to the Workflow Management Coalition's reference model [WfM95] and typically handle *production* workflows [McC92]. In environments where several workflow databases coexist, each of them typically stores a different subset of processes instances. Hence, if one database fails, all process instances running off that database will stop their execution. This behavior is unacceptable for several critical business processes which cannot be delayed. While all other components of the WFMS can be replaced dynamically by using the appropriate architecture [AKA+94], the database remains a single point of failure that may have grave consequences. To address this problem, our approach is to use process instance replication to guarantee that if the database where the particular instance resides fails, execution can be resumed elsewhere based on the replicated information.

The novelty of our approach is to use workflow semantics[1] as the basis for replication and backup, as opposed to low level constructs such as pages or log records used in traditional database backup techniques. The reasons are as follows:

- *Necessity for WFMSs that are database independent:* Relying on the backup facilities supplied by the underlying database ties the system to the platforms where such a database runs. Since we envision a system in which heterogeneous databases, even as different as relational and object-oriented to be used as backup for each other, low level techniques as those provided by database management systems cannot be used. By using application (workflow) level backup, the replication scheme becomes independent of the underlying platform, thus allowing the use of heterogeneous databases.
- *Need for flexibility in replicating process instance data:* In traditional database backup techniques, the units of exchange tend to be pages or log records [GMP90, BGHJ92, MTO93] which makes it difficult to control which data is actually being replicated. If a database backup technique is used to replicated data between two workflow databases, log records of changes to all objects, including those that correspond to static information about processes will be replicated. This is not really required and there can be excessive overheads when several hundred thousands processes run concurrently. Instead the overheads can be minimized by providing different availability levels (each having different overheads and degree of reliability) and exploiting workflow semantics to minimize the amount of information that is replicated.

The specific contributions of this paper are:

1. Provision for different availability levels by categorizing process instances as *normal*, *important*, and *critical*, each with its own backup approach.
2. A backup architecture and mechanisms to efficiently use the resources available in a distributed workflow system to perform backup,*i.e.*, use the workflow servers and workflow databases to function both as a primary (for some process instances) and as a secondary (for some other process instances).
3. Backup algorithms to replicate process instance data for each of the above categories during normal processing and when there are failures at the primary or backup site.

The rest of the paper is organized as follows: in the next section we describe the basic ideas behind a WFMS. Backup requirements in WFMSs and our approach is briefly presented

[1]From the perspective of the database that stores the meta-information, the WFMS is an application and hence workflow semantics can also be referred to as *application* semantics.

in section 3. A discussion on backup techniques and how they can be adapted to WFMS is presented in section 4. Section 5 discusses availability levels, the backup architecture and issues related to object state replication. In section 6 we introduce the necessary data structures and described the backup algorithms. Section 7 concludes the paper.

2. Workflow Management

In this section we present the basic concepts of workflow management. These concepts will be used when discussing particular aspects of the implementation of our replication and backup technique. Specific details of workflow management systems are described based on FlowMark.

2.1. Workflow Concepts and Systems

A common term used to refer to the work performed in large organizations is *business process*, defined by the Workflow Management Coalition as "a procedure where documents, information or tasks are passed between participants according to defined sets of rules to achieve, or contribute to, an overall business goal" [WfM95]. A *workflow* is a particular representation of a business process. Accordingly, a *workflow management system*, WFMS, is defined as "a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic" [WfM95]. To be more precise, the role of a WFMS is the scheduling and coordination of all the activities encompassing a business process. This involves determining the order of execution, interacting with each activity's tools, establishing the data flow between activities, mapping activities to users, checking the timeliness of the execution, monitoring the progress and determining when a process has terminated.

These concepts have been around for a number of years. Only recently, however, the technology required to implement suitable systems has been available. Today there are more than 70 products that claim to be workflow solutions [Fry94]. Historically, many areas have influenced workflow management: form management [Tsi82], electronic mail systems [GSS92, MGL+87], office automation [KW86, KHK+91], and advanced transaction models [Elm92], to mention a few. Currently, there is a considerable amount of attention devoted to this area [GHS95, Hsu95, BMR94, She94, Hsu93, TAC+93, MS93, MMWF93, DHL91, DHL90].

2.2. Workflow Model

A workflow model is a description of a business process. For this purpose, the Workflow Management Coalition has proposed the following reference model [WfM95]. A business process is represented by a schema that includes the *process* name, version number, start and termination conditions and additional data for security, audit and control. A process consists of *activities* and *relevant data*. Each step within a process is an *activity*, which has a name, a type, pre- and post-conditions and scheduling constraints. It also has a *role* and may have an *invoked application* associated with it. The *role* determines who will execute the activity. The *invoked application* is the tool to be used in the execution, which may range from computer programs to human activities with no specific tool, e.g., a meeting. Invoked applications have a name, type, execution parameters and, in the case of programs, a location or access path. Furthermore, activities use *relevant data* defined as the data passed to and produced by the activities. Relevant data includes name and type, as well as the path specifying the activities that have access to it. The flow of control within a process - what to execute next - is determined by *transition conditions*, usually boolean expressions based on relevant data. The users of the system are represented in terms of *roles*. Finally, and for notational purposes, a specific invocation of a process is referred to as a *process instance*. For a given process, there can be many instances of it running concurrently. The instances have the same components and structure as the process.

The architectural components of the reference model are as follows. A *process-definition tool* is used to define the schema of each business process. A *workflow engine* creates a process instance, interprets the process schema (also referred to as *navigation*) and determines which activities are ready for execution. If an activity is to be performed by a role, then the eligible role is notified by placing the activity on the role's *worklist*. The role has to then select the activity for execution from the worklist using a *worklist-handler*. On the other hand, if the activity is to be performed automatically, then the workflow engine selects it for execution. If an application program is to be invoked to execute the selected activity, the workflow engine notifies an *application-agent* that exists at the location (node) where the program is to be invoked. The application agent then executes the program and returns the results of the program to the workflow engine.

2.3. FlowMark Architecture

FlowMark runs across different platforms[2] and supports distribution of most of its components. It is based on a client/server architecture and uses a centralized database to store meta-information about workflows. In the case of FlowMark, an object-oriented database[3] is used. In particular, the schema definition of a business process and all the runtime information related to active process instances are stored in this database. We have selected FlowMark as the specific WFMS for our study since its architecture closely resembles the architecture described by the reference model. Besides the database server, FlowMark is organized into four other components. The *FlowMark Server* acts as the workflow engine, the *Runtime Client* acts as the worklist-handler, the *Program Execution Client* acts as the application-agent, and the *Buildtime Client* acts as the process-definition tool. A single database usually acts as a repository to several workflow (FlowMark) servers.

2.4. Navigation in FlowMark

Navigation is the procedure by which the server determines the next activities to execute. It takes place through transactional updates to the centralized database. However, the reference model of the Workflow Management Coalition does not provide implementation details. For this purpose, we use FlowMark since its model closely resembles the reference model. In FlowMark, omitting users and invoked applications, an activity has the following concepts associated with it: the *flow of control*, which is specified by *control connectors* between activities, is the order in which these activities have to be executed. Each control connector has a state. Before the activity from where it originates finishes, the control connector is *unevaluated*. Once the activity terminates, the control connector is evaluated and its state is set to TRUE or FALSE. The evaluation involves checking whether the *transition condition* for the connector is true or false. Such transition conditions are boolean expressions involving data returned by activities in the output containers. For each activity, there is an *input data container* and an *output data container* where the invoked application takes its input and places its output. The *flow of data*, specified through *data connectors*, corresponds to mappings between output data containers and input data containers and allows passing data from activity to activity. Finally, activities have a *start condition*, which determines whether the activity will be executed or not, and an *exit condition*, which determines whether an activity has been successfully completed.

[2]Currently on AIX, OS/2, and Windows.

[3]ObjectStore