# FlowBack: Providing Backward Recovery for Workflow Management Systems

Bartek Kiepuszewski Ralf Muhlberger Maria E. Orlowska

Distributed Systems Technology
Centre Pty Ltd
The University of Queensland
Qld 4072 Australia
+61 7 3365 4310

<ralf, maria>@dstc.edu.au

## 1. ABSTRACT

**The Distributed Systems Technology Centre (DSTC) framework for workflow specification, verification and management captures workflows transaction-like behavior for long lasting processes. FlowBack is an advanced prototype functionally enhancing an existing workflow management system by providing process backward recovery. It is based on extensive theoretical research ([3],[4],[5],[6],[8],[9]), and its architecture and construction assumptions are product independent. FlowBack clearly demonstrates the extent to which generic backward recovery can be automated and system supported. The provision of a solution for handling exceptional business process behavior requiring backward recovery makes workflow solutions more suitable for a large class of applications, therefore opening up new dimensions within the market. For the demonstration purpose, FlowBack operates with IBM FlowMark, one of the leading workflow products.**

### 1.1 Keywords

Workflow, transactions, advanced transaction models, backward recovery, compensation, FlowBack, FlowMark



## 2. INTRODUCTION

Historically, the coordination of business activities, the order of operations, and recovery from exceptional process behavior have all been performed manually. In recent years, the possibility of automating the flow and scheduling of tasks has been explored and substantial development has been made in what is generally called workflows technology. The workflows represent organization flow of information from one processing entity, either manual or automated, to another. The workflow management systems, partially or fully, take over the responsibility for coordination of the multi-system execution of tasks. Workflow technology has emerged as a multi-disciplinary field with significant contributions from the areas of software engineering, software process management, database management and distributed systems.

There exists a strong school of thought that supports a need for concurrency control and synchronization of workflow processes, addressing correctness concerns during workflow execution. In particular, production workflows, on which this project is focused, involve the coordination of organizational information processing systems that are usually based on database management systems, having multi-tier, client-server architectures with a central workflow server responsible for management of business processes. These workflows, in contrast to ad-hoc or administrative document management workflows, have well defined procedures for a rigorous and multiple repetitive process instance execution, which may span several heterogeneous information systems. Thus workflows management systems are complex software products which should provide a number of functions/services to different groups of users. Basically, among others, they should have extensive features to define the internal task structure, control the execution of activities involving different types of processing entities and support reliable forward recovery services (for system failures) and backward recovery services (for external actions such as, for example, cancellation of a customer request).

The support for backward recovery service is the main focus area of the FlowBack project.

## 3. RECOVERY FOR COMPLEX PROCESSES

The term "transactional workflows" has been introduced [14] to clearly recognize the relevance of transactions in the context of workflows. Transactional workflows involve coordinated execution and suggest selective use of transactional properties for individual tasks or entire workflows. Basically, they use advanced transaction models as a core concept to specify workflow correctness, data consistency and reliability [1],[10],[16]. A great deal of research has been devoted to broaden the scope of classical transactions [1]. However, study of transaction models for workflows is still in its early phase [7] [13] [14]. So far it seems to emphasize only flow control within applications. In this project we have considered another issue: a semantic transaction model focusing on flow control between tasks of different processes in the case of semantic failure (such as a change of business rules or cancellation of previously requested service provision) resulting in the necessity for backward process recovery.

It is well understood that the "classic" flat transaction model (although extremely successful in database environments) does not sufficiently reflect the processing patterns of complex, multi-system applications. The most fundamental drawback of transaction systems in the context of long-lived applications is their notion of transactions being concurrent and completely unrelated units of work. As a consequence, any existing interrelations between individual processes, like control flow and other semantic connections, cannot be implemented by the system, but have to be handled by the application.

An important step in the evolution of a basic transaction model was the extension from single-level transaction structure to multi-level structures. Solutions such as a nested transaction [10], open nested transactions [17], the Saga model [2], split-and-join transactions [11] , flexible transactions [12] , poly transaction [15], ConTract [16], all introduce various ways to handle long lasting transactions without the necessity of long held locks on relevant data resources. However, a fundamental problem with many extended and relaxed transaction models is that they provide a pre-defined set of properties that may or may not be required by the semantics of a particular activity.

Another problem with adopting these concepts for designing and implementing workflows is that the systems involved in the processing of a workflow may not provide support for facilities implied by an extended transaction model (e.g. no visible "prepare-to-commit" state). Furthermore, and most importantly, in order to support long-running computations the proposed models either assume that only forward recovery is supported after a crash (unacceptable limitations which implies human controlled backward recovery!) or that compensating transactions will be performed, which semantically undo what previously original transactions have done. Consequently, if the transaction later aborts, its failure atomicity may require that the effects of already committed subtransactions be undone by executing compensating subtransactions. Moreover, as in a multidatabase environment, relaxing the isolation property may cause violation of global consistency (global serializability) since other transactions may observe the effects of subtransactions that will be compensated later.

It seems to be hard, or often impossible, to provide fully automated compensation transactions. But as a matter of fact, compensation as a concept is a standard technique, if applied successfully (especially by a human). Since a workflow process is heavily dependent on the organizational structure and business policies within the organization's individual process, instances may require different semantic "undo", i.e., compensation activity. Thus the complete compensation should be performed by executing for each task instance its corresponding compensating activity (semantic undo). In this context we may define workflow management systems that provide the backward recovery feature as a systems that will automatically recover an aborted process to a state that is determined to be acceptable from a business perspective. Automatic backward recovery is of extreme importance from the user's perspective, as the manual solution to this problem is both very time consuming and labor intensive, with a correspondingly high cost. Providing backward recovery feature for workflow management systems is a goal of the FlowBack project.

The issue of a thorough design of compensation activities is a related problem to the one described above. We will not benefit much from a proper scheduling of compensation activities if we are not able to define properly the semantics of compensation actions. There is much work done so far in this field, including some internal research associated with the FlowBack project ([8]).

## 4. FLOWBACK

As stated in the previous section, the FlowBack project concentrates more on the process scheduling aspects, rather than the design of the compensation activities, and is an attempt to provide the workflow management systems with the feature of backward recovery by generating the compensation path and managing its execution.

From the user perspective, FlowBack is an exciting opportunity to:

- enhance the semantics of their business processes by including the definition of compensation for the individual steps of the process;
- make them more comprehensive and reliable;
- speed up the deployment process;
- address more business needs.

This can open up new domains within the workflow market that were formerly inaccessible due to the above inefficiencies.

## 4.1 Architectural considerations

From the design point of view we may consider several ways of dealing with the execution of a compensation plan: namely static, dynamic or semi-dynamic. The basic concept behind the static approach is as follows: knowing the model of the original process we automatically generate another process called the abort workflow model. This additional workflow process is instantiated and executed every time the user or process owner decides to abort the process. Its main goal is to compensate every activity that was executed during the runtime of the original process model. After the automatic generation of the abort workflow model, the user is allowed to make some modifications to it.

The dynamic approach is built on the assumption that it is possible to build the compensation path during the execution of the original model. When the original model is executing, the workflow system is building a compensation path. This compensation path is taken when the user decides to abort the original workflow. The dynamic approach is the most transparent solution to the user. The drawback of the dynamic approach is that the user is not allowed to make any modifications to the generated compensation plan, which can cause problems for certain applications.

The semi-dynamic solution is the combination of the former two. In the first step the static approach is taken and the static abort workflow model is generated. When the user aborts the original process, a simplified model that contains only activities needed for compensation is generated based on the static model and the log files. This simplified model is then executed.

## 4.2 FlowBack architecture

FlowBack is a module that can be put on top of any standards-compliant production workflow system. The solution we are proposing is vendor independent and is based on the architecture model as proposed by the Workflow Management Coalition [18] and supported by many leading vendors.

To achieve its goals FlowBack uses the static compensation generation plan model. It consists of two main modules: the FlowBack Reverser and FlowBack Compensator.

FlowBack Reverser is a module that takes the definition (*.FDL file) of a process model and generates the definition of a compensation plan. The output of this module is a *.FDL file that can be modified, approved and interpreted by a user. To be able to properly generate the compensation plan some enhancement to the original FlowMark process definition language is made. These additional constructs allow the user to specify which activities are to be compensated, what are the corresponding compensation activities, change the default user assignment, and finally define the way to compensate the nested constructs and loops in the original process model.

FlowBack Compensator is a module that allows a user to abort a process and invoke the compensation plan. By analysis of the workflow log (audit trail) the appropriate execution path of the compensation plan is taken.

## 4.3 Environment

FlowBack in its current version has been build on top of IBM FlowMark. It runs on OS/2 and Windows NT platforms and is written in Visual Age C++ and Microsoft Visual C++ respectively.

## 4.4 Demonstration Description

The demonstration is based on a process describing telecommunication provisioning. The original business process involves steps like creating the user data file, establishing physical and logical connections, initializing metering systems, setting up the directory entry for the client and finally sending the invoice to the client. In the demonstrated scenario, after performing a process half way through the client decides to resign from the service because of the relocation to another city. DSTC FlowBack is used to abort the original process and initiate the compensation activities like disconnecting the line, removing the client information from the directory and charging the customer for establishing a physical connection line to their previous location.

# 5. ACKNOWLEDGMENTS

The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science & Tourism.

# 6. REFERENCES

A full listing of the references for this paper can be found in

Kiepuszewski B, Muhlberger R, Orlowska M, FlowBack: Providing Backward Recovery for Workflow Management Systems, Distributed Systems Technology Centre, Technical Report, DSTC-TR-9840, 1998, http://www.dstc.edu.au/DDU/publications/tech-reports/TR-9840.ps