Chapter 6
Maintaining Transactional Integrity in Long Running Workflow Services: A Policy-Driven Framework

Stephan Reiff-Marganiec
University of Leicester, UK
Manar S. Ali
University of Leicester, UK

ABSTRACT
This chapter presents a framework to provide autonomous handling of long running transactions based on dependencies which are derived from the workflow. Business Processes naturally involve long running activities and require transactional behaviour across them. This framework presents a solution for forward recovery from errors by automatic application of compensation to executing instances of workflows. The mechanism is based on propagation of failures through a recursive hierarchical structure of transaction components (nodes and execution paths). The authors discuss a transaction management system that is implemented as a reactive system controller, where system components change their states based on rules in response to triggering of events, such as activation, failure, force-fail, completion, or compensation events. One notable feature of the model is the distinction of vital and non-vital components, allowing the process designer to express the cruciality of activities in the workflow with respect to the business logic. Another novel feature is that in addition to dependencies arising from the structure of the workflow, the approach also permits the workflow designer to specify additional dependencies which will also be enforced. Thus, the authors introduce new techniques and architectures supporting enterprise integration solutions that cater to the dynamics of business needs. The approach is implemented through workflow actions executed by services and allows management of faults through a policy-driven framework.

DOI: 10.4018/978-1-4666-4193-8.ch006
INTRODUCTION

Enterprise integration is significantly eased by the use of Service-driven architectural approaches as diverse systems can be encapsulated as Services which in turn can communicate readily through standard interfaces. This is further enhanced by the concept of executable (and possibly dynamic) Business Processes or Workflows which add a technical layer between the services and the business process as seen by a business analyst (Montangero, Reiff-Marganiec, & Semini, 2011; Gorton et al., 2009). However, enterprise integration is challenging and many of the challenges persist even in the Service-driven environment. One crucial aspect is that of ensuring correct transactional behaviour – both in the sense of not failing in states that are undesirable, and also in terms of attempting to complete the process (suitable concepts would be backward and forward recovery).

The database community has always strived to ensure consistency of data and allow for transactions to complete (possibly at a later stage or later attempt), or to be rolled back to a previous consistent state, and many solutions currently exist. It would seem straightforward to simply employ existing techniques; however, this is hampered by a number of new challenges arising in business processes. The two most crucial are the long running nature of business transactions and the delicate and complex nesting that naturally occurs. Database transactions usually complete within a matter of seconds and the most common solution for addressing the transactional integrity challenge involves some form of locking of resources. This is not applicable to business transactions, also often called Long running transactions (LRTs), these often span several days or even weeks (they typically involve humans making decisions such as approval of applications or time intensive operations such as shipping of physical goods) which clearly forbids any resource locking approach. Also, business processes often perform many actions in parallel or inside nested structures with complex control operators and this must be reflected in the transactions.

One of the important aspects in managing Long Running Transactions is in preserving consistency of the systems being involved in the LRT. This is done by guaranteeing that an LRT will always ensure that the data integrity is preserved and that systems are maintained consistently. This is made possible by ensuring that the execution of the LRT terminates in an accepted state from the business and the transaction modeling points of view. This will normally occur in the absence of a failure, but the same behaviour should manifest even if the LRT has not completed its normal path of execution due to a failure that causes termination of the LRT or diverts the execution to an abnormal path. This is usually achieved by adopting effective compensation and fault-handling techniques.

Workflows are usually composed out of workflow patterns (van der Aalst et al., 2000), and Bhiri, Perrin, and Godart (2006) have proposed transactional patterns to provide an understanding of the transactional consequences of workflow patterns. We have extended the concepts introduced in the transactional patterns to support multi-level nesting and we introduce the novel concept of vitality of actions, allowing the process designer to express the cruciality of activities in the workflow with respect to the business logic. We also present a framework based on propagation of failures through a recursive hierarchical structure of transaction components (nodes and execution paths). Our transaction management system COMPMOD is implemented as a reactive system controller, where system components change their states based on rules in response to triggering of events, such as activation, failure, force-fail, completion, or compensation events and policy rules enforce good transactional management at runtime.

We analysed a number of example business processes and derived the following list of aspects that we consider essential for a transaction