Applying Reo to Service Coordination in Long-Running Business Transactions*

Natallia Kokash and Farhad Arbab
CWI, Amsterdam, The Netherlands
firstName.lastName@cwi.nl

ABSTRACT

This paper presents an approach to formal modeling of long-running business transactions. Our solution is based on the channel-based exogenous coordination language Reo, which is an expressive, compositional and semantically precise design language that admits formal reasoning.

Keywords

Reo coordination, Transaction, Business Process Modeling

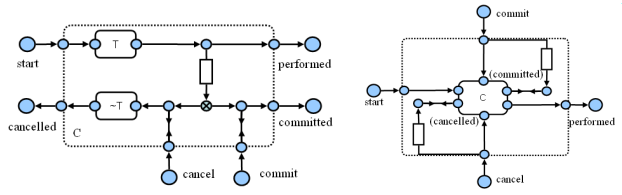
1. INTRODUCTION

Each business transaction in SOA can be kept open much longer than in traditional distributed database systems. The concept of Long Running Transactions (LRT) has been introduced to refer to such scenarios. Any changes performed during an LRT execution must be reverted back if a failure occurs somewhere in the flow of the transaction. Instead of the traditional rollback, data consistency and integrity in LRTs is preserved by executing *compensation* activities.

A number of formally grounded approaches have been proposed to specify LRTs[1]. However, none of these works considers full-featured LRT management as a specific, although rather arduous, case of service coordination. In this paper, we present an approach to formal modeling of LRTs, which relies on the channel-based exogenous coordination language Reo [1]. A graphical notation along with several formal semantic models have been defined for Reo. This makes it applicable both for graphical design and automated process verification. We envision Reo-based coordination in SOA as a bridge between domain-level design languages such as BPMN, and process execution languages such as WS-BPEL. In this way, we assume a-priori transactional behavior analysis which takes place before the system has been actually implemented. In our previous work [2], we define Reo connectors that simulate the behavior of the main BPMN modeling objects. By composing these connectors, one can model arbitrary complex process workflows.

[1]For an overview of related work please refer to a full version of this paper `http://www.dit.unitn.it/~kokash/documents/LRTInReo.pdf`

SAC'09 March 8-12, 2009, Honolulu, Hawaii, USA



(a) Internal view (b) Input/output hiding

Figure 1: Compensation pair

2. USING REO FOR LRT MODELING

An atomic task T with an associated compensation activity $\sim T$, written as $T \div \sim T$, can be modeled as shown in Fig. 1(a). In this Reo circuit, after the task T has executed, a token flows into the FIFO1 channel and enables the connector to accept cancel or commit messages. If a cancel message arrives, the compensation activity $\sim T$ is executed and the task T is considered to be canceled. If a commit message arrives, the status of the task changes to "committed" and the task effects cannot be undone anymore.

Figure 2 shows a Reo model for a sequential transaction $P \equiv C_1; C_2; ...; C_n$. Here, each connector $C_i \equiv T_i \div \sim T_i, 1 \leq i \leq n$, stands for an aforementioned compensation pair with hidden outputs representing the "canceled" and "committed" states (see Fig. 1(b)). At the end of the successful transaction execution a token is back-propagated to commit all performed activities. If, instead, a cancel message has been received, it is picked up at a place where the execution token currently resides and back-propagated to cancel all performed activities. By using FIFO1 channels instead of synchronous channels going to the "cancel" port of each compensation pair $C_i, 1 \leq i \leq n$, each compensation activity can be activated independently.

In both variants of the above circuit the compensation activities are performed concurrently. However, a process may require ordered execution of compensation activities. Fig. 3 shows a transaction in which the effects of the tasks are compensated for in the reverse order with respect to the normal flow order. A cancel message is sent to the "cancel" port of the circuit C_{i-1} only if the circuit C_i produces an

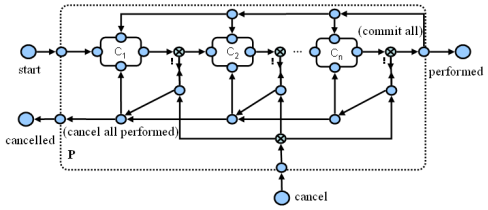


Figure 2: Sequential transaction

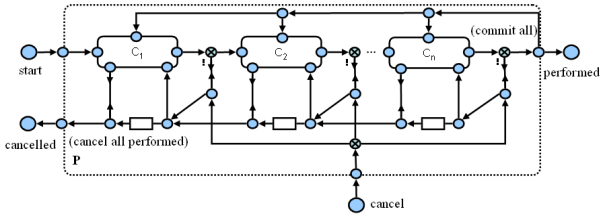


Figure 3: Sequential transaction with compensation occurring in reverse order

output signalling that its task has been compensated for.

Suppose now that compensation activities may fail, that is, the task $\sim T$ has two possible outcomes, namely, successful completion, which means that the effects of the task T have been completely canceled, and exception or hazard, which signals that something went wrong while canceling the effects of the source task. Figure 4 models a transaction process consisting of a set of sequentially executed activities with a possible hazard output. We need to ensure that all completed activities have been successfully canceled. This involves a structure similar to the one for executing and canceling parallel activities [2]. First, all performed compensation pair connectors receive cancel messages. Second, messages confirming the successful execution of all compensation activities must be received. If instead any of the compensation activities fail, we can signal the hazard event and clean up tokens returned by each of the invoked compensation activities. To accomplish this task, the exclusive router Y redirects the exception token to one of the places $y_i, 1 \leq i \leq n$, where the cancelation token currently resides, and both are disposed of in the corresponding synchronous drain (y_i, z_i). Additionally, tokens flow from this point into all available FIFO1 channels and wait until all compensation activities have disposed their tokens, either through the cancel output or through the exception output.

A Reo circuit modeling a parallel process $P \equiv C_1|C_2|...|C_n$ is essentially composed of a parallel fork and a parallel join gateways with n outgoing and n incoming branches [2]. When an activity $T_i, 1 \leq i \leq n$, has completed, the corresponding token waits until other activities complete as well. After that, the token flows to the circuit output. For interrupting the process, a cancel message, either coming from an external source, or spawned by a failed activity in one of the parallel flows, is asynchronously directed to each of the remaining branches. A similar Reo connector can be used to cancel parallel activities within an LRT. Additionally to the aforementioned pattern, we must commit each activity after all branches have completed successfully.

Another interesting patterns involving parallel activities

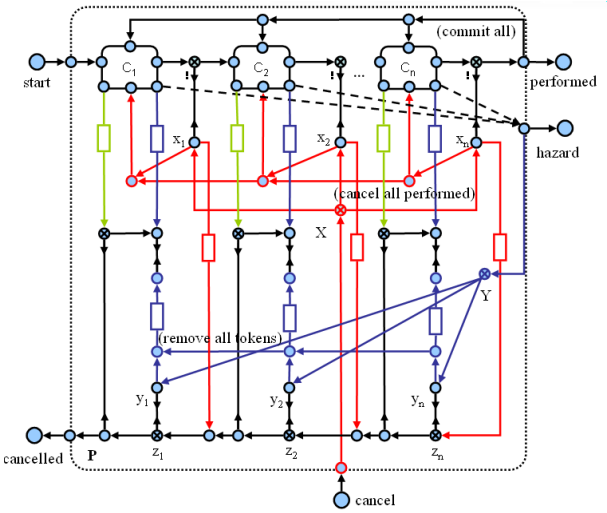


Figure 4: Sequential transaction with possible failures in compensation activities

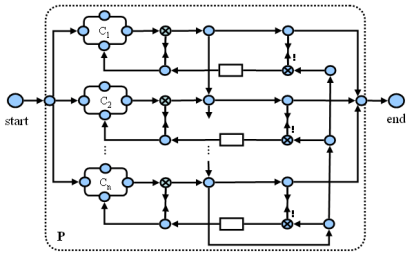


Figure 5: Transaction with competing parallel flows

is a *discriminator choice* which allows alternatives to be explored in parallel. Once one branch finishes successfully, all the remaining alternatives are stopped and compensated for. A Reo circuit modeling such a behavior is shown in Fig. 5. The first completed branch initiates the compensation for all other branches. The compensation is performed asynchronously when the corresponding compensation pair connector is ready to accept the cancel message.

3. CONCLUSIONS

In this paper, we proposed a Reo-based approach to formal LRT modeling. Our approach enables the specification of complex compensation handling scenarios using a small number of modeling primitives. The fact that Reo provides a comprehensive visual notation for its channels, makes it easy to use by software designers without any experience in process calculi.

4. REFERENCES

[1] F. Arbab. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.

[2] F. Arbab, N. Kokash, and M. Sun. Towards using Reo for compliance-aware business process modelling. In *Proc. of the Int. Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, volume 17 of *LNCS*. Springer, 2008.