

# White Paper

## Modeling Transactions with BPMN 2.0

WP0077 | May 2013



### Gregor Polančič

Gregor is an assistant professor at the University of Maribor and has a decade of experience in BPMN since its first version in 2004.

He participated in the development of one of the first BPMN modeling utilities - a package of plugins for Visio, which were introduced in 2004 and is the main author of the first BPMN poster.

In 2008, he was one of the first authors who published an article dedicated to the experiences and practical use of BPMN. The article was published in "BPM and Workflow Handbook" in association with the Workflow Management Coalition (WfMC).

He is currently researching BPMN from different technological and user aspects.

**IT experts usually associate the term transaction with distributed database systems operations. In this context, transactions play a crucial role to maintain the integrity of data. A single database transaction consists of several independent units of work, each reading or writing information to a database. When a transaction is finished it is important to ensure that the corresponding units of work leave the database in a consistent state.**

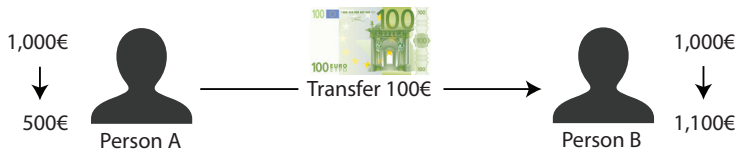
Similar to databases, a transaction can also occur in business processes. A business process consists of a series of tasks which are performed in a predefined order. In some cases, the tasks are interrelated in a way which requires successful execution of several tasks in order to fulfill a business objective. In the opposite case (if only some of the required tasks are executed) a serious problem might occur.

The difference between database transactions and business process transactions is in the corresponding implementations. Database transactions are usually implemented with "two-phase commit protocol" which requires almost immediate execution of transactional activities. On the other hand, a business process can be executed over several days or even months and this requires another transactional handling. This article will focus on business transactions and their representation in BPMN 2.0.

Access our **free**, extensive library at  
[www.orbussoftware.com/downloads](http://www.orbussoftware.com/downloads)

# What is a Transaction?

Let us explain a transaction in the following example in which a person intends to transfer money to another person (perhaps at another bank) by using a bank transfer. In case both persons possess 1,000€ before the bank transfer, Person 'A' would possess 900€ and Person 'B' would possess 1,100€ at the end of successful 100€ bank transfer (*Figure 1*).



**Figure 1: External view of a bank transfer**

At first sight, the bank transfer looks like a simple task, however the internal view of a bank transfer is more complex (*Figure 2*). When money is transferred between two accounts (e.g. between two banks with two distinct information systems), two interrelated activities

occur. One activity withdraws money from 'Person A' account, where another activity deposits money into 'Person B' account. Besides, the balances of both accounts need to be updated.

The negative scenario of the bank transfer occurs if any of these activities are unsuccessful:

- If money is withdrawn from 'Person A' account where the money isn't deposited into 'Person B' account.
- If money is deposited into 'Person B' account where no money is withdrawn from 'Person A' account.



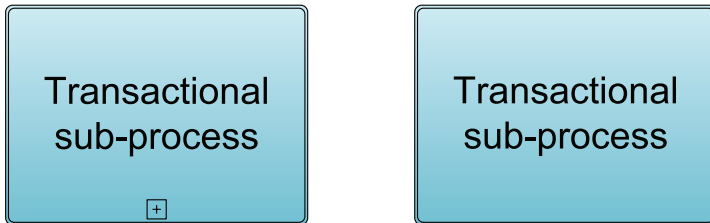
$$1,000\text{€} - 100\text{€} = 900\text{€} \quad 1,000\text{€} + 100\text{€} = 1,100\text{€}$$

**Figure 2: Internal view of a bank transfer**

In both negative scenarios the resulting balances of bank accounts are incorrect meaning that money transfer between accounts can be correctly completed only when the withdrawal, deposit, and balance update are done consecutively and all of them are executed without any errors. Such "a series of tasks that are meaningful only when all of the tasks are completed appropriately" are called a transaction. A transaction is defined by an execution rule that sets the execution results of tasks in a transaction to be, as a whole, whether "all tasks are executed" or "none of them is executed." The tasks are tentatively

executed first, and if all the tasks are successfully completed, the process continues. Otherwise, all of them are undone and started over again.

Two phase commit protocol works following: First phase checks if all IT systems can perform required transactional activities. If yes, the activities are performed (second phase) otherwise none of the activities is performed.



**Figure 3: Collapsed (left) and expanded (right) transactional sub-process**

## BPMN 2.0 Specification defines a *Transaction* as the following:

“A sub-process that represents a set of coordinated activities carried out by independent, loosely-coupled systems in accordance with a contractually defined business relationship. This coordination leads to an agreed, consistent, and verifiable outcome across all participants.”

## Transactions in BPMN 2.0

BPMN provides built-in support for business transactions. A business transaction differs from a database transaction, because its activities are not locked while the transaction is in progress. Instead each activity in the transaction executes normally in its turn, but if the transaction as a whole fails to complete successfully, each of its activities that have already completed are undone by calling a so-called compensating activity.

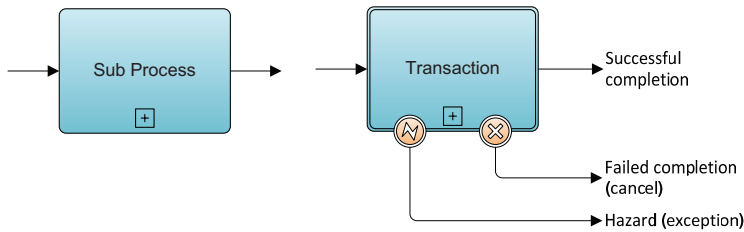
BPMN transactions are represented as special sub-processes, either collapsed or expanded. A transactional sub-process (transaction) visually differs from a normal sub-process (sub-process), because the borders consist of double lines.

Modeling a BPMN transaction differs from modeling a sub-process in the relationships to other process elements as well as the interior.

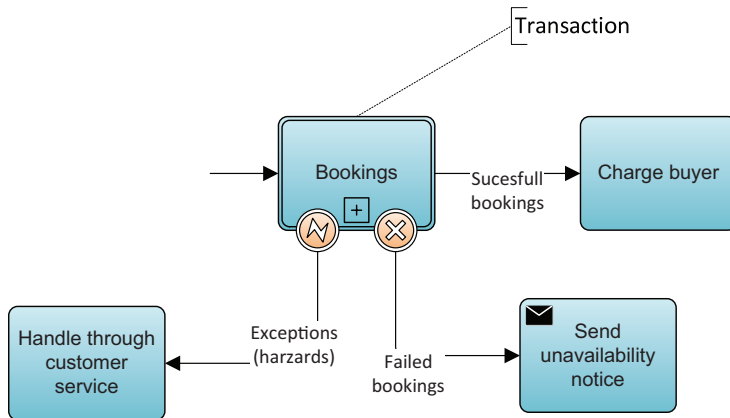
## External view

A transaction is a special sub-process, which is always part of a higher level process or sub-process. Similar to a sub-process, a transaction is “triggered” by a token which arrives at the incoming sequence flow. However, on the resulting side, a transaction differs from a normal sub-process since it can result in three outcomes (*Figure 4*):

- **Successful completion.** It is modeled with a normal sequence flow that leaves the transaction. In this case, all tasks in a transaction are completed successfully, meaning that the transaction and the corresponding process will proceed normally.
- **Failed (unsuccessful) completion (Cancel).** Failed completion is modeled with a sequence flow, which starts at a cancel intermediate event, attached to the boundary of a transaction. This scenario occurs if any of the pre-determined criteria of failure of the transaction is satisfied or in case an ‘Abort’ message is received from outside of the transaction. In both cases the non-normal flow is executed instead of the normal flow, meaning that the normal business process is not executed, and none of the tasks in the transaction are completed.
- **Hazard (Exception).** Hazard is modeled with a sequence flow, which starts at an error intermediate event, attached to the boundary of a transaction. A hazard means that something went terribly wrong and



**Figure 4: Sub-process's flows (left) and transaction's flows (right)**



**Figure 5: Booking example modeled as "collapsed" transaction**

that a normal success or cancel is not possible. When a hazard occurs, any of the tasks in the transaction end up not being executed or compensated.









The behavior of the successful completion at the end of a transaction is different than that of a normal sub-process. When each path of the transaction reaches the end event, the flow does not immediately move back up to the higher-level parent process, as does a normal sub-process. First, the transaction protocol needs to verify that all the participants have successfully completed their end of the transaction. In other cases, a cancel or exception flow will be triggered. Figure 5 represents a partial BPMN bookings diagram, where the 'bookings' sub-process is modeled as a transaction.

The diagram shows that a buyer is charged only if all bookings (represented implicitly in

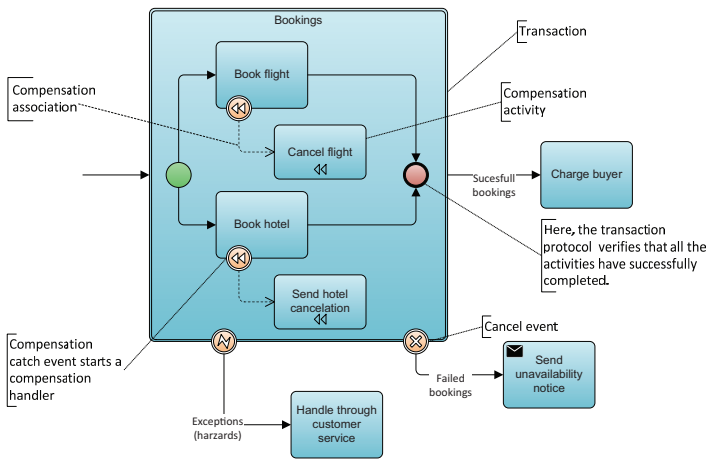
a collapsed transaction) are successful. In case the transaction fails, the unavailability message will be sent, where in the case of an exception, the issue will be handled through customer service.

# Internal View

From the internal view, a transaction differs from a normal sub-process, because it has to include additional transaction-specific elements (Table 1).

BPMN 2.0 element	Graphical representation	BPMN 2.0 specification based explanation
Transactional sub-process		A transaction is a sub-process that is supported by a special protocol that ensures that all parties involved have complete agreement that the activity should be completed or cancelled. The attributes of the activity will determine if the activity is a transaction.
Compensation activity		The compensation activity, which can be either a task or a sub-process, has a marker to show that it is used for compensation only and is outside the normal flow of the process.
Intermediate non-interrupting boundary cancel event		This type of event is triggered if a cancel end event is reached within the transaction sub-process. It can also be triggered if a transaction protocol "cancel" message has been received while the transaction is being performed. A cancel event always interrupts the activity to which it is attached.
End cancel event		This type of end event is used within a transaction. It will indicate that the transaction should be cancelled and will trigger a cancel intermediate event attached to the sub-process boundary. In addition, it will indicate that a transaction protocol cancel message should be sent to any entities involved in the transaction.
Non interrupting sub-process compensation start event		This type of start event is only allowed for triggering an in-line compensation event sub-process when a compensation occurs. It does not interrupt the process since the process has to be completed before this event can be triggered.
Intermediate non-interrupting boundary compensation event		The Event will be triggered by a thrown compensation targeting that Activity. When the Event is triggered, the Compensation Activity that is associated to the Event will be performed. Compensations can only be triggered after completion of the Activity to which they are attached. Thus they cannot interrupt the Activity.
Intermediate throwing compensation event		In normal flow, this intermediate event indicates that compensation is necessary. If an activity is identified, and it was successfully completed, then that activity will be compensated. The activity must be 'visible' from the compensation intermediate event.
End compensation event		This type of end indicates that compensation is necessary. If an activity is identified, and it was successfully completed, then that activity will be compensated. The activity must be 'visible' from the compensation end event.

**Table 1: BPMN 2.0 transaction specific elements**



**Figure 6: Booking example modeled as “expanded” transaction**

The following diagram (Figure 6) represents a common modeling approach of a transaction. It represents the internal view of the booking example (Figure 5).

While a transaction is a sub-process it consists of several activities (e.g. book hotel and book flight). Beside these activities, which represent a normal flow, a transaction includes special ‘compensation activities’ (‘cancel flight’ and ‘send hotel cancelation’). These activities are not part of a normal flow and represent transaction’s compensation. A compensation activity is connected to a corresponding normal-flow activity with a compensation association (dotted arrows). A compensation association occurs outside the normal flow

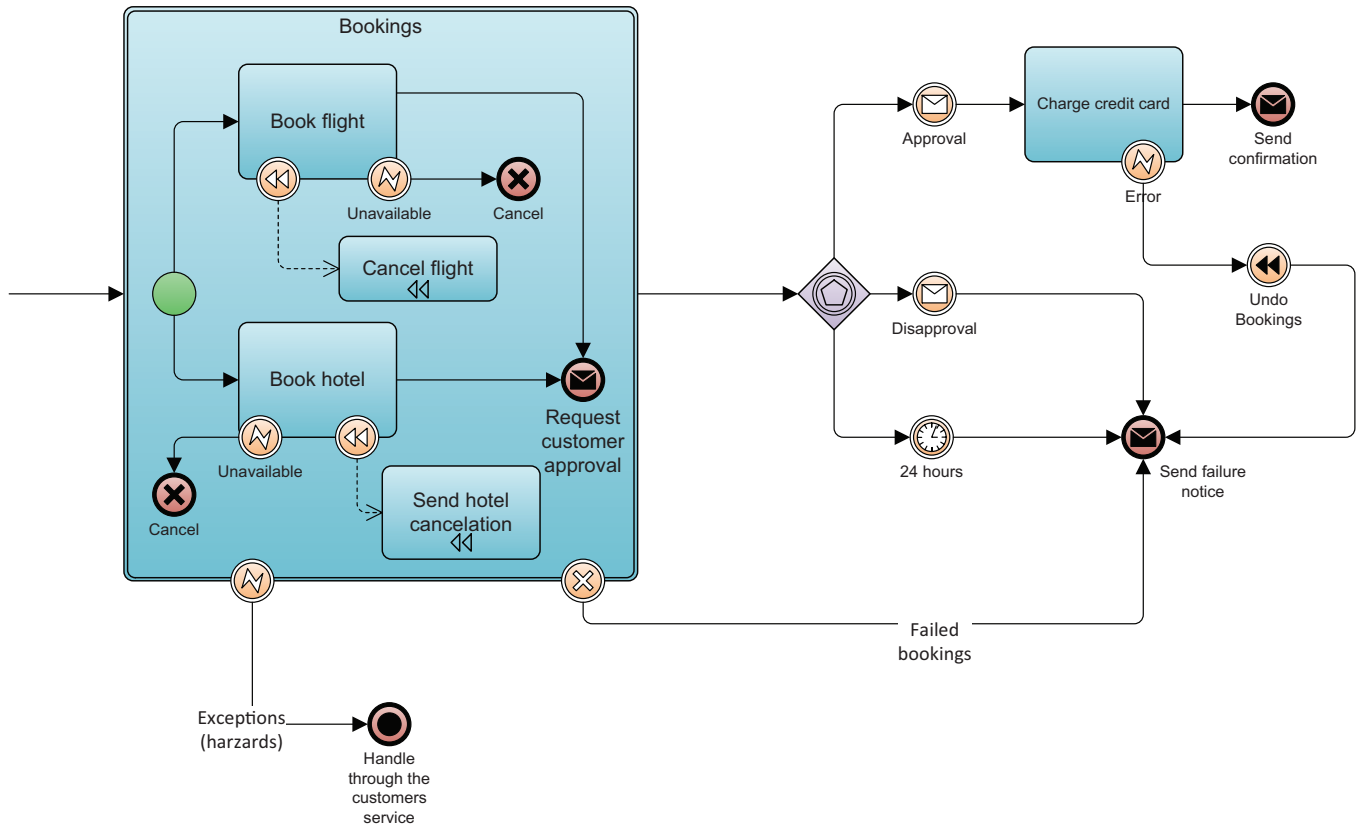
of the process and is based upon a compensation intermediate event that is triggered through the failure of a transaction or a throw compensation event.

The term ‘compensation’ is concerned with undoing steps that were already successfully completed, because their results and possibly side effects are no longer desired and need to be reversed. If an activity is still active, it cannot be compensated, but rather needs to be cancelled.

Cancellation in turn can result in compensation of already successfully completed portions of an active activity, in case of a sub-process. Compensation is performed by a compensation handler.

A compensation handler is a set of activities that are not connected to other portions of the BPMN model. The compensation handler starts with a catch compensation event. That catch compensation event is either a boundary event, or, in case of a compensation event sub-process, the handler’s start event. A compensation handler performs the steps necessary to reverse the effects of an activity. In case of a sub-process, the compensation handler has access to sub-process data at the time of its completion (“snapshot data”). Compensation is triggered by a throw compensation event, which typically will be raised by an error handler, as part of cancellation, or recursively by another compensation handler. That event specifies the activity for which compensation is to be performed, either explicitly or implicitly.

The next example diagram builds on the previous one. It includes a mechanism that enables the customer to cancel the transaction after it has been successfully completed. In this case, cancel event cannot be used to undo the transaction, because cancel throw must come from the interior of the transactional sub-process.



**Figure 7: Throwing compensation event**

The diagram in *Figure 7* resolves this by using a throwing compensating event ('undo bookings'), which can be positioned outside the transaction. The throwing compensation event triggers all of the compensating activities within the transactional sub-process.

## Conclusion

Business process transactions are important since it is common that process's activities are interrelated in a way, which requires comprehensive and consistent execution. BPMN provides built-in support for modeling business transactions, which are represented as a sub-process with double line borders. The main difference between a sub-process and a transaction is that the relationships among tasks in a transaction are very strong and constraints regarding their execution are placed on the tasks while a sub-process is merely a set of related tasks.

Despite of the importance of business transactions, modelers commonly avoid using them, since they require advance modeling skills – special transactional elements as well patterns of their use. In this case, a simplified, two stage approach, can be used. In the first stage, a modeler can represent a transaction as a “black-box”, since this representation requires only a fraction of required skills. When collapsed transactions are mastered, he or she can start focusing on the internal view of a transaction, which requires the knowledge of the remaining set of transactional elements as well as patterns of their use.

## References

- [1] Business process library, available at:  
[http://en.q-bpm.org/mediawiki/index.php/Transaction\\_\(BPMN\)](http://en.q-bpm.org/mediawiki/index.php/Transaction_(BPMN))
- [2] BPMN 2.0 specification, available at:  
<http://www.omg.org/spec/BPMN/2.0/>
- [3] B. Silver, BPMN method and style. Aptos  
Calif.: Cody-Cassidy Press, 2009.

© Copyright 2013 Orbus Software. All rights reserved.

No part of this publication may be reproduced, resold, stored in a retrieval system, or distributed in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission of the copyright owner.

Such requests for permission or any other comments relating to the material contained in this document may be submitted to: [marketing@orbussoftware.com](mailto:marketing@orbussoftware.com)

**Orbus Software**

3rd Floor  
111 Buckingham Palace Road  
London  
SW1W 0SR  
United Kingdom

+44 (0) 870 991 1851  
[enquiries@orbussoftware.com](mailto:enquiries@orbussoftware.com)  
[www.orbussoftware.com](http://www.orbussoftware.com)

