# Merging Business Process Models

Marcello La Rosa[1], Marlon Dumas[2], Reina Uba[2], and Remco Dijkman[3]

[1] Queensland University of Technology, Australia
m.larosa@qut.edu.au
[2] University of Tartu, Estonia
{marlon.dumas,reinak}@ut.ee
[3] Eindhoven University of Technology, The Netherlands
r.m.dijkman@tue.nl

**Abstract.** This paper addresses the following problem: given two business process models, create a process model that is the union of the process models given as input. In other words, the behavior of the produced process model should encompass that of the input models. The paper describes an algorithm that produces a single configurable process model from a pair of process models. The algorithm works by extracting the common parts of the input process models, creating a single copy of them, and appending the differences as branches of configurable connectors. This way, the merged process model is kept as small as possible, while still capturing all the behavior of the input models. Moreover, analysts are able to trace back which model(s) a given element in the merged model originates from. The algorithm has been prototyped and tested against process models taken from several application domains.

## 1 Introduction

In the context of company mergers and restructurings, it often occurs that multiple alternative processes, previously belonging to different companies or units, need to be consolidated into a single one in order to eliminate redundancies and create synergies. To this end, teams of business analysts need to compare similar process models so as to identify commonalities and differences, and to create integrated process models that can be used to drive the process consolidation effort. This process model merging effort is tedious, time-consuming and error-prone. In one instance reported in this paper, it took a team of three analysts 130 man-hours to merge 25% of two variants of an end-to-end process model.

In this paper, we consider the problem of (semi-)automatically merging process models under the following requirements:

1. The behavior of the merged model should subsume that of the input models.
2. Given an element in the merged process model, analysts should be able to trace back from which process model(s) the element in question originates.
3. One should be able to derive the input process models from the merged one.

The main contribution of the paper is an algorithm that takes as input a collection of process models and generates a *configurable process model* [15]. A

R. Meersman et al. (Eds.): OTM 2010, Part I, LNCS 6426, pp. 96–113, 2010.

configurable process model is a modeling artifact that captures a family of process models in an integrated manner and that allows analysts to understand what these process models share, what their differences are, and why and how these differences occur. Given a configurable process model, analysts can derive individual members of the underlying process family by means of a procedure known as *individualization*. We contend that configurable process models are a suitable output for a process merging algorithm, because they provide a mechanism to fulfill the second and third requirements outlined above. Moreover, they can be used to derive new process models that were not available in the originating process family, e.g. when the need to capture new business procedures arises. In this respect, the merged model can be seen as a reference model [4] for the given process family.

The algorithm requires as input a mapping that defines which elements from one process model correspond to which elements from another process model. To assist in the construction of this mapping, a mapping is suggested to the user who can then adapt the mapping if necessary.The algorithm has been tested on process models sourced from different domains. The tests show that the process merging algorithm produces compact models and scales up to process models containing hundreds of nodes.

The paper is structured as follows. Section 2 introduces the notion of configurable process model as well as a technique for proposing an initial mapping between similar process model elements. Section 3 presents the process merging algorithm. Section 4 reports on the implementation and evaluation of the algorithm. Finally, Section 5 discusses related work and Section 6 draws conclusions.

## 2 Background

This section introduces two basic ingredients of the proposed process merging technique: a notation for configurable process models and a technique to match the elements of a given pair of process models. This latter technique is used to assist users in determining which pairs of process model elements should be considered as equivalent when merging.

### 2.1 Configurable Business Processes

There exist many notations to represent business processes, such as Event-driven Process Chains (EPC), UML Activity Diagrams (UML ADs) and the Business Process Modeling Notation (BPMN). In this paper we abstract from any specific notation and represent a business process model as a directed graph with labeled nodes as per the following definition. This process abstraction allows us to merge process models defined in different notations.

**Definition 1 (Business Process Graph).** *A* business process graph $G$ *is a set of pairs of process model nodes—each pair denoting a directed edge. A node* $n$ *of* $G$ *is a tuple* $(id_G(n), \lambda_G(n), \tau_G(n))$ *consisting of a unique identifier* $id_G(n)$ *(of type string), a label* $\lambda_G(n)$ *(of type string), and a type* $\tau_G(n)$*. In situations where there is no ambiguity, we will drop the subscript* $G$ *from* $id_G$*,* $\lambda_G$ *and* $\tau_G$*.*