Information Systems 72 (2017) 77–94

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Designing business capability-aware configurable process models

Wassim Derguech [a,*], Sami Bhiri [b], Edward Curry [a]

[a] Insight Centre for Data Analytics, National University of Ireland, Galway (NUIG), Ireland
[b] OASIS, National Engineering School of Tunis, University Tunis El Manar, Tunisia

ARTICLE INFO

Article history:
Received 2 June 2017
Revised 2 October 2017
Accepted 3 October 2017
Available online 12 October 2017

Keywords:
Process aware information systems
Business process modeling
Configurable business process modeling
Merging business process models
Business capability

ABSTRACT

Process Aware Information Systems manage processes within organisations on the basis of business process models. These models can be created either from scratch or by reusing exiting reference process models.

Particular types of reference models are configurable process models that are created by merging multiple models into a single one that can be customized to the needs of the business experts. Using those models presents two main challenges: their creation and their configuration.

In this paper, we focus on the first challenge and propose a novel algorithm for merging process models into a configurable process model. The difference in our work is the pre-annotated process models with their business capabilities that report on what actions each process element achieves. Our algorithm generates configurable models that are also annotated with their capabilities that can be used to face the second challenge of these models: the configuration phase.

We tested our algorithm using real-world process models to evaluate the required creation time and resulting compression rate after merging the input models. The results show that the models can be created in few milliseconds and achieving a compression rate of 50%. We further carried out interviews with domain experts to assess the usefulness and the level of maturity of this work. The results show the importance of the automation of process merging using a tool support that we proposed. However, further adaptation efforts are required to integrate this work in the working environments of the interviewed experts.

1. Introduction

Reference process models describe proven practices for a specific industry. They are often aligned with emerging industry-specific and cross-industry standards [1,2]. One of the scenarios of use of reference process modelling is the reference process model customization [3]. It begins with a reference process model that provides configuration facilities. This model can be configured to specific needs of an enterprise e.g., by refining business rules or enabling/disabling some activities. Such reference models are called configurable business process models [4]. It is a reference model that can be tailored by end-users in order to meet their requirements and satisfy their business needs [4]. The management of such models, brings various challenges for their creation and configuration.

The basis of a configurable business process model is the integration of multiple behaviours of business processes into a single model. These behaviours are captured in *various* business process models that are called *business process variants* [4]. Configurable process models are constructed either via mining techniques [5,6] or the manual or automated merging/aggregation of several variants of a process model [4,7–10]. Manual creation of configurable process models is tedious, time-consuming and error-prone task. It requires the identification of common process parts, merging them and explicitly representing differences between models in terms of configuration options. The literature provides several approaches to overcome this challenge [5,9,11], the main issue with such approaches is that the resulting configurable models capture their configuration options in terms of model restrictions that are difficult to manipulate by end-users during the configuration phase.

The configuration of these reference models consists of enabling/disabling several branches of the model through manipulating configuration options [12]. This phase is difficult and requires advanced modelling skills for identifying and selecting the configuration options. Furthermore, the users cannot determine the impact (i.e., what functionality are they enabling or disabling from the configurable model) of each configuration decision they take unless they manually trace each branch of the configurable node and determine the functionality resulting from each of them. This

* Corresponding author.
E-mail addresses: wassim.derguech@insight-centre.org (W. Derguech), sami.bhiri@gmail.com (S. Bhiri), edward.curry@insight-centre.org (E. Curry).

https://doi.org/10.1016/j.is.2017.10.001
0306-4379/

can be resolved by creating an explicit link between the model configurations and the domain requirements and lifting the configuration phase from manipulating model restrictions to domain requirements. La Rosa [12] proposed to model domain requirements as a set of questions with answers explicitly linked to configuration options. In this case, the configuration phase consists on answering these domain related questions. Even though this solution helps in guiding the configuration, it requires a lot of manual work for creating these questions and linking them to the model restrictions.

The contribution of this paper is an algorithm that allows merging a pair of business capability-annotated process variants given as input and delivers a business capability-annotated configurable process model. Several methods have been proposed to merge business process variants [7–10], however, their main weakness resides in the fact that they do not consider tasks capabilities for matching business process tasks. They rely exclusively on the task labels for this operation. In contrast to existing proposals, this paper uses capabilities for matching similar tasks in different models. The resulting configurable model is also annotated with capabilities which facilitates the configuration and individualization steps [4,12,13].

In order to carry out a quantitative evaluation of the merging algorithm proposed in this paper, two main metrics are considered: *time* required for merging business process models and the *compression rate* gained after the merging operation. These two metrics have been used by La Rosa et al. [9] for evaluating their business process merging algorithm.

- *Time*: for organisations, time is important and should not be spent on manual creation of configurable models. La Rosa et al. [9] mentioned that it took a team of five analysts and 130 man-hour to merge *manually* 25% of an end-to-end process model. Therefore, an automation support for merging business process variants is needed to help saving time and money.
- *Compression rate*: the compression of a repository of business process variants into a single configurable model has multiple benefits: guaranteeing consistency between business process models, avoiding business process clones [14], etc.

This paper evaluates also the proposed algorithm with respect to a set of requirements that the have been used previously in the literature:

1. *[Behaviour Subsumption]* The merged model should allow for the behaviour of all the original models. Traditionally, the merging operation is manually made by business analysts which comes with the risk that some aspects of the original models are accidentally neglected [7]. With automation support for merging process variants, this risk can be minimized considerably.
2. *[Traceability]* Each element of the merged process model should be easily traced back to its original model [9,10]. A business analyst needs to understand what the process variants share, what are their differences, etc. This can be made possible if they can trace back to the variant from which an element originates.
3. *[Deriving Original Models]* Business analysts should be able to derive the input models from the merged process model [9,10].

The remainder of this paper is organized as follows: Section 2 further describes the concept of configurable business process models and introduces the formal definition of a capability-annotated configurable business process model. Section 3 introduces a running example that will be used in the rest of the paper. Section 4 presents the merging algorithm. Section 5 reports on the implementation and validation of the algorithm. Section 6 analyzes the related work and Section 7 concludes the paper and discusses future research directions.

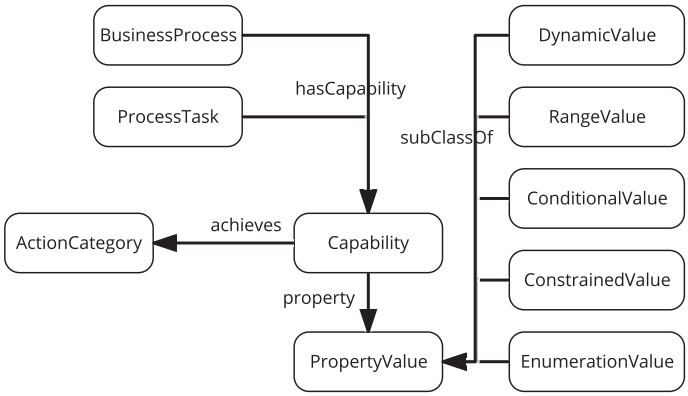


Fig. 1. Snippet of the business capability meta-model.

2. Basic concepts

2.1. Business capability

An important concept used in this paper is the *Business Capability*. It has been defined in the literature from various perspectives:

- From an organizational and resource perspective: Organisational Capability: the ability of organizations to efficiently use their resources (i.e, human capital, knowledge, available data, etc.) to generate value and achieve their objectives [15,16].
- From a control flow perspective: Planning Capability : the way organizations achieve their goals by capturing explicitly process tasks and their temporal and logical order [17].
- From a service perspective: IT Capability: the effect of a service in terms of data generated or change of the world [18] that are explicitly represented in terms of Inputs, Outputs, Preconditions and Effects (IOPE for short).
- From a functional perspective: Business Capability: the action performed by a service, computer program, etc. that creates a value for the customers [19].

In this paper, we consider the business capability from a functional perspective. We argue that this concept is highly required for describing what is being achieved by enterprise services, business processes and tasks. As depicted in Fig. 1, we propose to model a business capability as an action category enriched by (zero or many) functional or non-functional properties. These properties refine the given capability by giving more details about aspects of interest of the corresponding action.

More formally, in the proposed model, capabilities are defined as a Category and a set of property entries (see Definition 1). A property entry is a couple (property, value) where *property* is a domain-specific functional feature or a domain-independent non-functional property and *value* is the value or the possible values that a property can have. Both *property* and *value* refer to ontological terms.

Definition 1 (Business Capability)**.** A couple *Cap = (Category,Properties)* is a business capability, where:

- *Category*: This concept is similar to [19] that defines, in a natural language, what is the action being described. Different to [19], we consider the category as a concept from a domain related ontology that comes form a shared agreement on its semantics. A category is a specific property that is present in all business capability descriptions via the property *achieves* (see Fig. 1).
- *Properties*: Represents a set of pairs *(Property, Value)* that correspond to the set of features of the business capability.

Details about the capability modeling approach that we are adopting in this paper can be found in prior contributions [20,21].

2.2. Capability annotated process model

In the area of business process modelling various efforts, either from industry or academia, have been put towards proposing modelling languages such as Event-driven Process Chains (EPC), Business Process Modelling Notation (BPMN), and Unified Modelling Language Activity Diagram (UML AD). This paper aims to abstract from any of these notations to model business processes without any ties to existing business process modelling languages. The advantage of this abstraction is to make its contributions easily applicable to other modelling languages. Actually, a business process model is presented as a directed graph and formally described in Definition 2.

Definition 2 (Capability Annotated Process Graph)**.** *A Capability Annotated Process Graph* is a directed graph $G =< N, C, A, T, Cap, Cond >$, where N is a set of work nodes including *InitialNode, FinalNode* and *IntermediateNode* that are both event and activity nodes; C is a set of graph connectors: i.e., ANDsplit, ANDjoin, ORsplit, ORjoin, XORsplit, and XORjoin and A is a set of directed arcs for interconnecting all the graph nodes. T is a type function, it associates with each node its respective type (i.e., a string to indicate: activity, event, XorSplit, etc.). *Cap* is an annotation function that associates with each activity node n a tuple Cap(n) = (ActionCategory (n), Properties(n)). *Cond* is an annotation function that associated with each event node n a condition c (i.e., Cond(n)=c).

For a Capability Annotated Process Graph G, its set of work nodes is denoted N_G. Each work node n in N_G has a type depending on the modelling language being considered. For example, in BPMN and EPC there are two types of nodes: Events and Functions. In N_G there are two particular nodes: *InitialNode* and *FinalNode* for marking the beginning and the end of the process. These nodes have dedicated graphical representations in BPMN specification [22] while in EPC these are regular events without incoming arcs for the *InitialNode* and without outgoing arcs for the *FinalNode*.

Connectors, also known as routing nodes, of a Process Graph are denoted C_G. Each c in C_G can be either a split or join connector. Split connectors have a single input arc and multiple output arcs while join connectors have multiple input arcs and a single output arc. A split connector indicates that (i) the flow of activities continues into multiple parallel branches (i.e., in case of an ANDSplit), (ii) a choice has to be made towards one possible active branch (i.e, in case of a XORSplit) or (iii) multiple branches can be activated (i.e, in case of an ORplit) after this node. A join connector indicates that the process has to wait until (i) all the branches (i.e., in case of an ANDJoin), (ii) exactly one branch (i.e, in case of a XORJoin) or (iii) multiple branches (i.e, in case of an ORJoin) are activated before this node.

Each of the nodes of an Annotated Process Graph are interconnected with directed arcs denoted A_G. These arcs define either causal or temporal relations between these nodes. Syntactic restrictions on possible arcs between nodes can be imposed based on the modelling language. For example, in EPC, arcs cannot exist between two functions or events.

In order to get the capability of an annotated process graph, the function Cap_G is used. It associates for each activity node (e.g., function node in EPC) its capability (see Definition 1).

2.3. Capability annotated configurable process model

This paper uses EPC notation for illustrating basic processes and C-EPCs [4,23] for configurable processes. C-EPC stands for Configurable EPC. It is an extended version of EPC where some *connectors* can be marked as configurable. A configurable connector can be configured by reducing its incoming branches (in the case of a join) or its outgoing branches (in the case of a split) [10]. The result will be a regular connector with a reduced number of incoming or outgoing branches. Functions and events can also be configured by adjusting their labels. Additionally, functions can be set to enabled, skipped or conditionally skipped. This paper adds another configuration dimension to function nodes based on their capabilities. The capability of a function can be adjusted by adding, removing or changing any of its properties with respect to the capability domain ontology.

Recall, this paper's contribution is a merging algorithm that takes as input a set of capability-annotated process models and generates a capability-annotated configurable process model. Input models are formally presented as directed graphs that were formally described in Definition 2 [Capability-Annotated Process Graph]. On top of this definition, Definition 3 formally describes a capability-annotated configurable process graph that is used to formally describe the output of the proposed merging algorithm.

Definition 3 (Capability-Annotated Configurable Process Graph)**.** A *Capability-Annotated Configurable Process Graph* is a directed graph $G =< N, C, A, T, Cap, CN, CC, Tag >$, where *N, C, A, T* and *Cap* are as specified in Definition 2: N is a set of work nodes that are both event and activity nodes; C is a set of graph connectors: i.e., ANDsplit, ANDjoin, ORsplit, ORjoin, XORsplit, and XORjoin; A is a set of directed arcs for interconnecting all the graph nodes; T is a type function, it associates with each node its respective type (i.e., a string to indicate: activity, event, XorSplit, etc.); and *Cap* is an annotation function that associates with each activity node n a tuple Cap(n) = (ActionCategory (n), Properties(n)).

$CN \subseteq N$ is the set of configurable nodes. $CC \subseteq C$ is the set of configurable connectors. *Tag* is a tagging function that associates for each item in *N, C* and *A* the identifier of the model it originated from.

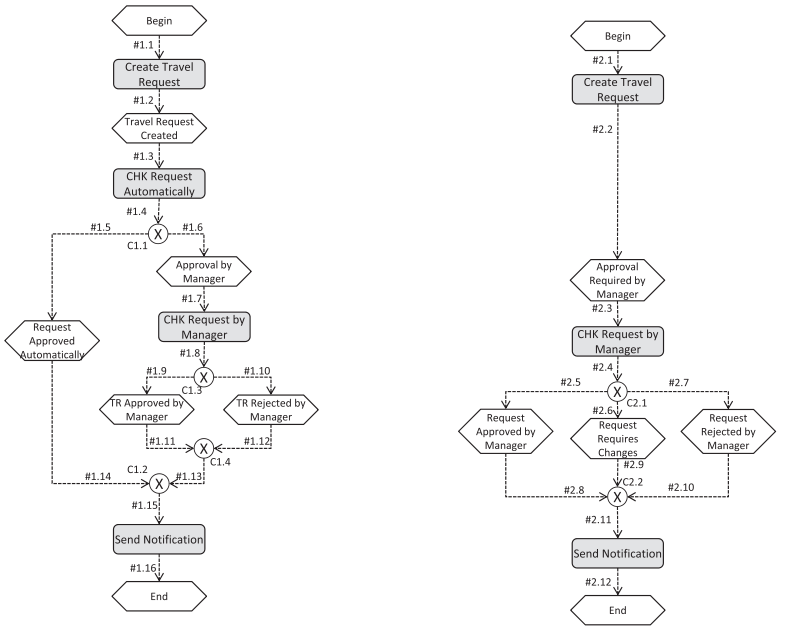
The tagging function *Tag* is used in order to be able to trace back the origin of each item of the configurable process graph (see Requirement *Traceability*). La Rosa et al. [9] use traceability also as a requirement during the creation of configurable process models. Knowing the origin of each item helps end-users, during the configuration phase, to know in what context (i.e., original model) a particular function/event has been used.

3. Running example

The running example depicted in Fig. 2 presents two business process variants that follow the EPC notation[1]. These process models are taken from SAP Workflow Scenarios in Travel Management [26], they describe two travel request approval processes: automatically [24] (see Fig. 2a that is referred as *SAP_TR_A*) and manually [25] (see Fig. 2b that is referred as *SAP_TR_M*). These models involve four functions: "*Create Travel Request*", "*CHK Request Automatically*", "*CHK Request by Manager*" and "*Send Notification*".

- "*Create Travel Request*" consists of filling a form with the details of the travel request. This function appears in *SAP_TR_A* and *SAP_TR_M* and in both variants it is triggered with the same start event (i.e., "*Begin*").
- "*CHK Request Automatically*" is an automated process for approving a travel request with respect to the requested budget for the travel. This function appears only in *SAP_TR_A*. It is triggered by the event "*Travel Request Created*".

[1] Please note that, for presentation purposes, identifiers for arcs and connectors are added which is not part of the original EPC notation.



(a) Automatically Approving Travel Requests (adapted from [24])

(b) Manually Approving Travel Requests (adapted from [25])

Fig. 2. Two business process variants from SAP Workflow Scenarios intravel management [26].

- "*CHK Request by Manager*" asks the manager to decide about the travel request, it is triggered when the "*Approval by Manager*" is required. In *SAP_TR_A* the manager can either approve or reject the travel request, this results into two respective events: "*TR Approved by Manager*" or "*TR Rejected by Manager*". In *SAP_TR_M* the manager can also ask for more clarifications or make changes to the travel request and this is shown via the event "TR Requires Changes".
- "*Send Notification*" consists of sending a notification to the requester. This function appears in *SAP_TR_A* and *SAP_TR_M* and in both variants it terminates the business process.

More formally, and with respect to Definition 2 introduced in Section 2.2 both business process models are defined as follows:

- $SAP\_TR\_A = < N_{SAP\_TR\_A}, C_{SAP\_TR\_A}, A_{SAP\_TR\_A}, Cap_{SAP\_TR\_A} >$, where $N_{SAP\_TR\_A}$, $C_{SAP\_TR\_A}$ and $Cap_{SAP\_TR\_A}$ are shown in Table 1 and $A_{SAP\_TR\_A} = \{\#1.n \mid n \in [0, 16]\}$ as shown in Fig. 2a (e.g., #1.1 = (Begin,Create Travel Request) and #1.4 = (CHK Request Automatically, C1.1)).
- $SAP\_TR\_M = < N_{SAP\_TR\_M}, C_{SAP\_TR\_M}, A_{SAP\_TR\_M}, Cap_{SAP\_TR\_M} >$, where $N_{SAP\_TR\_M}$, $C_{SAP\_TR\_M}$ and $Cap_{SAP\_TR\_A}$ are shown in Table 2 and $A_{SAP\_TR\_M} = \{\#2.n \mid n \in [0, 12]\}$ as shown in Fig. 2b (e.g., #2.1 = (Begin,Create Travel Request) and #2.4 = (CHK Request by Manager, C2.1)).

It is important to note that the original models [26] were incomplete and not well structured. They have been manually adapted to ensure that there are no deadlocks, dead-end paths, incomplete terminations, etc [27]. Additionally these models were not annotated with any capability, the capabilities of each function item have been manually created using the capability meta-model introduced in Definition 1.

4. Merging capability-annotated process models: The merging algorithm

This section presents a novel algorithm for creating configurable process models by merging pairs of process variants. The input of this algorithm is a pair of *configurable process models* and the output is a *configurable process model*. If the input models are not configurable, it starts by transforming them into *configurable models* that mainly assures that the models' items are annotated with the identifier of the model they originate from in order to fulfill the traceability requirement [Traceability] (see Section 1).

The assumptions for this algorithm are as follows:

1. For both input models, every function item is annotated with its capability (see Definition 1).
2. Both models are annotated with concepts from the same ontologies (i.e., same actions ontology and same capability domain ontology) and use the same language. In the absence of this requirement, an alignment of the ontologies used [28] or a cross-lingual comparison of business terms [29] is required.
3. Both models are well structured: there are no deadlocks, dead-end paths, incomplete terminations, etc [27].

The Merging algorithm can be split into three steps:

1. *Merging both processes' items*: first, match and merge each event and function item of a first model with its corresponding item of the second model. This is followed by integrating the rest of the models' items (i.e., connectors and arcs) into the resulting model without any matching step. This step is detailed in Section 4.1.
2. *Post-processing the merged process graph*: the previous step provides a process graph that does not respect the modelling languages syntactic rules. The object of this step is to detect modelling problems and correct the resulting model. This step is detailed in Section 4.2.
3. *Reduction of the configurable process graph*: when resolving syntactic problems, the Merging algorithm will create additional routing nodes that generate several connector chains. This step aims to reduce connector chains for a more compact configurable process graph. Details are discussed in Section 4.3.

Table 1
Listing of the nodes of SAP_TR_A with their types and business capabilities.

Node: *n*	Type: *T(n)*	Business capability: *Cap(n)*
Begin	InitialNode	*Not applicable*
Create Travel Request	Function	:CreateTravelRequest_Cap_A a cmm:Capability ; cmm:achieves bt:FillTravelRequestForm ; bt:name xsd:String ; bt:destination dbo:City ; bt:departureDate xsd:Date ; bt:returnDate xsd:Date ; bt:budget xsd:Double ; bt:purposeOfTravel xsd:String .
Travel Request Created	Event	*Not applicable*
CHK Request Automatically	Function	:CHKRequestAutomatically_Cap_A a cmm:Capability ; cmm:achieves bt:CheckTravelRequestAutomatically ; bt:budgetLimit xsd:Double .
Request Approved Automatically	Event	*Not applicable*
Approval by Manager	Event	*Not applicable*
CHK Request by Manager	Function	:CHKRequestByManager_Cap_A a cmm:Capability ; cmm:achieves bt:CheckTravelRequestByManager ; bt:decision bt:accept, bt:reject.
TR Approved by Manager	Event	*Not applicable*
TR Rejected by Manager	Event	*Not applicable*
Send Notification	Function	:SendNotification_Cap_A a cmm:Capability ; cmm:achieves bt:SendNotification ; bt:notificationMessage xsd:String .
End	FinalNode	*Not applicable*
C1.1	XORSplit	*Not applicable*
C1.2	XORJoin	*Not applicable*
C1.3	XORSplit	*not applicable*
C1.4	XORJoin	*not applicable*

Table 2
Listing of the nodes of SAP_TR_M with their types and business capabilities.

Node	Type	Capability
Begin	InitialNode	*Not applicable*
Create Travel Request	Function	:CreateTravelRequest_Cap_M cmm:achieves bt:FillTravelRequestForm ; bt:name xsd:String ; bt:destination dbo:City ; bt:departureDate xsd:Date ; bt:returnDate xsd:Date ; bt:budget xsd:Double ; bt:purposeOfTravel xsd:String .
Approval Required by Manager	Event	*Not applicable*
CHK Request by Manager	Function	:CHKRequestByManager_Cap_M a cmm:Capability ; cmm:achieves bt:CheckTravelRequestByManager ; bt:decision bt:accept, bt:reject, bt:adjust.
Request Approved by Manager	Event	*Not applicable*
Request Rejected by Manager	Event	*Not applicable*
Request Requires Changes	Event	*Not applicable*
Send Notification	Function	:SendNotification_Cap_M a cmm:Capability ; cmm:achieves bt:SendNotification ; bt:notificationMessage xsd:String ; bt:meansOfCommunication vcard:Email .
End	FinalNode	*Not applicable*
C2.1	XORSplit	*Not applicable*
C2.2	XORJoin	*Not applicable*

4.1. Merging processes' items

The Merging algorithm requires as input two configurable process graphs. If the input models are not configurable, they need to be transformed to be compliant with Definition 3. Given the process graphs depicted in Fig. 2, this step needs to transform them into Configurable Process Graphs as per Definition 3. This is a trivial operation because for both models $CN = \emptyset$, $CC = \emptyset$ and $CO = \emptyset$. The function *Tag* simply consists of tagging of each item in both models with their respective model's identifier. Both models become configurable models with single variants and without any configurable nodes.

4.1.1. Merging events

The following step of the Merging algorithm consists of matching each *event* and *function* from both input models. The object of this operation is to identify similar events and functions in order to merge them into a single node. A straight forward solution to this step can be carried out by imposing the use of common labels for events and functions in all variants. This is a valid assumption when all process variants are created within the same modelling environment with a well defined organizational taxonomy that defines a controlled vocabulary to design processes using the concepts of information entity, business process, organizational unit, actor, business schedule and business goal [30].

In the absence of an agreement on a common taxonomy in process design, modelers may loosely agree on some terminology represented in a large corpus of text used within their business environment. Such corpus can be used to construct distributional models of meaning to generate semantic similarity and relatedness between the used terms. Semantic similarity between terms is based on the co-occurrence of terms in similar contexts in the corpus [31]. Two of the most powerful distributional semantic models are the Latent Semantic Analysis (LSA) [32] and the Explicit Semantic Analysis (ESA) [31]. In both approaches, a large corpus of text documents is indexed to extract statistical properties about terms. Wikipedia is a good example of corpus initially used by ESA. Upon such indices, a semantic similarity/relatedness measure is operated. In such context, Freitas et al. [33] proposed an approximate query processing approach for databases based on distributional semantics and validated it within a natural query scenario over graph databases. This paper adopts the same approach and uses ESA-based over a domain independent corpus (Wikipedia) for matching events using their labels.

Table 3, shows the scores of matching events from $G_{SAP\_TR\_A}$ to $G_{SAP\_TR\_M}$. Each cell represents the score that is computed as follows: the similarity between two event labels El_1 and El_2 is the average of the similarities between each air of words (W_1, W_2) such that $W_1 \in EL_1$ and $W_2 \in EL_2$. For example, the matching score between "Approval by Manager" and "Approval Required by Manager" is 0.347. This score is computed after removing the stop-word "by" and computing the similarity between six possible pairs of words from these labels (e.g., Similarity("Approval", "Approval") = 1, Similarity("Approval", "Required") = 0.042, Similarity("Approval","Manager")=0.005)[2]. Note that for

[2] Please note that scores are computed after stemming.

Table 3
ESA-based matching scores matrix for events from the running example of the two process models for organizing a trip (see Fig. 2).

	Events from SAP_TR_M					
Events from SAP_TR_A	Begin	Approval required by manager	Request approved by manager	Request requires changes	Request rejected	End
Begin	1	0.016	0.005	0.021	0.003	0.046
Travel Request created	0.010	0.015	0.121	0.179	0.174	0.005
Request approved automatically	0.005	0.126	0.232	0.186	0.183	0.003
Approval by manager	0.005	0.347	0.340	0.026	0.018	0.008
TR Approved by manager	0.005	0.347	0.340	0.028	0.018	0.008
TR Rejected by manager	0.004	0.181	0.179	0.018	0.259	0.010
End	0.046	0.007	0.006	0.005	0.004	1

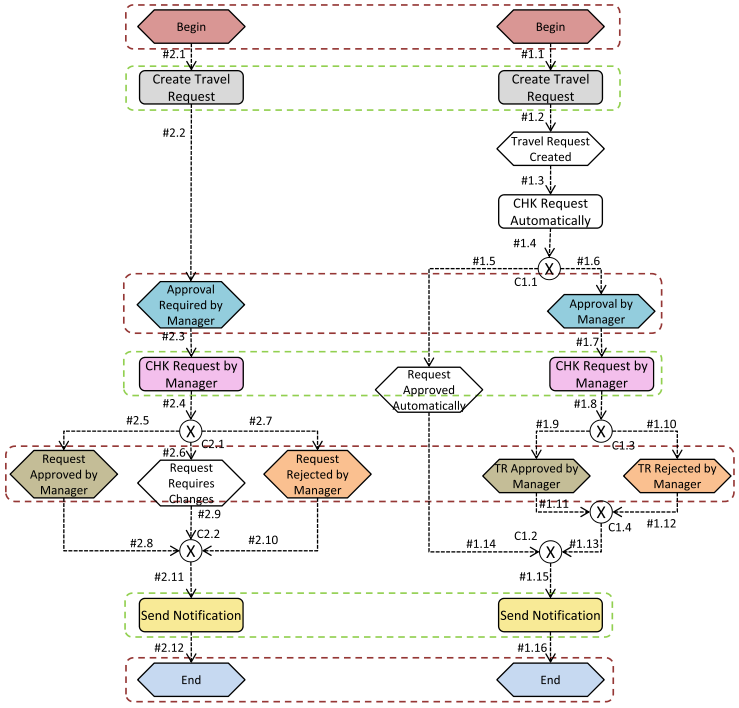


Fig. 3. Matching events and functions from both model variants.

each line in Table 3, the two highest matching scores for each pair of event labels are highlighted. If for a line the score is 1, this means that the labels are identical, then obviously they merge together without further verifications, otherwise, the user can be asked to select the most appropriate label ordered with the highest score.

The matching score is used mainly for helping the user to select the best matching between events. Even though the matching score has been used to fully automate the merging of events in prior research [9,10], we prefer giving the user the possibility to take the final decision. Indeed, in some cases, one can find a better matching of events according the score while the actual matching is different (see the matching score of "TR Approved by Manager" and "Approval Required by Manager" in Table 3). It is also possible to find high matching scores while there is no actual correct matching (see the matching score of "Request Approved Automatically" and "Request Approved by Manager" in Table 3).

After choosing the best matching between pairs of events, each pair of events is merged into a single one with the most appropriate label that the user can select. For traceability purposes, it is possible keep both labels in the merged event with additional tagging of the origin of each of them. However, this work keeps only one of the labels.

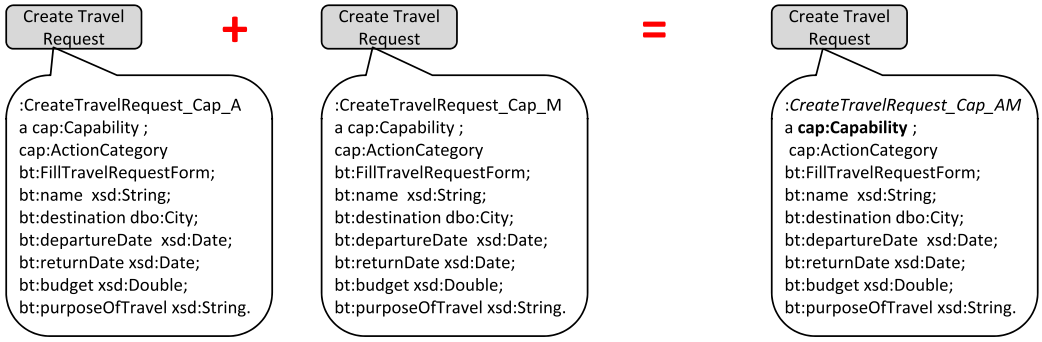
Fig. 3 highlights the matching of events of the input models proposed in the running example in Fig. 2. Each event from the first model share the same colour with its corresponding event from the second model. Events kept in white do not have any corresponding event in the other model.

4.1.2. Merging functions

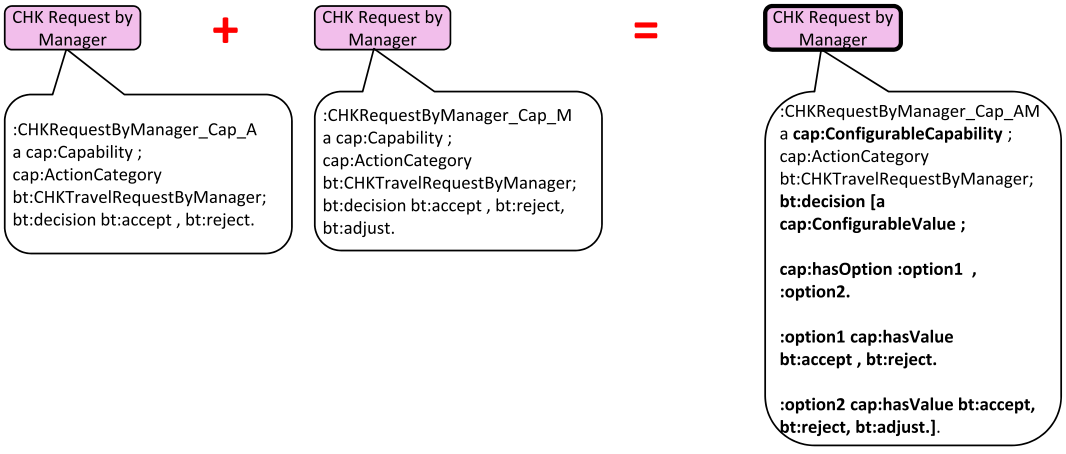
A primary assumption of the proposed algorithm imposes that both input models are annotated with their capabilities. This makes the matching of function items simpler than event items. Indeed, similar functions in essence achieve the same action and consequently should have the same *action category* of their capabilities. The matching of function items is simply done via comparing their action categories. However, the resulting merged function item should consider all differences between the capabilities from original models. Consequently, the merging of function items is a two-step operation that first identifies the corresponding items (see Fig. 2 as example) then the second generates their merged capability.

For determining the resulting merged capability, the algorithm covers all possible cases:

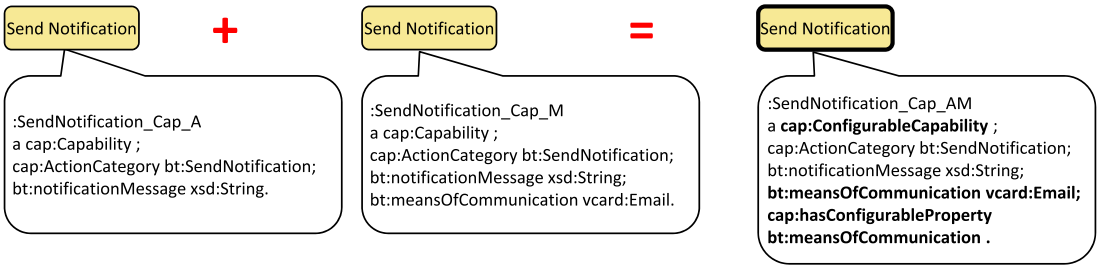
- Both function items have the same capability: the resulting merged capability remain as it is (see Fig. 4a).



(a) Merging "Create Travel Request" functions: regular capability



(b) Merging "CHK Request by Manager": configurable capability with configurable property value



(c) Merging "Send Notification": configurable capability with configurable property

Fig. 4. Merging functions and creating configurable capabilities.

- Both capabilities share the same property but with different values: the resulting merged capability is a *Configurable Capability* with a property that has a *Configurable Value* (see Fig. 4b). In all cases the configurable value is an *Enumeration* of both options originating from input capabilities.
- One of the capabilities has one additional property: the merged capability is a *Configurable Capability* that has a *Configurable Property*: i.e., the additional property (see Fig. 4c).

4.1.3. Merged process graph

Note that the previous matching steps relate only to events and functions. Connectors from the first model can also be matched to connectors from the second one as it has been proposed by La Rosa et al. [9,10]. The matching operation is done via a context similarity score by considering the connector neighbourhood (i.e., incoming and outgoing nodes). This operation is not necessary and adds more complexity to the matching operation when connector chains appear in the model (i.e., various consecutive connectors).

The following step consists of creating the integrated configurable process graph denoted $CG = < N_{CG}, C_{CG}, A_{CG}, T_{CG}, Cap_{CG}, CN_{CG}, CC_{CG}, CO_{CG}, Tag_{CG} >$ (see Definition 3). Let $G1 = < N_{G1}, C_{G1}, A_{G1}, T_{G1}, Cap_{G1}, CN_{G1}, CC_{G1}, CO_{G1}, Tag_{G1} >$ and $G2 = < N_{G2}, C_{G2}, A_{G2}, T_{G2}, Cap_{G2}, CN_{G2}, CC_{G2}, CO_{G2}, Tag_{G2} >$ be two input configurable process graphs, the resulting CG is constructed as follows:

- $N_{CG} = N_{G1} \cup N_{G2}$: the set of nodes of the configurable graph is the union of nodes of both input models. The matching operator for merging events is based on the events' labels, while

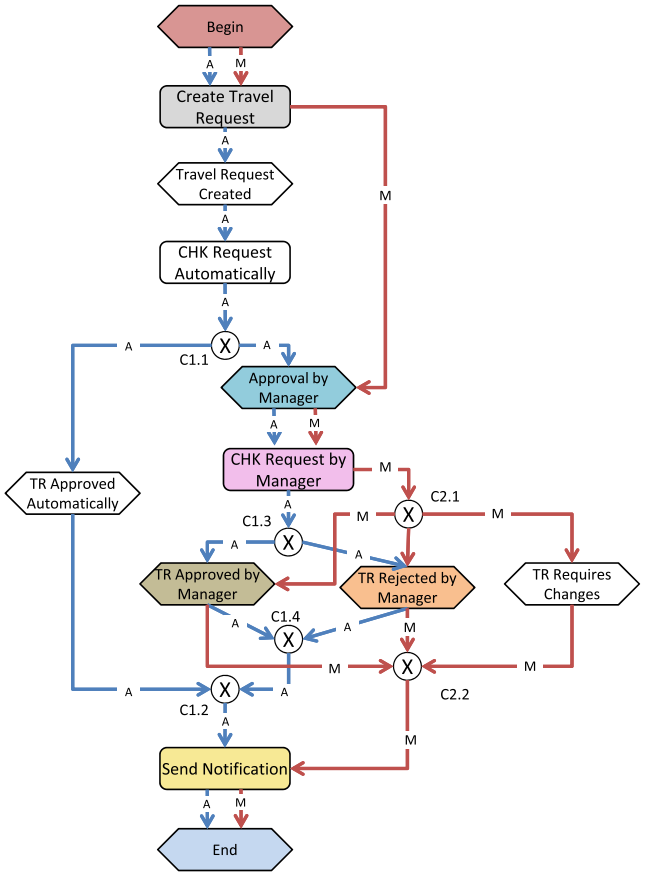


Fig. 5. Merging items from both model variants.

the matching of functions is based on capabilities as explained previously in Sections 4.1.1 and 4.1.2.

- $C_{CG} = C_{G1} \cup C_{G2}$: the set of connectors of the configurable graph is the union of connectors of both input models without any merging. If two connectors from both input models share the same identifier, then one of them is changed in order to make sure that routing information is preserved.
- $A_{CG} = A_{G1} \cup A_{G2}$: the set of arcs of the configurable graph is the union of arcs from both models while considering updating sources and destinations of arcs with respect to the unified events, functions or updated identifiers of connectors.
- T_{CG} and Cap_{CG} are two functions that report on the type of the node and the capabilities of the function items, respectively.
- $CN_{CG} = CN_{G1} \cup CN_{G2} \cup CN_{G12}$: contains the list of function items that have configurable capabilities either originally defined in input models (i.e, $CN_{G1} \cup CN_{G2}$) or resulting from the merging of function items (i.e, CN_{G12}) (see Section 4.1.2).
- $CC_{CG} = CC_{G1} \cup CC_{G2}$: contains the list of all configurable connectors from the original models. In the running example, there are no configurable connectors, consequently at this stage, $CC_{CG} = \{\}$.
- The tagging function Tag_{CG} will assign for each element of the graph the identifiers of the model they originated from (e.g., id(G1) and id(G2)).

Fig. 5 illustrates the resulting configurable process model after the merging operation. Only the tags on the arcs are shown in this Figure. One can easily notice that this model is not well structured: there are (1) duplicate arcs (e.g., from "Begin" and "Create Travel Request"), and (2) events/function nodes with multiple incoming/outgoing arcs. A post-processing step is required to resolve these issues.

4.2. Post-processing the merged process graph

The resulting *CG* needs some post-processing in order to be well structured and respect the set of requirements imposed by the modelling notation (EPC in this case) [34]. After the merging operation two requirements are violated. These requirements are

1. for each $n \in N_{CG}$: $|\bullet n| \leq 1$. This requirement means that each event/function item must have at most one input.
2. for each $n \in N_{CG}$: $|n\bullet| \leq 1$. This requirement means that each event/function item must have at most one output.

To ensure that each event/function item has a single entry, Algorithm 1 operates as follows: if a work node has more than one input, it creates a configurable connector (XOR-Join) that becomes the new destination of all input edges of that work node (i.e., lines 7 to 12). Finally, it creates a new edge from the new configurable connector to the work node that previously had more than one entry (i.e., lines 16 to 20).

Algorithm 1: Single entry: Ensuring that each work node has a single entry.

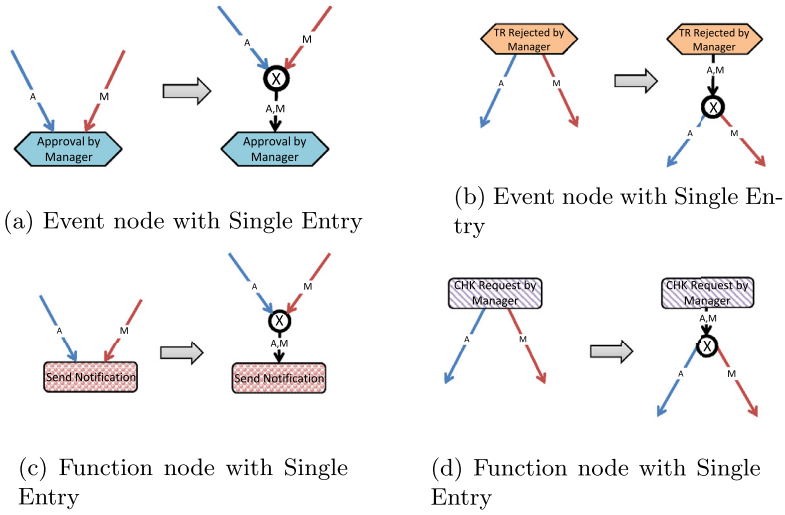
Input: Graph G: A graph that represents a configurable process graph.

```
1  begin
2    foreach n in N_G do
3      Tags ← {};
4      if |•n| > 1 then
5        CreateNewCXOR(CXOR);
6        CXOR.Type ← XORjoin;
7        foreach a ∈ A_G do
8          if a.Destination == n then
9            a.Destination ← n;
10           Tags.add(a.Tag);
11         end
12       end
13       CXOR.Tag ← Tags;
14       C_G.add(CXOR);
15       CC_G.add(CXOR);
16       CreateNewArc(Arc);
17       Arc.Source ← CXOR;
18       Arc.Destination ← n;
19       Arc.Tag ← Tags;
20       A_G.add(Arc);
21     end
22   end
23 end
```

Fig. 6a and c illustrate how this transformation is done. The left hand side of Fig. 6a depicts two input arcs for the event *"Approval by Manager"* which has been changed in the right hand side of this figure by inserting a configurable XOR connector and an arc from this connector to *"Approval by manager"* that is tagged with all tags of original arcs (i.e., "'A,M").

A similar algorithm operates to ensure that each event/function item has a single exit. Fig. 6b and 6d illustrate how this transformation is done. The left hand side of Fig. 6b depicts two output arcs for the event *"TR Rejected by Manager"* which has been changed in the right hand side of this figure by inserting a configurable XOR connector and an arc from *"TR Rejected by Manager"* to this connector that is tagged with all the tags of the original arcs (i.e., "'A.M").

At this level, the merged process model is completely constructed and Fig. 7 depicts the resulting model. However, during this post-processing step, several configurable connectors have



(a) Event node with Single Entry

(b) Event node with Single Entry

(c) Function node with Single Entry

(d) Function node with Single Entry

Fig. 6. Introducing configurable connectors to make work nodes with a single entry and a single exit.

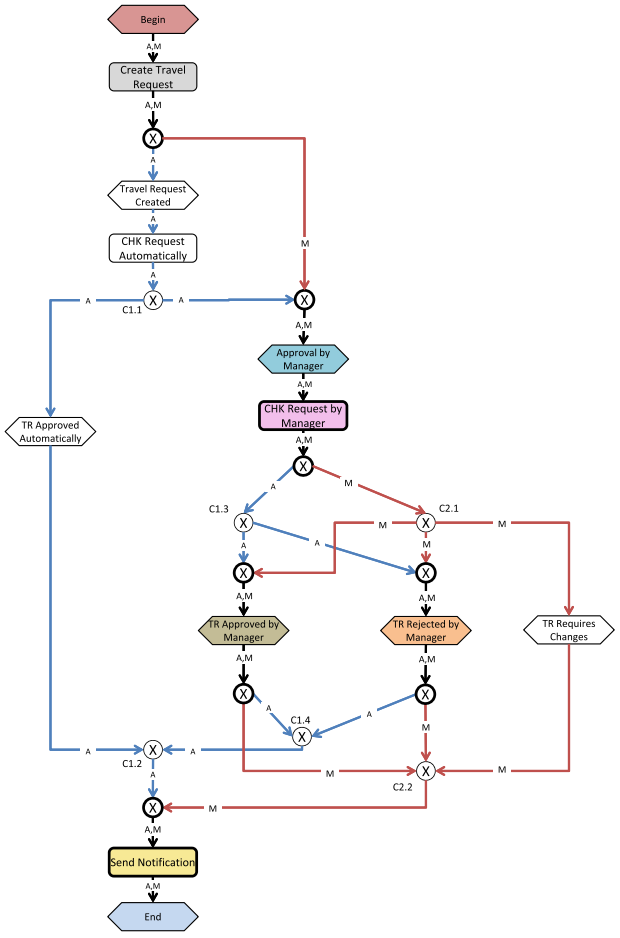


Fig. 7. Correct configurable model after post-processing step.

been inserted. This may lead to the appearance of several connector chains that make the model more complex to read. The following section shows how to reduce these connector chains in order to obtain a reduced configurable process model.

4.3. Reduction of the configurable process graph

This section presents two rules that help reduce connector chains. These rules (inspired from the work of La Rosa et al. [9]) should reduce the process graph while preserving its behaviour. Reducing a process graph consists of deleting some routing nodes which are not mandatory. These reduction rules are: (i) Merging consecutive split/join connectors and (ii) Removing trivial connectors.

4.3.1. Merge consecutive split/join connectors

Algorithm 2 identifies and merges consecutive split/ join connectors into a single connector. It starts by parsing all the arcs of *CG*. If the source and the destination of an arc are two connectors of the same type (i.e., join or split) (i.e., line 2), then the algorithm fetches all the adjacent connectors having that type in order to create a set of connectors that have to be reduced (i.e., lines 5 to 7). The reduction of this set of connectors is made by creating a new configurable connector that replaces them. If one of the connectors that needs to be reduced is either an AND or an OR (i.e., line 11), then the new connector must be a configurable OR keeps a trace back to the original operator with the identifier of the process where it originates from (i.e., line 14) otherwise it is a configurable XOR (i.e., line 9). Then, the algorithm continues to parse the remaining arcs to detect input/output edges of the current connectors in order to link them to the new connector (i.e., lines 18 to 23). Line 25 of Algorithm 2 allows merging arcs having the same source and destination.

In Fig. 8a (as it appears in this use case), connectors *CG*1, *C*1.3, and *C*2.1 are three consecutive split connectors which are merged into *CG* in Fig. 8b. In Fig. 8b there are two arcs from *CG* to *CG*2 with tags “A” and “M” which respectively originate from (Fig. 8a) the arc from *CG*1 to *C*1.3 and the arc from *CG*1 to *C*2.1. These two arcs are merged into a single one with the tag “A,M” in Fig. 8c.

4.3.2. Remove trivial connectors

A trivial connector is a connector that has only one input and one output arc. It does not provide any useful routing information. Thus, it can be removed without altering the process behaviour. In the running example, Fig. 8c depicts two trivial connectors (i.e., *CG*2 and *CG*3) that can to be removed. Fig. 9b depicts the resulting optimal configurable process model of the running example of this paper.

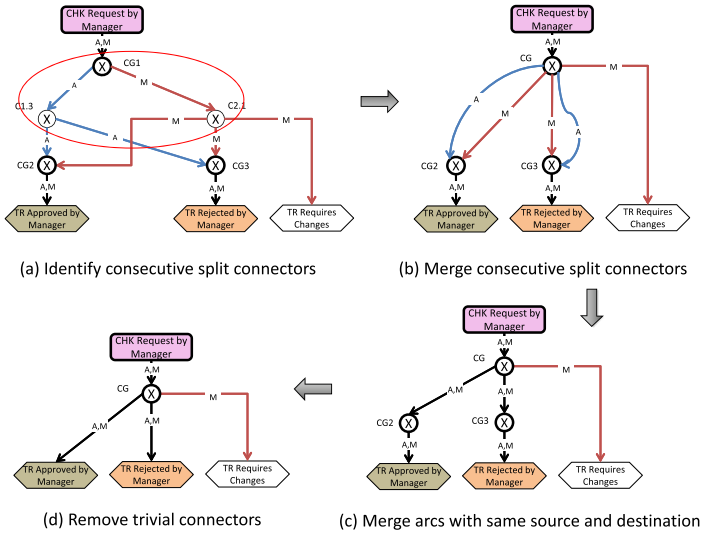
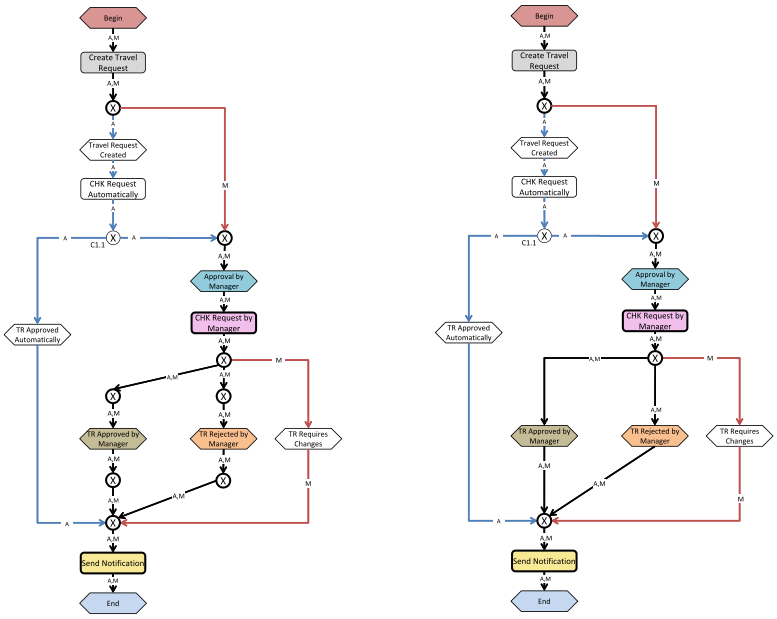


Fig. 8. Reducing a process graph by merging consecutive split connectors.



(a) Configurable Model after Reducing Connector Chains

(b) Configurable Model after Removing Trivial Connectors

Fig. 9. Resulting configurable models after post-processing and reduction steps.

5. Tool support and evaluation

5.1. Overview

To motivate the need of having a tool support, in Section 5.2, we carried out an experiment with users that have been asked to manually merge three pairs of process models. Users find the manual merging very difficult without a tool support. As proof of concept, the proposed algorithm has been implemented as an extension of EPCTools [35], Section 5.3 reports on this extension. This tool has been used, in Section 5.4, to carry out further evaluations for measuring the compression rate gained by using this tool for merging a set of business process models and assess the required execution time. Furthermore, interviews with domain experts have been carried out in order to have a feedback on this contributions from practitioners in Section 5.5.

5.2. The need for a tool support

To further motivate the need to design an automated tool to merge process models, we carried out a user testing evaluation for manually merging three pairs of process models. Users involved in this evaluation are postgraduate students from information technology and business background that have been lightly trained in process modeling concepts.

5.2.1. Methodology

The objective of this experiment is to show the complexity of the manual merging of process models and show how lightly trained users finds the task difficult and can make very obvious mistakes even with simple process models. To do so, we proceeded as follows:

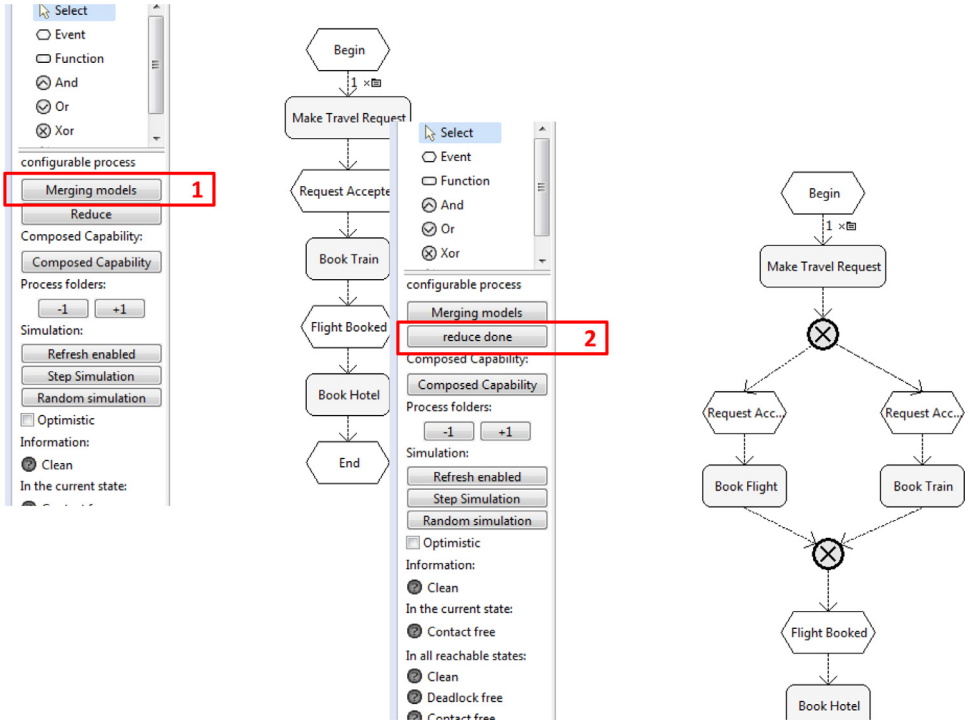


Fig. 10. Extended version of EPCTools that supports the creation of capability-annotated configurable business process models.

Algorithm 2: Merge consecutive connectors.

Input: Graph G: A graph that represents a configurable process model.

begin
  foreach $(Src1, PID1, Dest1)$ *in* $E_G/\{Src1, Dest1\} \subset RN_G$ *and* $(\lambda(Src1) = \lambda(Dest1))$ **do**
    $ToBeReduced \leftarrow \{Src1, Dest1\}$;
    foreach $(Src2, PID2, Dest2) \in E_G/Src2$ *or* $Dest2 \in ToBeReduced$ *and* $(\lambda(Src2) = \lambda(Dest2) = \lambda(Src1))$ **do**
      $ToBeReduced \leftarrow \{Src2, Dest2\}$;
    end
    CreateNewConnector(Configurable-Connector);
    Operator = "XOR";
    foreach $Connector \in ToBeReduced$ **do**
      if $\tau(Connector) = (PID, "AND")$ *or* $\tau(Connector) = (PID, "OR")$ **then**
        Operator = "OR";
      end
      $\tau(Configurable - Connector) \leftarrow \tau(Connector)$;
    end
    $\tau(ConfigurableConnector) \leftarrow \{(id(G), Operator)\}$;
    $\eta(ConfigurableConnector) \leftarrow true$;
    foreach $(Src, PID, Dest) \in E_G/Src$ *or* $Dest \in ToBeReduced$ **do**
      $E_G \leftarrow E_G \setminus (Src, PID, Dest)$;
      if $Dest \in ToBeReduced$ *and* $Src \notin ToBeReduced$ **then**
        $E_G \leftarrow (Src, PID, ConfigurableConnector)$;
      end
    end
  end
  $MergeEdgesWithSameSource\&Destination(G)$;
end

- 20 users have been invited to take part of this experiment. Users have been lightly trained to the modeling of process models using EPC as a modeling language.
- 10 of these users know the overall objective of this work and are more familiar with the proposed approach.
- Each user has been given three pairs of process models (see Section 5.2.2) that they have to merge.
- We asked the users to measure the time they needed to merge the model.
- Feedback from the users were collected and discussed in Section 5.2.3.

5.2.2. Process models used

We created three pairs of process models that users needed to merge manually (see Table 4). Each of the pairs has a different

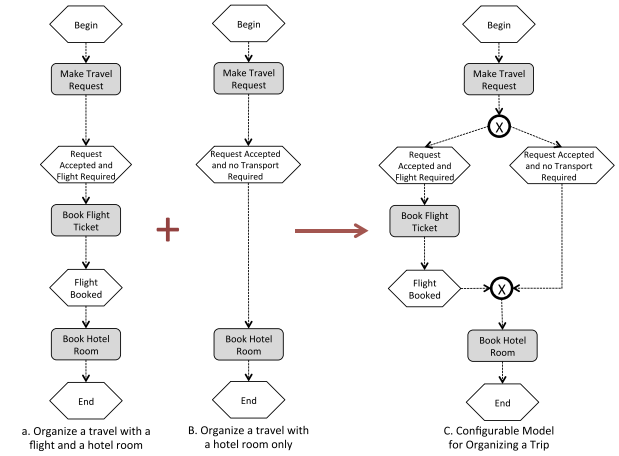


Fig. 11. A configurable process model for organizing a trip created by merging two simple variants.

Table 4
Details of the pairs of process models used in the manual merging user testing.

Pair	Size of the first model	Size of the second model	Domain
P1	4 events; 3 functions; 0 connectors	3 events; 2 functions; 0 connectors	Booking a travel request
P2	7 events; 4 functions; 4 connectors	6 events; 3 functions; 2 connectors	Approval of travel request
P3	12 events; 8 functions; 10 connectors	10 events; 7 functions; 8 connectors	Generic process (using letters for labels of events and functions)

Table 5
Observations from the user tests.

Pair	Time required to merge the models	Percentage of correct results (%)	General comment
P1	3 to 6 mn	100	The models are easy to merge
P2	6 to 17 mn	40	Using new connectors is confusing
P1	14 to 30 mn	0	The models are difficult to merge

level of difficulty. The first pair is easy to merge; we used a process of booking a travel request as users are familiar with; each of the models uses a sequence flow of events and functions (see Fig. 11). The second pair is a little more difficult to merge; we used a process of approval of travel requests as users are familiar with too; this is the pair used as the running example of the paper (see Fig. 2 in Section 3). The third pair is difficult to merge; this pair has been adapted from customs clearance procedures [36], we replace the labels by letters as users are not familiar with the domain of customs clearance. All the models are well structured and have a single starting event and single end event.

5.2.3. User testing results

As shown in Table 5, merging the first pair of models was quick and without modeling errors. Each of the models represents only a sequence of functions and events. Consequently, the merging was straightforward to all the users that needed to introduce a pair of connectors (the variation points) to include a choice between two possible paths in the merged model. All users agreed that this pair is very simple to merge and with some practice, they can improve the time required to merge them. Given that the task was completed without errors in less than 6 min, a tool support at this stage was not necessary.

The merging of the second pair of models took almost twice the time required for merging the first pair with only 40% of correct results. The problem that most users encountered was the use of consecutive connectors. One of the users said "*I was not confident about what I was doing, so I propose two versions of this model*" that both were unfortunately wrong.

At this stage, none of the users mentioned the need for a tool support, however, they mention that a more advanced training was required. For example, one of the users mentioned that "*this is not as easy as it looks, but I think one can get better and faster after going through some training*".

The merging of the third pair of the models was the most challenging exercise of all the participants. The actual time required for proposing a solution was between 22 and 30 min. The user that finished at 14 min proposed an incomplete model mentioning that he got confused with functions that were used at different locations in both models. Indeed, we deliberatively used a pair of models where the order of two functions is reversed from one model to the other.

In this particular case, one of the possible solutions is create multiple connectors resulting in various connector chains. This made the model complicated to create as mentioned by one of the users "*even if you get really good at this, there is a high possibility of what is called* "Human Error" *like the third example*".

At this stage, all the users agree that a tool support is needed. Few tests for merging the same models with a proposed tool support described in Section 5.3 were carried out showing a high level of appreciation by most of the users. The tests showed that the merging was done in few seconds, a more detailed time evaluation is reported in Section 5.4.

5.3. Tool support

The presented business process merging algorithm has been implemented as an extension of EPCTools [35]. EPCTools is an open source initiative toward a tool for Event Driven Process Chains (EPCs) that supports the tool independent EPC interchange format EPML [37] implemented as an Eclipse Plug-in.

As shown in Fig. 10, after opening one of the two process models, the user has to click on the "Merging models" button (see 1 in Fig. 10), then a new dialog window is open, the user selects the second process model and clicks on ok, in this step the new configurable process is created. The user can optionally decide to apply the reduction step by selecting the "Reduce" button (see 2 in Fig. 10).

These very simple instructions together with an initial introduction to EPC modeling language were given to a group of 20 users with the objective to get their feedback on how easy their merging operation is.

The evaluation was centered only on the instructions given after a short demo. No further evaluations regarding the user interface and how the user interacts with this tool have been carried out. The user experience evaluation might be influenced by the modelling environment and is not relevant in the context of the contribution of this paper.

All of the users completed the merging of the pairs of the models used in Section 5.2.2 in few seconds and in the worst case it took less than 2 min due to some technical issues with Eclipse.

Furthermore, the tool has been used for merging business process models for other types of evaluations such as the compression rate and carrying out interviews with domain experts that are reported in the following sections.

5.4. Compression rate and time evaluation

5.4.1. Methodology

The objective of the compression rate evaluation is to highlight the benefit of merging business process variants into a single configurable business process model by avoiding duplicate process elements in process repositories. Furthermore, as for organisations time is important and should not be spent on manual creation of configurable models, this evaluation shows how quick the merging algorithm delivers configurable process models.

The evaluation of compression rate and execution time has been carried out as follows:

1. A test collection of real-world business process models have been manually created.
2. Each of the input models have been quantified in terms of number of process elements (i.e., events, functions and connectors).
3. Using the tool support, we have created configurable process models from the input models.

Table 6
Results of merging registration processes of Dutch municipalities.

Process Number	Input size (Number of Nodes)	Output size before reduction	Output size after reduction	Execution Time (ms)
P1	190 (29+56+52+29+24)	131 (31%)	71 (62%)	302
P2	347 (63+84+73+57+70)	276 (20%)	180 (48%)	453
P3	507 (76+127+127+114+63)	298 (41%)	214 (57%)	721
P4	355 (56+111+91+67+30)	266 (25%)	160 (54%)	482

4. Each resulting configurable process model has been quantified in terms of number of process elements.
5. Measuring the compression rate by comparing the sizes of the input models and the output configurable model.
6. Measuring the execution time of the merging process.

Please note that the execution of the merging steps has not been interrupted with any manual task. In this regard the following actions have been taken:

- Event matching has been based on exact matching of events (in order to reduce manual input from the end-user).
- All function items are annotated with the same capability ontology. For this evaluation, each function is annotated only with its action category (i.e., there are no related properties).
- All the model variants are merged at once (instead of merging each pair one by one manually).
- The reduction step has been carried out automatically after merging (no manual decision regarding the reduction step).

5.4.2. Test collection

We have manually created a test collection for evaluating the proposed merging algorithm. The process variants that we used in the experiment are those used by Gottschalk in his thesis [13]. They were subject of a case study [38] in which techniques for managing configurable process models were extensively tested in a real-world scenario. We have also used this test collection for evaluating an earlier version of this work [11]. The process models used in this case study are four processes out of the five most executed registration processes in the civil affairs department of Dutch municipalities [13]:

- *P1: Acknowledging an unborn child*: This process is executed when a man wants to register that he is the father of an unborn child in case he is not married to his pregnant partner.
- *P2: Registering a newborn*: This process describes the steps for registering a newborn and get his birth certificate.
- *P3: Marriage*: This process describes all the steps required before getting married in a Dutch municipality.
- *P4: Decease*: This process describes the steps required by relatives to bury the deceased and get a death certificate.

The process variants considered in this evaluation are initially available in Protos[3]. Each process has five process variants. Consequently, a total of $5 \times 4 = 20$ process models were considered in this work (similar to the case study [38]). We have manually translated these models into EPC and used the extended version of EPCTools (see Section 5) for merging them in order to create configurable process models for each process.

5.4.3. Observations

During the merging steps, two metrics were observed: process models sizes (before, and after the merging) and the execution time of the merging steps. These metrics are shown in Table 6.

Table 6 shows the size of the input and output models (size in terms of number of EPC nodes). Column one states the four processes considered here (P1: Acknowledging an unborn child, P2: Registering a newborn, P3: Marriage and P4: IDecease). Column two shows the size of the input models, entries of this column present the sum of the number of nodes of each variant as it is mentioned between parenthesis. Columns three and four show the size of the output models before and after the reduction step of the proposed algorithm which represent the size of the constructed configurable process model. The percentage value between parenthesis shows the compression rate gained from the creation of the configurable process models. Column five shows the execution time in milliseconds needed for merging the input process models.

5.4.4. Discussion

The reduction approach can gain around 50% in terms of space for storing several process variants. Besides this space gain, we can see that in a few milliseconds a set of five process variants can be automatically merged which would take much longer for a business analyst to perform the task manually.

The compression rates are considerably higher after the reduction step. This step removes only connectors that are created for ensuring that events and functions have a single input and single output (see Section 4.2). In fact, generated connector chains can be reduced into a single connector without losing any routing information as per the reduction step discussed earlier in Section 4.3.

In general, compression rates are high because most of the process models share various process elements. Indeed, all the used variants, are from various Dutch municipalities that are initially defined from a high level reference model [38]. Depending on the population and the available resources of each municipality, few process tasks are either skipped or replaced by other ones. This keeps most of the process functions sequentially aligned. Consequently, the merged model observe a big number of common functions and events.

The difference in results compared to our previous version of this work [11] is perceived only in the execution time (i.e., the compression rate is the same). In the earlier version of this work, we did not consider capabilities of functions, we simply considered exact matching of their labels. The parsing of annotations and the identification of matching function items adds an estimation of 200 ms for the time required to create each configurable model.

5.5. Interviews with domain experts

5.5.1. Introduction

In this part of the evaluation, we carried out a series of semi-structured interviews with the five domain experts that have strong background and are currently active in business process management activities. Their profiles include two information systems architects, one project manager, one IT engineer and one consultant and training expert. We target these four types of stakeholders as each of them has his own perspective and usage of business process models.

The objective of this evaluation is to assess the usefulness of the proposed merging algorithm for designing configurable process models integrating business capabilities of process activities.

[3] Protos is part of Pallas Athena's BPM toolset BPM|one.

In this evaluation, a design science methodology was followed [39] together with formal guidelines for conducting and reporting case study research [40].

The interviews were done after explanation of the objectives of this work and details about configuration-based modelling and business capability foundations. The main targeted outcome of these interviews is to identify if these experts see that the business capability-driven configuration of process models is useful and can be adopted in their working environment.

5.5.2. Participants

This is a high level description of the profiles of the five participants that were recruited for this semi-structured interview. These five experts were from different levels of expertise in the area of service computing and information systems design and development. The age group of these participant is 30–50 years old and their professional background includes a minimum of 5 years experience and are currently active in their field. Their profiles include:

- two system architects: working as designers of information systems for clients of a multinational company.
- one project manager: leading teams of developers of information systems for the management of seaports in different countries.
- and one IT engineer: working as developer in start-up offering automated post services.
- one information system consultant and training expert: working as a consultant and trainer in the area of business process management.

5.5.3. Approach

The approach used for this evaluation follows the case study research process proposed by Runeson and Höst [40]:

- *Case study design*: the objective of the evaluation is to assess the usefulness of the proposed merging algorithm for designing configurable process models integrating business capabilities. Interviews run individually using online conferencing tool. Each interview took about 1 h for each participant.
- *Preparation for data collection*: The discussions were semi-structured to give the participants the freedom to give additional comments and get as much feedback as possible from them. The structure of the interviews was as follows[4]:
  1. 5 min discussion about the profile of the participant and his knowledge about reference process modelling and particularly configuration-based business process modelling.
  2. 15 min presentation of the business capability-driven design and configuration of configurable business process models.
  3. 5 min for manually merging a pairs of small business process models. The models used are the two variants of the process of organising a trip depicted in Fig. 11.
  4. 10 min for manually merging a pairs of larger business process models. The models used are the two variants of the process of importation from the customs clearance procedures as this was their domain of expertise.
  5. 10 min demo and interaction with the tool support.
  6. 15 min discussion about the contribution of this paper.
- *Analysis of collected data*: A post interview analysis of the collected feedback is reported in Section 5.5.4.
- *Reporting*: A discussion of the resulting feedback is summarised in Section 5.5.5 and shared among the participants.

[4] Please note that the durations used here are approximative. Some of the interviews run for few minutes more or less for each section.

5.5.4. Results

Key results from these interviews are as follows:

5.5.4.1. General comments on the experience of the experts in using reference process modeling approaches.

- All of the experts except the IT engineer (P4) are aware of reference process modelling in general and configuration-based modelling in particular.
  "*Of course we are familiar with reference process modelling. A lot of our missions consist of configuring our system to clients needs.*" (P5)
- Most of the configuration tools they used were focused on IT configurations, as stated by P1, P3 and P5.
- The only non-IT related configuration option that P5 encountered was the role assignment for tasks. For example, a particular task can be achieved by project managers and can be configured to other roles such as budget holder, director, etc.

5.5.4.2. Feedback on the business process merging approach.

- The manual creation of pairs of small process models by all the experts was quick for small models and done within the 5 min slot given for this task
- The manual creation of pairs of larger process models was not complete within the 10 min slots given for this task.

5.5.4.3. Feedback on the use of the business capabilities in configurable process models.

- The use of a single business capability ontology to annotate business process variants was pointed as weak point of this research by P5. Using multiple ontologies is more likely a common practice for these experts.

5.5.4.4. Feedback on the use of the tool support.

- Using the tool support to merge the models that they have manually created was highly accepted.
- Fully automated merging is not always useful (P2 and P3), it is better to consider human intervention to validate some merging decisions.
  "*The tool is useful for guaranteeing a rapid merging of models. However, it is good to give the user the possibility to take some of the merging decisions.*" (P3)
- Manual changes of the resulting configurable model are also pointed as needed by P2, P3 and P4.

5.5.5. Discussion

Most of the current solutions that these experts use are configuration-driven but not from a business capability perspective. Configurations consist of setting the communication protocols, the form fields that need to be available in the process tasks, the monitoring indicators, etc. This confirms the fact that current information systems are mainly focused on the IT engineering part of business processes and not targeting other aspects such as the business capability. The consultant and training expert (P5) finds that this is a major issue with customers that adopt for the first time their information system. For this reason an extensive training period is required in order to make business experts more familiar with the vocabulary used by the solution provided. These solutions are not flexible enough to integrate changes that customers want. It is the customer that has to adapt his work to the solution rather than the other way round.

All the interviewed experts positively perceive the usefulness of the configuration modelling approach in order to design business process models, however, they see that the major problem is how

to create these models and make them easy to manage by end-users. After showing the proposed solutions of this paper, the experts agree that this is one possible solution but remains limited in terms of using business capability-annotations using a unique ontology. This work can be further extended to include multiple ontologies for annotating business process models.

The automatic merging was a valuable addition from these experts point of view. Some of them (P1, P2 and P5) had to create manual merging of business process models in other context and they recognise that manual creation is time consuming and does not necessarily guarantee a correct result. Those that did not experience this in their working environment (P3 and P4) also adhere to this point of view after trying the manual merge of the large models during the interview.

Testing the tool reveals that it is simple to use but very EPC focused. Applying the same approach to other languages can be a good addition.

The experts pointed the need to have human interventions during the merging operation. Validating the merging of functions or events for some cases might need the expert's decision. Even after the completion of the merging process, some manual changes can be required for more flexible and tailored configurable model. Indeed, the creation of configurable process models is not exclusively done by merging process variants, other techniques such as mining process logs can be used [5,6]. It is important to point that the resulting configurable model after merging the process variants can be tailored to include other configuration options and execution paths. A future direction to look at, in the context of this research, is a hybrid solution that uses both process merging techniques together with process mining for the design of configurable process models.

The major challenge for business capability-driven configuration modelling to join industry is the fact that current industrial solutions are mature enough and hard to replace. Current solutions have been built over years of analysis, engineering and research that are proven to be effective. Replacing these solutions has never been taken as a serious option. However, features such as those proposed in our research (i.e., process annotations with capabilities and configuration of processes using their business capabilities) can be seen as additional options to the current systems but a lot of adaptation work is required.

6. Comparative analysis with related approaches

6.1. Units of analysis

This evaluation aims to compare the merging algorithm proposed in this chapter with other related business process merging approaches. As units of Analysis, we are using the requirements introduced in Section 1 . The following is a recall of these requirements:

1. *[Behaviour Subsumption]* The merged model should allow for the behavior of all the original models [7].
2. *[Traceability]* Each element of the merged process model should be easily traced back to its original model [9,10].
3. *[Deriving Original Models]* Business analysts should be able to derive the input models (as well as new ones) from the merged process model [9,10].

The comparative analysis carried out in this Section focused exclusively on business process modelling approaches. Other contributions for merging multiple perspectives of process models [41] or merging database schemas [42] and Object and Class Diagrams [43] are not considered in this analysis.

6.2. Related approaches

6.2.1. Process Merging for Version Control

In a collaborative business process modelling environment, multiple stakeholders can be involved in the design of business process models. Starting from a common version of a certain model, different users can adapt it to meet their needs resulting into multiple versions of the same model. At some point, when each of the stakeholders wants to commit his version to a common repository, each of the new versions needs to be merged in order to generate a new common version of a certain model. This is the research context that motivates the work carried out by Gerth and Luckey [44]. In their previous works [8,45–47], the authors propose a formalism to detect equivalent business process models based on the detection of equivalent fragments contained in these models. The objective it to detect and resolve version conflicts during the merging of process variants. Authors here refer to their existing tool support for model merging in IBM WebSphere Business Modeler [8]. The merge procedure defined is not intended to be fully automated, it is rather developed for reducing the number of false-positive differences and conflicts in models management. The resulting model is obtained after selecting a set of change operations and applying them on the current model. This new model is called the merged model that becomes the new common version of the process model.

6.2.2. Critique

Authors did not give attention to the first requirement of behaviour subsumption. Indeed, the resulting model can exclude the behaviours of input models in the resulting model after a stakeholder validation. Given the motivation of this work, i.e., consolidating multiple process version into a single common reference model, this approach does not satisfy neither the second requirement of keeping track on the origin of the element of the reference model nor the third requirement of deriving input models from the merged one.

6.2.3. Merging Event Driven Process Chains

Gottschalk et al. [7] define an approach exclusively intended for merging models following the EPC notation. This approach consists first of transforming EPCs into a so called abstraction of EPCs, namely function graphs. The second step is the combination of these function graphs by means of set union operations. Finally, they transform back the combined function graph into an EPC. The object in their approach is not to create a configurable EPC, there are no configurable connectors introduced which would allow for extracting one of the original models.

6.2.4. Critique

Gottschalk et al. [7] use behaviour preserving set union operations over function graphs in order to satisfy the first requirement of behaviours subsumption. However, this approach does not allow for the second (i.e., Traceability) nor the third (i.e., Deriving Original Models) requirement. Indeed, the generated merged models do not allow to trace back where an element of the model originates from. Furthermore, they do not provide any possibility to configure the obtained model in order to derive one of the input models.

6.2.5. Merging Process Graphs to create Configurable Models

La Rosa et al. [9,10] propose a technique that satisfies the three requirements. Their technique starts by computing a similarity measure between nodes of pairs of process models. Then, given a mapping between different elements of the original models, they propose a merge operator that computes the Maximum Common Regions (MCR) and then links elements of the second models, which are not in the MCR, to the MCR of the first model.

Table 7
Comparative analysis of business process merging approaches.

Approach	Behavoussupsumption	Traceability	Deriving original models
Process Merging for Version Control [8,44–47]	(−) The merged model does not necessarily include the behaviours of all input models.	(−) The output model allows to roll back to the immediate previous version of the model and not to other ones.	(−) The output model is not configurable.
Merging Event Driven Process Chains [7]	(+) The combination of function graphs does not alter the bahaviour on input models.	(−) The is not traceability to any of the input models.	(−) The output model is not configurable.
Merging Process Graphs to create Configurable Models [9,10]	(+) The fusion of maximum common regions and applied reduction operations are behaviour preserving.	(+) The process arcs are tagged with input models identifiers.	(+) The generated model is configurable and allows to generate either original or new models.
Our Merging Algorithm	(+) All the operations of the merging algorithm are are behaviour preserving.	(+) All the process elements are tagged with input models identifiers.	(+) The generated model is configurable and allows to generate either original or new models.

Similar to the approach presented, they use arc annotations to allow for tracking the origin of an element.

6.2.6. Critique

In this work, the mapping between function items exclusively relies on their labels using approximate semantic matching between them, while our presented approach considers capabilities and domain ontologies. Furthermore, the proposed algorithm has a complexity of $O(|N|^3)$ for merging only one pair of process models [9] where $|N|$ is the number of nodes of the largest model. However, our merging algorithm has a reduced complexity of $O(|N^*P|^2)$, where $|N|$ represents the number of nodes of the largest model and $|P|$ represents the number of properties of the largest capability that is much less than the size of an average process model.

6.2.7. Summary

A summary of the discussed approaches is show in Table 7. The first three lines of this table list the details of previously reviewed approaches while the last line concerns our proposed Merging Algorithm. The proposed Merging Algorithm satisfies the three prementioned requirements as follows:

1. *[Behaviour Subsumption]* : all the operations of the merging algorithm do not remove any work node (i.e., events and function items). Furthermore, the order between these work nodes is preserved along the merging steps. The only removal is carried out during the reduction step when removing trivial connectors. Trivial connectors do not introduce any routing information and consequently they can be removed without altering the behaviour of the model.
2. *[Traceability]* : the initial step of the algorithm starts by tagging each element of the input process models with the identifier of their corresponding model. This is maintained via the function *Tag* that returns the identifier of the model where an item originates from.
3. *[Deriving Original Models]* : the main target of the merging model is to provide models that can be tailored for generating either input model or new ones. The concept of configurable models fulfils this requirement and more precisely via the use of configurable connectors and function items. A possible way to assist users in deriving one of the input models is to keep from the merged model only items tagged with the original model that the user wants to extract.

6.3. Discussion

From these related approaches, we can distinguish other units of analysis that help further compare these approaches and distinguish our contribution:

- *Target modelling language*: all of the reviewed approaches including the proposed merging algorithm of this paper except [7] make use of abstraction into business process graphs so that minor changes can be applied to the proposed approaches to be applicable on other modelling languages.
- *Matching of process elements*: Gerth et al. [8,44–47] focus on matching change operations using exact matching of labels of model elements. Gottschalk et al. [7] relies on exact matching of process elements. Only La Rosa et al. [9,10] and our proposed merging algorithm use approximate semantic matching of labels of process elements. Furthermore, the proposed merging algorithm uses ontologies and capabilities for matching functions of process models.
- *Complexity*: it is only discussed by La Rosa et al. [9], they mention that their algorithm has a complexity of $O(|N|^3)$ for merging only one pair of process models where $|N|$ is the number of nodes of the largest model.

The proposed Merging algorithm in this paper is linear depending on the size of the largest input business process graph. The first step of the algorithm consists of transforming both input models into two configurable models is a simple revisit to all the models' nodes and thus has the complexity of $O(|N|)$ where $|N|$ represents the number of work nodes of the largest input model. Without using the semantic similarity between event nodes, their merging starts by matching from both models the corresponding events which is bound to the number of nodes and thus has the complexity of $O(|N|^2)$. With respect to matching and merging function items, the matching operation is similar to the events matching and consequently has the complexity of $O(|N|^2)$; the merging step involves the merging of their corresponding capabilities that is bound to the number of properties p a capability can have which corresponds to the complexity of $O(|P|^2)$ where P is the maximum number of properties of a given capability. Consequently the complexity of the function items matching and merging is $O(|N^*P|^2)$. The merging of arcs is bound to the number of arcs of the largest model which also corresponds to the number of nodes of the model and thus the complexity is $O(|N|)$. The post-processing steps has the complexity $O(|N|)$ as it is a simple loop over the nodes of the merged model while the reduction step has the complexity $O(|N|^2)$ as at each node, all neighbour are visited.

In the worst-case, the complexity of our merging algorithm is $O(|N^*P|^2)$, where $|N|$ represents the number of nodes of the largest model and $|P|$ represents the number of properties of the largest capability used in the input models. This is the complexity of the matching and merging of functions that dominates the complexity of the other steps.

7. Conclusion

This paper discussed the idea of early integration of business capabilities in process models in order to be used for creating capability-annotated configurable process models. The paper proposes an algorithm that takes as input a set of capability-annotated process models and outputs the corresponding capability-annotated configurable model. The resulting model should subsume the behaviour of the input models and allows to derive either these input models or other new ones. This feature is fulfilled by the concept of configurable model that has been extended in this paper with configurable capabilities. The resulting capabilities can be used at a later step for driving the configuration of such models.

The proposed algorithm has been evaluated using real world business process variants that have been manually created and annotated. Two main dimensions were considered for the evaluation: time and compression rate. These two metrics were used by previous researchers to evaluate similar contributions. Furthermore, we carried out semi-structured interviews to assess the usefulness of the proposed merging algorithm for designing configurable process models integrating business capabilities. Results show that the approach is promising but further research is needed to reach a certain level of maturity to facilitate the adoption of this work in enterprise settings.

The use of capabilities in the configuration of reference process models as discussed in this paper is intended towards the design of configurable models that capture variation options in terms of capabilities properties. Future work in the configuration phase is needed and ideas of explorations are as follow:

- The configuration-based modelling introduces two steps in the modeling phase of business process management: (1) design of configurable models and (2) configuration and individualisation. The work in this paper contributes to the first step, while the second step has not been tackled. Future works should include (1) formally defining of the configuration and individualisation phase, (2) identifying the configuration dependencies in order to direct the user to a starting configuration point and take subsequent configuration decisions, (3) controlling the configuration steps to ensure correct results, (4) enhancing the user experience during the configuration, (5) recommending possible configurations via process mining techniques, etc.
- The main assumption of this paper is the integration of business capabilities in process models using a well structured model. The model can be further extended to model other aspects of the information systems: roles, resources, costs, etc. These aspects can also be used for driving the configuration of business processes and identify the impact of configuration decision when for example changing roles, substituting available resources, changing a certain supplier, etc.

Acknowledgement

This publication received the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

References

[1] R. Braun, W. Esswein, Classification of reference models, in: Advances in Data Analysis, in: Studies in Classification, Data Analysis, and Knowledge Organization, Springer, 2007, pp. 401–408.
[2] P. Fettke, P. Loos, J. Zwicker, Business process reference models: survey and classification, in: Proceedings of Business Process Management Workshops, in: Lecture Notes in Computer Science, 3812, Springer, 2005, pp. 469–483.
[3] J.M. Küster, J. Koehler, K. Ryndina, Improving business process models with reference models in business-driven development, in: Proceedings of Business Process Management Workshops, in: Lecture Notes in Computer Science, 4103, Springer, 2006, pp. 35–44.
[4] M. Rosemann, W.M.P. van der Aalst, A configurable reference modelling language, Inf. Syst. 32 (1) (2007) 1–23.
[5] F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, Mining reference process models and their configurations, in: On the Move to Meaningful Internet Systems: OTM 2008, in: Lecture Notes in Computer Science, 5333, Springer, 2008, pp. 263–272.
[6] N. Assy, N.N. Chan, W. Gaaloul, An automated approach for assisting the design of configurable process models, IEEE T. Serv. Comput. 8 (6) (2015) 874–888.
[7] F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, Merging event-driven process chains, in: On the Move to Meaningful Internet Systems: OTM 2008, in: Lecture Notes in Computer Science, 5331, Springer, 2008, pp. 418–426.
[8] J.M. Küster, C. Gerth, A. Förster, G. Engels, A tool for process merging in business-driven development, in: Proceedings of the Forum at the CAiSE'08 Conference, in: CEUR Workshop Proceedings, 344, CEUR-WS.org, 2008, pp. 89–92.
[9] M. La Rosa, M. Dumas, R. Uba, R. M. Dijkman, Business process model merging: an approach to business process consolidation, ACM Trans. Softw. Eng. Methodol. 22 (2) (2013) 11.
[10] M. La Rosa, M. Dumas, R. Uba, R.M. Dijkman, Merging business process models, in: On the Move to Meaningful Internet Systems: OTM 2010, in: Lecture Notes in Computer Science, 6426, Springer, 2010, pp. 96–113.
[11] W. Derguech, S. Bhiri, An automation support for creating configurable process models, in: Web Information System Engineering - WISE 2011, in: Lecture Notes in Computer Science, 6997, Springer, 2011, pp. 199–212.
[12] M. La Rosa, Managing Variability in Process-Aware Information Systems, Queensland University of Technology, Brisbane, Australia, 2009 (Ph.D. thesis).
[13] F. Gottschalk, Configurable Process Models, Technische Universiteit Eindhoven, Eindhoven, Netherlands, 2009 (Ph.D. thesis).
[14] M. Dumas, L. García-Bañuelos, M. La Rosa, R. Uba, Fast detection of exact clones in business process model repositories, Inf. Syst. 38 (4) (2013) 619–633.
[15] S. Dutta, O. Narasimhan, S. Rajiv, Conceptualizing and measuring capabilities: methodology and empirical application, Strat. Manag. J. 26 (3) (2005) 277–285.
[16] R. Amit, P.J.H. Schoemaker, Strategic assets and organizational rent, Strat. Manag. J. 14 (1) (1993) 33–46.
[17] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer, 2007.
[18] OASIS reference model for service oriented architecture 1.0, 2006, (http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf). (accessed: 16.04.17).
[19] P. Oaks, A.H.M. ter Hofstede, D. Edmond, Capabilities: Describing What Services Can Do, in: Lecture Notes in Computer Science, 2910, Springer, 2003, pp. 1–16.
[20] W. Derguech, S. Bhiri, Modelling, interlinking and discovering capabilities, in: Proceedings of ACS International Conference on Computer Systems and Applications, AICCSA 2013, 2013, pp. 1–8.
[21] S. Bhiri, W. Derguech, M. Zaremba, Modelling Capabilities as Attribute-Featured Entities, in: Proceedings of Web Information Systems and Technologies - WEBIST 2012, Revised Selected Papers, in: Lecture Notes in Business Information Processing, 140, Springer, 2012, pp. 70–85.
[22] OMG - Object Management Group, 1989, (http://www.omg.org/).
[23] B. Weber, J. Mendling, M. Reichert, Flexibility in process-aware information systems (proflex) workshop report, in: Proceedings of 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises WETICE 2006, IEEE Computer Society, 2006, pp. 269–270.
[24] SAP, Automatically Approve Travel Requests, 2006a, (https://help.sap.com/doc/b8dcdc53b5ef424de10000000a174cb4/2.6/en-US/acdcdc53b5ef424de10000000a174cb4.html). (accessed: 16.04.17).
[25] SAP, Automatically Approve Travel Requests, 2006b, (https://help.sap.com/doc/b8dcdc53b5ef424de10000000a174cb4/2.6/en-US/acdcdc53b5ef424de10000000a174cb4.html). (accessed: 16.04.17).
[26] SAP, Workflow Scenarios in Travel Management, 2006c, (https://help.sap.com/doc/b8dcdc53b5ef424de10000000a174cb4/2.6/en-US/5551cb5389d37214e10000000a174cb4.html). (accessed: 16.04.17).
[27] W. Sadiq, M.E. Orlowska, On correctness issues in conceptual modelling of workflows, in: Proceedings of the Fifth European Conference on Information Systems, ECIS 1997, Cork Publishing Ltd, 1997, pp. 943–964.
[28] R. Sahay, R. Fox, A. Zimmermann, A. Polleres, M. Hauswirth, A methodological approach for ontologising and aligning health level seven (HL7) applications, in: Availability, Reliability and Security for Business, Enterprise and Health Information Systems, in: Lecture Notes in Computer Science, 6908, Springer, 2011, pp. 102–117.
[29] S. O'Riain, B. Coughlan, P. Buitelaar, T. Declerck, U. Krieger, S.M. Thomas, Cross-lingual querying and comparison of linked financial and business data, in: The Semantic Web: ESWC 2013 Satellite Events - Revised Selected Papers, in: Lecture Notes in Computer Science, 7955, Springer, 2013, pp. 242–247.
[30] C.M. Pereira, A. Caetano, P.M.A. Sousa, Using a controlled vocabulary to support business process design, in: Proceedings of CAiSE Workshops 2011 - Selected Papers, in: Lecture Notes in Business Information Processing, 88, Springer, 2011, pp. 74–84.
[31] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using wikipedia-based explicit semantic analysis, in: Proceedings of the 20th International Joint Conference on Artifical Intelligence, 2007, pp. 1606–1611.
[32] S.T. Dumais, G.W. Furnas, T.K. Landauer, S. Deerwester, R. Harshman, Using latent semantic analysis to improve access to textual information, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1988, pp. 281–285.

[33] A. Freitas, J.a.G. Oliveira, S. O'riain, J.a.C.P. Da Silva, E. Curry, Querying linked data graphs using semantic relatedness: a vocabulary independent approach, Data Knowl Eng 88 (2013) 126–141.
[34] J. Mendling, M. Nüttgens, EPC syntax validation with XML schema languages, in: Proceedings of EPK 2003, GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, 2003, pp. 19–30.
[35] C. Nicolas, K. Ekkart, EPC Tools, Jan 2006, (http://www2.cs.uni-paderborn.de/cs/kindler/research/EPCTools). (accessed: 16/04/2017).
[36] W. Derguech, F. Gao, S. Bhiri, Configurable Process Models for Logistics Case Study for Customs Clearance Processes, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Proceedings of Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part II, Lecture Notes in Business Information Processing, 100, Springer, 2011, pp. 119–130.
[37] J. Mendling, M. Nüttgens, EPC Markup language (EPML): an xml-based interchange format for event-driven process chains (EPC), Inf. Syst. E-Bus. Manag. 4 (3) (2006) 245–263.
[38] F. Gottschalk, T.A.C. Wagemakers, M.H. Jansen-Vullers, W.M.P. van der Aalst, M. La Rosa, Configurable process models: experiences from a municipality case study, in: CAiSE 2009, in: Lecture Notes in Computer Science, 5565, Springer, 2009, pp. 486–500.
[39] R. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer, 2014.
[40] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (2009) 131–164.
[41] J. Mendling, C. Simon, Business process design by view integration, in: Proceedings of BPM Workshops 2006, in: Lecture Notes in Computer Science, 4103, Springer, 2006, pp. 55–64.
[42] E. Rahm, P.A. Bernstein, A survey of approaches to automatic schema matching, VLDB J. 10 (4) (2001) 334–350.
[43] D. Ohst, M. Welle, U. Kelter, Differences between versions of UML diagrams, in: Proceedings of the 11th ACM SIGSOFT Symposium on Foundations of Software Engineering, ACM, 2003, pp. 227–236.
[44] C. Gerth, M. Luckey, Towards rich change management for business process models, Softw.-Trends 32 (4) (2012).
[45] J.M. Küster, C. Gerth, A. Förster, G. Engels, Detecting and resolving process model differences in the absence of a change log, in: BPM 2008, in: Lecture Notes in Computer Science, 5240, Springer, 2008, pp. 244–260.
[46] C. Gerth, J.M. Küster, G. Engels, Language-independent change management of process models, in: A. Schürr, B. Selic (Eds.), Model Driven Engineering Languages and Systems MODELS 2009, Lecture Notes in Computer Science, 5795, Springer, 2009, pp. 152–166.
[47] C. Gerth, M. Luckey, J.M. Küster, G. Engels, Detection of semantically equivalent fragments for business process model change management, in: Proceedings of IEEE International Conference on Services Computing, SCC 2010, IEEE Computer Society, 2010, pp. 57–64.