7-1-2013

Towards A Minimal Cost Of Change Approach For Inductive Reference Model Development

Peyman Ardalani
Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany, peyman.ardalani@iwi.dfki.de

Constantin Houy
Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany, constantin.houy@iwi.dfki.de

Peter Fettke
Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany, peter.fettke@iwi.dfki.de

Peter Loos
Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany, loos@iwi.uni-sb.de

TOWARDS A MINIMAL COST OF CHANGE APPROACH FOR INDUCTIVE REFERENCE PROCESS MODEL DEVELOPMENT

Ardalani, Peyman, Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Campus, Building D3$_2$, 66123 Saarbrücken, Germany, peyman.ardalani@iwi.dfki.de

Houy, Constantin, Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Campus, Building D3$_2$, 66123 Saarbrücken, Germany, constantin.houy@iwi.dfki.de

Fettke, Peter, Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Campus, Building D3$_2$, 66123 Saarbrücken, Germany, peter.fettke@iwi.dfki.de

Loos, Peter, Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Campus, Building D3$_2$, 66123 Saarbrücken, Germany, peter.loos@iwi.dfki.de

Abstract

Business Process Management (BPM) has advanced to be one of the most intensely discussed topics in Information Systems (IS) research. Based on the growing maturity of its concepts, methods and techniques BPM has, furthermore, gained tremendous importance in organizational practice. More and more organizations develop and use individual business process models as a fundament for their operations. As the development of individual business process models is a complex and expensive endeavour, organisations aim at reusing and adapting already existing business process models, so called reference models. However, so far only a limited amount of reference models for selected domains exist, which makes the development of new and reliable reference models a highly relevant issue. In this contribution, we present a new heuristic approach for the inductive development of reference process models based on existing individual process models. Our approach focuses on minimal cost of change (MCC) (based on minimal graph edit distance) when adapting the developed reference model to match the underlying individual process models. Furthermore, an according software prototype supporting automated inductive reference model development is presented as a proof of concept. Finally, our approach is compared to other state of the art approaches in this field in order to qualitatively evaluate its feasibility and usefulness.

Keywords: Business Process Management, BPM, Reference Modeling, Reference Models.

1 Introduction

Business Process Management (BPM) has advanced to be one of the most intensely discussed topics in Information Systems (IS) research (van der Aalst 2013). In addition to the growing maturity of its concepts, methods and techniques, BPM has gained tremendous importance in research as well as in organizational practice. Business process models are the basis for most of the functionalities which are relevant in the BPM life cycle and more and more organizations document and improve their business operations using business process models. As the development of individual business process models is a complex and expensive endeavour, many organisations aim at reusing already existing business process models and at adapting these so-called reference models according to their individual needs (Becker et al. 2011). Using reference models as a basis for the development of individual process models is a promising approach which can provide considerable advantages such as acceleration of the design of individual models, improvement of model quality or improvement of communication about the model content in organizations based on a consistent terminology.

However, even though some more or less established business process reference models for certain domains already exist, the development of new reference models for different domains is still a highly relevant endeavour to support BPM in organisations in different industries. Thus, adequate and well-performing approaches for business process reference models development are of considerable importance. In this context, several approaches for reference model development are known which can be differentiated into *deductive* and *inductive* approaches (Becker et al. 2007). Deductive approaches focus on building up reference models based on a theoretical fundament or mere conceptual considerations and represent the majority of applied approaches for reference model development. In contrast, inductive strategies take existing individual process models or real-world business process execution logs, e. g. from different organizations in a certain domain, as a basis for the development of a reference model. In such inductively developed models, similarities between individual models or process execution logs are considered in order to create a "consensual" and at the same time abstracted model (Walter et al. 2013). Against the background that reference model development is a time-consuming endeavour and that more and more relevant data from different IS is available which can describe processes in organizations (Houy et al. 2011), inductive methods for reference model development provide considerable potential and gain more and more importance.

In literature, a few approaches are presented describing how reference models can be developed in an inductive way. However, these approaches often contain certain heuristic elements resulting in according calculation times or semi-automated elements which cause considerable manual efforts. It is the *goal* of our research to address these shortcomings and to contribute to the current state of the art research in inductive reference model development by investigating the possibilities of further automating and rendering the calculation process more efficient.

This article introduces a *minimal cost of change* (MCC) approach for the inductive development of business process reference models based on the idea of a minimized graph edit distance to match a set of given underlying process models. Furthermore, a software implementation will be presented as a proof of concept. Our approach and implementation will then be qualitatively compared to other current approaches for inductive reference model development and related work in order to evaluate its feasibility and usefulness.

Our contribution takes a *design-oriented research approach* as a basis and is *structured* as follows: In section two the conceptual foundations of reference modeling and the inductive development of reference models are presented and discussed. Section three introduces our MCC reference model development approach and the according algorithms for the prototypical implementation. Section four presents our proof of concept and its evaluation before section five compares the MCC approach to essential related work. Finally, section six concludes the paper.

2 Reference Modeling and Reference Model Development

The terms *reference modeling* and *reference model* have not been consistently defined in literature and there is still a lively discussion about the topic. This discussion shall not be recapitulated in this contribution. In general, business process reference models can be understood as business process models which should fulfil certain criteria and offer certain features. However, these criteria are still under discussion. Referring to Fettke and Loos (2007), we consider the following features as important:

(1) *Reusability*: Business process reference models represent blueprints for the development of process-oriented IS which can be reused in different IS development projects.

(2) *Exemplary practices*: Business process reference models can provide common, good or even best practices describing how business processes are actually designed in practice or how they could or should be designed and executed in order to reach certain goals. In this context, a *descriptive* as well as a *prescriptive* or even *normative* connotation of business process reference models becomes apparent depending on their interpretation.

(3) *Universal applicability*: Business process reference models do not only represent business processes of one particular organization but aim at providing universally applicable business process representations which are valuable for different organizations in a certain domain.

Reference models can provide benefits for both theory and practice. Besides the provision of general descriptions of enterprises, which is especially interesting from a theoretical point of view, practice profits, e. g. from decreases in modeling costs, modeling time and modeling risk as reference models can represent proven solutions (Becker et al. 2011). Furthermore, increases in model quality based on the reuse and adaption of already validated process models can be expected.

In literature several approaches for reference model development are discussed which can, as already mentioned, be differentiated into *deductive* and *inductive* approaches. Most of the established reference models have been initially developed based on deductive approaches which becomes apparent when consulting the *Reference Model Catalog* – an inventory of reference models which is administered by the Institute for Information Systems (IWi) at the DFKI (http://rmk.iwi.uni-sb.de). Deductive approaches use theories and conceptual considerations for developing reference models for certain domains or industries. In contrast to that, inductive approaches use existing real-world process models or process execution logs from organizations in one domain or industry for developing a domain- or industry-specific reference model based on real-world data. Hence, inductive approaches serve for developing *common practice* reference process models in the first place. Referring to Rehse et al. (2013), the following requirements are important for inductive reference process model development methods:

(1) *inductive development*: the method should support a systematic development of a reference process model based on a set of individual process models,

(2) *identification of similarities and commonalities*: the developed reference process model should contain similarities and commonalities of the underlying individual process models,

(3) *abstraction from details*: the developed reference process model should abstract from specialities of the individual process models,

(4) *generativity*: the individual process models should be deducible and developable on the basis of the reference process model and

(5) *support of natural language*: formulations in natural language are an essential part of process models and phenomena like homonymy, synonymy and language-related vagueness typically occur in process models. Thus, a method for inductive reference model development should consider this aspect.

In the following section, our MCC approach for inductive reference model development is presented in more detail. At first, an overview of the approach is given and the relevant definitions are introduced before the algorithm is explained in more detail.

3 The Minimal Cost of Change Approach

3.1 Overview and Definitions

Our minimal cost of change (MCC) approach supports the development of reference models with the minimal cost of change in the sense of a minimized graph edit distance to match a set of given underlying process models. In this contribution, we use *Event-driven Process Chains* (EPC) for the representation of business processes based on the following definition:

Definition 1 – EPCsSet: *Let EPC_i be a five-tuple (E_i, F_i, C_i, l_i, A_i) in which:*

- *E_i is the set of events ($E_i \neq \emptyset$);*
- *F_i is the set of functions ($F_i \neq \emptyset$, $F_i \cap E_i = \emptyset$);*
- *C_i is the set of connectors ($C_i \cap E_i = \emptyset$, $C_i \cap F_i = \emptyset$);*
- *l_i: $C_i \rightarrow \{and, or, xor\}$ labels connectors with their type;*
- *$A_i \subseteq (E_i \cup F_i \cup C_i) \times (E_i \cup F_i \cup C_i)$ is the set of arcs.*

EPCsSet is set of all given EPCs (EPCsSet = {EPC_1, EPC_2, ..., EPC_n}).

The following description of the MCC approach shows how underlying business process models are processed and assigned to different types of sets. Based on these sets the cost of change for each model element is calculated and an appropriate reference model is developed. In order to perform these calculations, we deal with sets and functions which are defined in the following:

Definition 2 – NodesSet: *Let E_{all} denote the set of all events in EPCsSet ($\cup E_i$) and F_{all} denote the set of all functions in EPCsSet ($\cup F_i$). Then NodesSet $=E_{all} \cup F_{all}$ and contains all events and functions in EPCsSet. Each member of NodesSet represents an event or a function and is called node.*

Definition 3 - cost function: *Assume that we can insert, delete and move nodes in the reference model and let O be the set of these operations (O={ins, mov, del}). cost: $O \times NodesSet \rightarrow \mathbb{N}$ is a function which indicates the cost of these operations for each node.*

Although these costs can differ for each node, in the following, we assume that all nodes have similar costs. These values are exemplarily chosen as follows: cost(ins) = 10, cost(mov) = 5 and cost(del) = 1. This means that inserting a node into a reference model has a ten times higher priority than deleting it.

Definition 4 – ElementsSet: *Let ElementsSet denote the set of all elements. An element is a tuple (node, f) which describes a node and its frequency of occurrence in EPCsSet:*

- ***f = f(element)**: Let $\mathbf{w_i}$ be the frequency of occurrence of EPC_i in EPCsSet ($\Sigma w_i = 1$) and exist be a function ($ElementsSet \times EPCsSet \rightarrow \{0,1\}$) which indicates if the element exists in EPC_i:*

$$exist(element, EPC_i) = \begin{cases} 1 & \text{if element's node exists in the } EPC_i \\ 0 & \text{if element's node does not exist in } EPC_i \end{cases} \quad (1)$$

Then f(element): ElementsSet→[0,1] is a function which returns the frequency of an element:

$$f(element) = \sum_{i=1} w_i * exist(element, EPC_i) \quad (2)$$

Definition 5 – RelationsSet: *Let RelationsSet denote the set of all relations. A relation describes an element and the connection – directly or via a particular connector – to its preceding element (preelement). A relation is a five tuple (element, preelement, ctype, f, costValue) in which:*

- ***preelement** is an element which directly or via a connector precedes an element;*
- ***ctype** represents the type of connector (ctype $\in$ {AND, OR, XOR, undefined});*
- ***f = f(relation)**: Let $\mathbf{w_i}$ be the frequency of occurrence of EPC_i in EPCsSet ($\Sigma w_i = 1$) and exist be a function ($RelationsSet \times EPCsSet \rightarrow \{0,1\}$) which indicates if the relation exists in EPC_i:*

$$exist(relation, EPC_i) = \begin{cases} 1 & \text{if relation exists in the } EPC_i \\ 0 & \text{if relation does not exist in } EPC_i \end{cases} \quad (3)$$

Then f(relation): RelationsSet→[0,1] is a function which returns the frequency of a relation:

$$f(relation) = \sum_{i=1} w_i * exist(relation, EPC_i) \quad (4)$$

- ***costValue = costValue(relation):*** *RelationsSet→ℕ is a function which indicates the cost which will be saved when we add an element at a certain position in the reference model. However, the insertion may cause some other cost-effects while matching the reference model with other process models. Considering these factors costValue is calculated as follows:*

$$\begin{aligned} costValue(relation) = {} & f(element)*cost(ins) - \overbrace{(f(element) - f(relation))*cost(mov)}^{move} \\ & - \underbrace{(1 - f(relation))*cost(del)}_{delete} - \underbrace{(1 - exist(preelement))*f(relation)*cost(mov)}_{preelement\_move} \end{aligned} \quad (5)$$

*In this formula, the function **exist(preelement):** ElementsSet→{0,1} refers to the existence of a preelement in the ReferenceModelRelationsSet (see definition 6) and is defined as follows:*

$$exist(preelement) = \begin{cases} 1 & \text{if preelement exists in the ReferenceModelRelationsSet} \\ 0 & \text{if it does not exist in the ReferenceModelRelationsSet} \end{cases} \quad (6)$$

In our approach, we extract the cost of change for each element from the *cost function* and calculate the *costValue* of each relation to prioritize the *candidate relations* for the reference model. These candidate relations will be stored in one of the following sets. The final reference model will be generated based on these sets.

Definition 6 - ReferenceModelRelationsSet: *The ReferenceModelRelationsSet is defined exactly like RelationsSet. It includes the relations which are chosen to be inserted into the final reference model. Selected relations will be moved from the RelationsSet into the ReferenceModelRelationsSet.*

Definition 7 – ReservedRelationsSet: *The ReservedRelationsSet is defined exactly like RelationsSet. It includes the relations selected to appear in the final reference model but whose preelement does not yet exist in the ReferenceModelRelationsSet in the current step of the algorithm. These relations will be kept in this set and moved into the ReferenceModelRelationsSet after the preelement has been added.*

As we can see in the general overview of the MCC algorithm *(algorithm 1)*, the MCC approach comprises three main steps: In step 1, *ElementsSet* and *RelationsSet* are initialized based on given process models. In step 2, the candidate relation with the highest costValue is identified and added to the relevant sets (*ReferenceModelRelationsSet* or *ReservedRelationsSet*). Then, its related relations are recalculated. This step contains a loop which is continued until a given threshold is met or all existing relations have been moved from *RelationsSet.* Finally, in step 3 the reference model is created.

```
begin
    /* Step 1: Initiation of sets*/
    InitiateSets();
    /* Step 2: Creation of the ReferenceModelRelationsSet */
    do
        candidate relation = getMaximumRelation();
        AddToRelevantSets(candidate relation);
        RecalculateRelatedRelations();
    while candidate relation.costValue > threshold or candidate relation <> null
    /* Step 3: Creation of the Reference Model*/
    CreateReferenceModel();
end
```

Algorithm 1. General overview of the MCC algorithm

Although the MCC approach especially focuses on providing an *abstracted* reference model which contains the most relevant elements of the underlying process models, the algorithm is also able to present a completely integrated model containing all nodes of the underlying process models if no *abstraction threshold* is defined. To shed more light on input and output of our approach an example is shown in *figure 1*. Three EPCs with different *frequencies of occurrence* ($\boldsymbol{w_i}$) in a model variant collection (*EPCsSet*) represent the input data. We can expect our approach – without setting a threshold – to create a common practice reference model like the one on the right hand side which can represent all observable process variants.

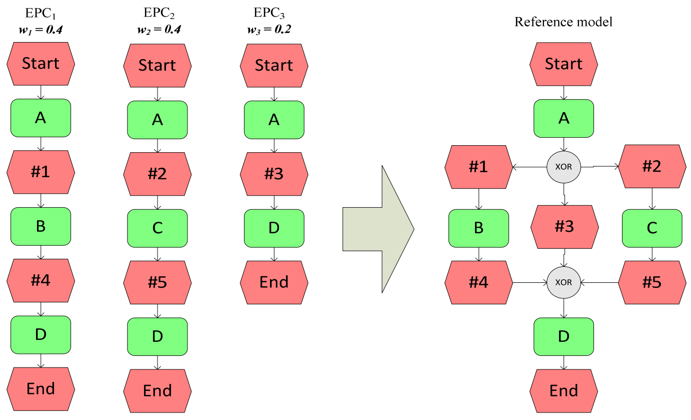


Figure 1. EPCs with different frequencies of occurrence and exemplary reference model

3.2 Step 1: Initiation of sets

Each member of the *ElementsSet* represents an event or a function in the given process models. To initiate this set, all events and functions are extracted from the given process models and their frequencies of occurrence ($\boldsymbol{w_i}$) in the EPCsSet are computed using formula (2). In this paper, we provisionally assume that each node has a clear label which serves for matching similar nodes. Therefore in our simplified setting, the *exist* function according to our definitions returns only 0 or 1. Of course, one could also make use of available mapping functions such as syntactic, semantic, contextual similarity or a combination of them (van Dongen et al. 2008), instead of this simple function to have further model similarity information. Against this background and according to the given models and their frequencies of occurrence in our example (figure 1) the values of *f(A)* and *f(B)* are 1 respectively 0.4.

As already mentioned, each relation in *RelationsSet* shows one element and its preelement (see definition 5). The *RelationsSet* is initialized while we extract the nodes and insert them as elements into the *ElementsSet*. For each element and preelement with a specific *ctype* (connector type) there is one member in this set. Thus, if one element is related to one preelement with different ctypes in EPCsSet, they will be displayed as two members in *RelationsSet*. In case there is no connector between the elements, ctype is *undefined* and the next probable preelement will affect it changing it according to its own ctype. For example, if *event X* is the direct preelement of *function Y* in one EPC with 30% frequency of occurrence and, at the same time, it is the preelement of *function Y* via an *AND* connector in another EPC with 25% frequency, these relations will be mapped into one relation in the RelationsSet with an *AND* ctype and a frequency of 0.55. Moreover, for each element which does not have any preelement, a *null* element will be added as its preelement.

According to the defined formulas, *function A* in figure 1 appears once as an element in *RelationsSet.* Its preelement is the *event Start* with an *undefined* ctype and its frequency is 1; while *function D* existed three times as an element in *RelationsSet* with an *undefined* ctype and different preelements: *event #3* with a frequency of 0.2, *event #4* with 0.4 and *event #5* with 0.4.

The *costValue* is related to defined values in the *cost function* and is the measure for evaluating the priority of each relation. The possible maximum value for this attribute in our example is 10 for *cost(ins)* which shows that the element exists in all models in a certain position. Another important factor in this step is the availability of the preelement in the *ReferenceModelRelationsSet*. If the preelement does not exist in the *ReferenceModelRelationsSet*, the corresponding node is certainly positioned in a "wrong place". Therefore, it needs at least one *move* operation to be positioned in the right place and we accordingly have to reduce the costValue by *cost(mov)*. Hence, as we can see from formula (5), the costValue will be reduced by *f(relation)*cost(mov)* if the preelement does not exist in the current *ReferenceModelRelationsSet*. Table 1 shows all the members of the *RelationsSet*. In the initiation step only the costValue of relations with a null preelement are not reduced by the cost for moving the preelement. For example the *costValue* for *function D* and preelement *event #3* in the first step is:

$$costValue(D,\#3,undefined) = 1*10 - \underbrace{(1-0.4)*5}_{move} - \underbrace{(1-1)*1}_{delete} - \underbrace{0.4*5}_{preelement\_move} = 5$$

By adding the selected relations in each step into the *ReferenceModelRelationsSet*, the costValue of related relations will be increased again. Thus, they will have a higher priority and will also be moved into the *ReferenceModelRelationsSet* in the next steps.

element	preelement	*f*(element)	*f*(relation)	ctype	costValue
Start	(null)	1	1	undefined	10
A	Start	1	1	undefined	5
B	#1	0.4	0.4	undefined	1.4
C	#2	0.4	0.4	undefined	1.4
D	#3	1	0.2	undefined	5
D	#4	1	0.4	undefined	5
D	#5	1	0.4	undefined	5
#1	A	0.4	0.4	undefined	1.4
#2	A	0.4	0.4	undefined	1.4
#3	A	0.2	0.2	undefined	0.2
#4	B	0.4	0.4	undefined	1.4
#5	C	0.4	0.4	undefined	1.4
End	D	1	1	undefined	5

Table 1. The members of RelationsSet and their calculated attributes in the first step

3.3 Step 2: Creation of ReferenceModelRelationsSet

The MCC approach provides a step-by-step technique for the selection of appropriate relations for the reference model. In each round of the second step of algorithm 1, one member of the *RelationsSet* will be transferred into *ReferenceModelRelationsSet* or *ReservedRelationsSet* and the related relations are recalculated. This algorithm either continues in order to meet the *reference model abstraction threshold* or until all available relations in *RelationsSet* have been transferred. If the threshold value is zero ("0") all nodes with a positive costValue are added to the reference model.

In each round of the algorithm's loop, the relation with the highest costValue will be extracted and added to the according set. In contrast to the *getMaximumRelation()* function in algorithm 1, which simply finds the relation with the maximum costValue, the two other functions are a bit more complicated. Thus, we will have a closer look at them. In the *AddToRelevantSets()* function (algorithm 2) we move a relation from *RelationsSet* into the *ReferenceModelRelationsSet*. For this transfer the preelement should exist in the *ReferenceModelRelationsSet.* If it does not exist, we store the candidate relation in the *ReservedRelationsSet* to make use of it in the next steps.

After adding new relations to the *ReferenceModelRelationsSet*, we may face some relations in the *ReservedRelationsSet* which are successors of relations recently added to the *ReferenceModelRelationsSet*. In each step these relations also have to be extracted from *ReservedRelationsSet* and moved to *ReferenceModelRelationsSet*.

```
function AddToRelevantSets(relation)
begin
    /*  check if the preelement exists in ReferenceModelRelationsSet*/
    if exists(preelement) then
        move relation to ReferenceModelRelationsSet;
        /*  add related reserved relations  */
        foreach reservedRelation in ReservedRelationsSet do
            if reservedRelation.preelement = relation.element then
                move reservedRelation to ReferenceModelRelationsSet;
            end
        end
    else
        move relation to ReservedRelationsSet;
    end
end
```

Algorithm 2. Adding candidate relations to relevant sets

After the execution of this function, the *RelationsSet* is updated due to the newly added relations. As considered before, the relations in *ReferenceModelRelationsSet* affect the costValue of the relations in *RelationsSet*. In the *RecalculateRelatedRelations()* function we update the costValues of the relations which are affected by changes in the *ReferenceModelRelationsSet*. In algorithm 3, *relatedRelations* refers to the relations in *RelationsSet* whose preelements have recently been added to the *ReferenceModelRelationsSet*. The costValue of these relations will be increased by the cost for moving the preelement (see formula (5)) and they will have a higher priority to be selected in the next steps.

```
function RecalculateRelatedRelations ()
begin
    relatedRelations = relations having newly added preelement;
    foreach relation in relatedRelations do
        relation.costValue += f(relation) * cost(mov);
    end
end
```

Algorithm 3. Updating the costValue of related relations in the RelationsSet

By meeting the threshold or having transferred all members of *RelationsSet*, this step in the algorithm ends and the *ReferenceModelRelationsSet* contains all required information for creating the reference model. In the next section, this set will be taken as a basis to create an EPC process reference model based on defined EPC modelling rules.

3.4 Step 3: Creation of the Reference Model

After the *ReferenceModelRelationsSet* has been completed, we have a set of relations containing elements and preelements. Thus, there is no element which does not have any preelement (apart from start elements) or which is not linked to other elements. This set also contains information about the *ctype* which determines what kind of connector should be used while developing the process reference model.

In relation to our example in section 3.1, the finally calculated *ReferenceModelRelationsSet* – without considering any threshold – is displayed in table 2. After 13 rounds, all relations in *RelationsSet* have been moved. Some of these relations are at first inserted into the *ReservedRelationsSet* and then moved to the *ReferenceModelRelationsSet*. Relations in the *ReferenceModelRelationsSet* are shown in the particular order in which they have been added to this set. The costValue column shows the last calculated costValue for each relation.

As we can see from table 2, there are some relations in this set which cannot be directly mapped onto a syntactically correct EPC. For example events *#1*, *#2* and *#3* have one preelement (*function A*), which is not possible without an adequate connector. In these cases, we insert fitting connectors to correctly link the nodes in the model.

Round	element	preelement	*f*(element)	*f*(relation)	ctype	costValue
1	Start	(null)	1	1	undefined	10
2	A	Start	1	1	undefined	10
3	#1	A	0.4	0.4	undefined	3.4
4	B	#1	0.4	0.4	undefined	3.4
5	#2	A	0.4	0.4	undefined	3.4
6	#4	B	0.4	0.4	undefined	3.4
7	D	#4	1	0.4	undefined	5
8	End	D	1	1	undefined	10
9	C	#2	0.4	0.4	undefined	3.4
10	#5	C	0.4	0.4	undefined	3.4
11	D	#5	1	0.4	undefined	5
12	#3	A	0.2	0.2	undefined	2.2
13	D	#3	1	0.2	undefined	5

Table 2. The completed ReferenceModelRelationsSet

Moreover, there are some other situations which prevent the *ReferenceModelRelationsSet* from being displayed as a correct EPC model. In our proof of concept we defined several rules to amend the structure of an EPC (table 3). These rules resolve possible invalid relations and normalize the presented reference model according to common EPC modeling rules.

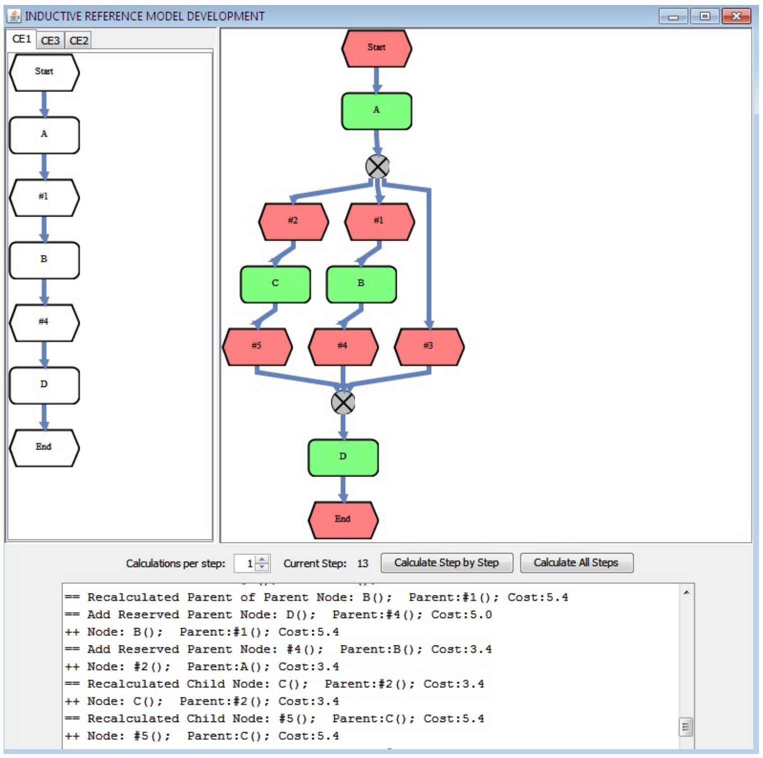
Conflict	**How to resolve it**
Functions are right after "OR" or "XOR" connector	One event should be inserted between each function and connector. One function should be inserted right in front of the connector in order to have an alternating order of functions and events.
First node is a function	A "Start" event should be inserted in front of the first node.
Last node is a function	An "End" event should be added after this function.
Nodes have the same preceding node with different connector types	Add connector between nodes and preceding node according to type of relations: • undefined – undefined : Connector type will be "XOR" • undefined – Other types : Connector type will be like other type • For relations with different connector types, an "OR" connector is added • For relations with same connector type, a connector with that type is added.
Two nodes are after each other with same connector type	Two connectors with the same type will be inserted in front of and after the nodes. For example, these two sequences: A > B > C > D and A > C > B > D will be interpreted to A > (B AND C) > D

Table 3. Rules of interpreting possible conflicts in ReferenceModelRelationsSet for EPCs

4 Proof of Concept and Evaluation

In order to provide a first evaluation of the presented approach and to analyse the obtained results, a proof of concept has been implemented using JAVA. This implementation represents a module of a software prototype for *reference model mining*. EPC models can be processed using the *Event-driven Process Chain Markup Language* (EPML) or the *ARIS Markup Language* (AML) as file format. In this platform, several approaches for inductive reference model development have been implemented. In this paper, however, we concentrate on the presentation of the MCC approach.

As shown in figure 2, the user interface of our proof of concept software comprises three main parts: Given EPCs are shown as a tab panel on the left side in top of the form (*CE1*, *CE2*, *CE3*). The created reference model can be seen on the right side of the tab panel after each processing step. At the bottom of the interface the calculation area is situated. A user can follow the algorithm either step-by-step or directly proceed to the final result. Descriptions of executed calculations in each step are shown in the area below the calculation buttons. There, all execution steps of the algorithm can be traced.



```
== Recalculated Parent of Parent Node: B();  Parent:#1(); Cost:5.4
== Add Reserved Parent Node: D();  Parent:#4(); Cost:5.0
++ Node: B();  Parent:#1(); Cost:5.4
== Add Reserved Parent Node: #4();  Parent:B(); Cost:3.4
++ Node: #2();  Parent:A(); Cost:3.4
== Recalculated Child Node: C();  Parent:#2(); Cost:3.4
++ Node: C();  Parent:#2(); Cost:3.4
== Recalculated Child Node: #5();  Parent:C(); Cost:5.4
++ Node: #5();  Parent:C(); Cost:5.4
```

Figure 2. User interface of our proof of concept, referring to our example in figure 1

Although the created reference models typically provide an abstraction from the underlying individual models, the approach can also generate a "complete" reference model containing all elements (as shown in our example). Using the mentioned step-by-step calculation feature, the threshold can be appropriately configured. After initiating the different sets, only relevant relations will be calculated and updated in each step. Therefore, redundancies are minimized and the speed of obtaining results can be increased in comparison to other approaches (see related work).

Our approach also has some *limitations* which are discussed in the following. First, many reference models have the claim to represent good or best practices, such as the *SAP reference model*, the *Information Technology Infrastructure Library* (ITIL) or the *Supply Chain Operations Reference Model* (SCOR). Hence, many models are considered and interpreted as *normative* models. However, our inductive approach serves for developing *common practice* reference process models in the first place. Such common practice models describe process schema which can indeed be observed in organizations. They are typically considered and interpreted as *descriptive* reference models instead of *normative* models. However, they can support the reuse of common model parts and can, nevertheless, serve as candidates for best practice reference models. Furthermore, our approach, so far, only supports the fundamental elements of EPCs (*functions*, *events*, *connectors*) in the first place. However, other important EPC elements such as *organizational units* or *process paths* have so far not been considered. Hence, it remains to be investigated how the presented algorithm can be further developed in order to handle this. Moreover, we have assumed that each node of a process model can be clearly identified by means of the label. However, there are several problems related to this assumption based on the vagueness of natural language. We aim at improving our approach and making use of available semantic similarity measurement methods for the further development of our approach in order to make it work in more realistic settings. Finally, a useful feature of the MCC approach is that the related cost of change as well as the abstraction threshold can be separately defined and adapted for different needs. However, we need further experience concerning how to define this threshold in order to get valuable common practice reference models which can also serve as good or best practice candidates. In the following, we will compare our approach to related work in the field of inductive reference process model development.

5 Related Work

We investigate three relevant contributions providing approaches for inductive reference model development. They serve as a basis of a comparison with our MCC approach. Li et al. (2010) provide a technique for discovering reference models out of process model variants with minimum efforts which is, thus, quite similar to our approach. However, in contrast to the MCC approach, it requires block-structured process models, which means that sequences and loops have to be represented as blocks with well-defined start and end nodes. Their method aims at merging the different process variants under consideration of internal order relations between the nodes. The block-structured process models are presented in a so-called *order matrix* containing the order relations between each pair of nodes. Based on these matrices, an aggregated order matrix is produced and the most similar nodes in the latter matrix are merged as one block. The new matrices are recalculated and the procedure continues until it meets a similarity threshold. In order to compute these matrices, all possible traces for each variant have to be extracted and all order relations have to be calculated in each step. Although further detailed proof using quantitative comparisons is needed, we can expect this approach to produce higher calculation costs than the MCC approach during the creation of a reference model. Moreover, its necessary precondition restricts its application to block-structured process models, which is not the case with the MCC approach.

La Rosa et al. (2012) suggest an approach to create a model that subsumes a collection of process models. It uses abstracted configurable business process graphs which are independent of a certain modeling technique and merges one pair of processes or sub-processes at a time. In this approach the union of edges is computed and then annotations for each edge are created in order to partition different model regions. In the next step, the most common regions are identified and reconnected to each other in order to form the merged business process graph. Moreover, a set of reduction rules to simplify the merged process graph is presented. However, in contrast to our approach this approach also contains semi-automated steps during the development of the reference model (the so-called "digest extraction") which are not necessary for the MCC approach.

The approach presented by Gottschalk et al. (2008) uses process mining techniques on normalized process execution log files in order to provide configurable process reference models. This method focuses on integrating process model representations based on their execution logs and uses process mining algorithms to extract a reference model from concatenated log files. In contrast to the MCC approach, this technique does not support inductive model development based on existing process models in the first place.

To conclude, our approach can contribute to the current state of the art of inductive reference model development providing a fully automated development of reference process models while offering the possibility to use process models without special preconditions.

6 Conclusion

The development of business process reference models has gained tremendous importance in the last years. Well-performing approaches for their development are, thus, of considerable importance. We have presented a minimal cost of change (MCC) approach for the inductive development of common practice reference models. Furthermore, a software implementation supporting EPCs has been developed as a proof of concept. Then, we have compared the MCC approach to other relevant approaches for inductive reference model development and illustrated its advantages. In the future, we will investigate the applicability of the MCC approach for inductively building up larger reference models in order to quantitatively evaluate its performance. Moreover, an extension of our approach towards the usage and integration of available model similarity measures is planned for the future.

Acknowledgement: The research described in this paper was partly supported by a grant from the German Research Foundation (DFG), project name: “Konzeptionelle, methodische und technische Grundlagen zur induktiven Erstellung von Referenzmodellen (Reference Model Mining)”, support code GZ LO 752/5-1. The authors would also like to thank the anonymous reviewers for their valuable comments which helped to improve this paper.

References

Becker, J., and Meise, V. (2011). "Strategy and Organizational Frame" in: Process Management. A Guide for the Design of Business Processes, J. Becker, M. Kugeler and M. Rosemann (eds.), Springer, Berlin, pp. 91-132.

Becker, J., and Schütte, R. (2007). "A Reference Model for Retail Enterprises" in: Reference Modeling for Business Systems Analysis, P. Fettke and P. Loos (eds.), Idea Group, Hershey, PA, pp. 182-205.

Fettke, P., and Loos, P., eds. (2007). Reference Modeling for Business Systems Analysis. Idea, Hershey, PA.

Gottschalk, F., van der Aalst, W. M. P., and Jansen-Vullers, M. H. (2008). "Mining Reference Process Models and Their Configurations" in: On the Move to Meaningful Internet Systems – OTM 2008 Workshops, LNCS 5333, R. Meersman, Z. Tari and P. Herrero (eds.), Springer, Berlin, pp. 263-272.

Houy, C., Fettke, P., Loos, P., van der Aalst, W. M. P., and Krogstie, J. (2011). Business Process Management in the Large. Business & Information Systems Engineering (BISE) 3 (6), 385-388.

La Rosa, M., Dumas, M., Uba, R., and Dijkman, R. M. (2012). Business Process Model Merging: An Approach to Business Process Consolidation. ACM Transactions on Software Engineering and Methodology (TOSEM) 22 (2).

Li, C., Reichert, M., and Wombacher, A. (2010). The MinAdept Clustering Approach for Discovering Reference Process Models out of Process Variants. International Journal of Cooperative Information Systems 19 (3), 159-203.

Rehse, J.-R., Fettke, P., and Loos, P. (2013). "Eine Untersuchung der Potentiale automatisierter Abstraktionsansätze für Geschäftsprozessmodelle im Hinblick auf die induktive Entwicklung von Referenzprozessmodellen" in: Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI-2013), R. Alt and B. Franczyk (eds.), Leipzig, Germany, pp. 1277-1291.

van der Aalst, W. M. P. (2013).Business Process Management: A Comprehensive Survey. ISRN Software Engineering (Article ID 507984).

van Dongen, B., Dijkman, R., and Mendling, J. (2008). "Measuring Similarity between Business Process Models" in: Advanced Information Systems Engineering – CAISE 2008, LNCS 5074, Z. Bellahsène and M. Léonard (eds.), Springer, Berlin, pp. 450-465.

Walter, J., Fettke, P., and Loos, P. (2013). "How to Identify and Design Successful Business Process Models: An Inductive Method" in: Promoting Business Process Management Excellence in Russia - Proceedings and Report of the PropelleR 2012 Workshop, April 24-26, Moscow, Russia, J. Becker and M. Matzner (eds.), European Research Center for Information Systems, Münster, pp. 89-96.