Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2008 Proceedings

European Conference on Information Systems (ECIS)

2008

Solving the Conflicts of Distributed Process Modelling: Towards an Integrated Approach

Joerg Becker
University of Munster, becker@ercis.de

Daniel Pfeiffer
European Research Center for Information Systems, pfeiffer@ercis.de

SOLVING THE CONFLICTS OF DISTRIBUTED PROCESS MODELLING – TOWARDS AN INTEGRATED APPROACH

Jörg Becker, European Research Center for Information Systems, University of Münster, Leonardo-Campus 3, 48139 Münster, Germany, becker@ercis.de

Daniel Pfeiffer, European Research Center for Information Systems, University of Münster, Leonardo-Campus 3, 48139 Münster, Germany, pfeiffer@ercis.de

Abstract

In this paper the basic structure of an integrated approach is discussed that addresses the major conflicts of distributed business process modelling. Our argumentation is grounded on the assumption that process modelling projects of a practically relevant size involve multiple modellers. Current modelling languages and methods, however, do only provide little support for such distributed modelling scenarios. Simultaneously, the conflicts that can arise in distributed process modelling projects are hardly discussed in the scientific literature. Therefore, this paper aims for two research results: Firstly, based on a formal framework an overview about the main conflicts of distributed process modelling is given. Secondly, a literature review is performed to identify possible solutions for these problems. Based on this the characteristics of an integrated approach are described. Formal modelling and deductive reasoning are applied as research methodology.

Keywords: Distributed Modelling, Process Modelling, Modelling Conflicts, PICTURE.

1 INTRODUCTION

Business process models (BPMs) are an important knowledge source for managerial decision making (Dalal et al. 2004, Davenport & Beers 1995, Fox & Gruninger 1998). BPMs are semi-formal, mostly graphical descriptions of the business processes of an organisation. They are used to describe how things are, could, or should be done to meet the objectives of an organisation. BPMs help to create clarity about the temporal and logical sequence of activities, the resulting products and services, the required resources and data, as well as the involved organisational units (Lindsay & Downs & Lunn 2003, Scheer 2000). Consequently, they have evolved as one of the main tools of business process management, are the basis of many business process reengineering projects, and provide guidance for decision makers.

Business process modelling projects of a practically significant size are distributed (Werth & Walter & Loos 2007). In order to collect decision relevant information about an organisation it is not sufficient to acquire a single process. Rather, it is necessary to create clarity about the process landscape of an organisation (Becker et al. 2006). Therefore, interviews with multiple persons are required to get adequate information. Subsequently, the data gathered has to be explicated in the form of BPMs. The size of most organisations and the corresponding efforts do not allow that these tasks are done by a single person, at one place, and at one time. Consequently, already modelling projects of a moderate size are distributed with regard to personnel, location, and time. In this paper we will focus on the question what impact the involvement of multiple modellers has on the results of a project. We are interested in the preconditions for a uniform representation of the process landscape of an organisation. We will disregard all aspects of location-based or temporal distribution (e.g., Speck & Schnetgöke 2003), consistency (e.g., Quartel & van Sinderen 2007), as well as specific factors of collaboration (e.g., Hoppenbrouwers & Proper & van der Weide 2005, Thomas & Scheer 2006,

Turetken & Demirors 2007, Werth & Walter & Loos 2007) and computer-supported cooperative work (CSCW) (e.g., Grudin 1994).

Popular process modelling languages and methods do not consider aspects of distribution. Languages such as Event Driven Process Chains (EPC) (Scheer 2000), Business Process Modelling Notation (BPMN) (Object Management Group 2006), or IDEF3 (Mayer et al. 1995) do not explicitly address the issue of multiple modellers. This situation is surprising as the quality of business decisions depends directly on the quality of its underlying models. Problems like a deviating terminology, a varying grade of abstraction, or a different understanding of the scope of a process can arise when multiple modellers are involved. These conflicts, in turn, can significantly impair the utility of process models or even lead to wrong managerial decisions.

IS literature has not sufficiently discussed the conflicts of distributed process modelling so far. Currently, processes modellers do not get proper guidance on how to collect and describe process data in a distributed way. A prerequisite in order to change this situation is a comprehensive overview about possible problems whenever different modellers get involved. Conflicts between data models have for example intensively been discussed in literature (e.g., Davis et al. 2003, Hakimpour & Geppert 2001, Kashyap & Sheth 1996, Lawrence & Barker 2001). Corresponding solutions have been proposed based on these findings. In contrast, in the area of business process modelling we are not aware of a conclusive overview that discusses the conflicts of distributed model construction.

Therefore, the following two research questions are addressed in the course of this paper:

(R1) What are the main conflicts of distributed business process modelling?

(R2) What proposals do exist in IS literature to avoid these conflicts and can they be integrated?

In order to answer both questions the paper proceeds as follows: In the next section a formal framework is presented that provides the theoretical basis to derive the conflicts. In the subsequent section a thought experiment is performed that results in a list of the main conflicts of distributed process modelling. This catalogue of conflicts is used in the fourth section of this paper to review the current literature in order to find possible solutions. This search leads to the identification of two different research streams. Based on these results the structure of an integrated approach to solve the distributed modelling conflicts is sketched. The paper closes with a short summary of the results achieved and an outlook to further research.

2 FORMAL ANALYSIS FRAMEWORK

For a rigorous discussion of the conflicts of distributed process modelling a formal analysis framework is required. In this context we define the terms business process modelling grammar (BPMG) and specify a BPM. For our formalisations we use a notation that is based on the work of Pfeiffer (2007), Patig (2004), and Balzer & Moulines & Sneed (1987). As research methodology we apply formal modelling and deductive reasoning.

A $BPMG = \langle C, R, V, G, Z \rangle$ is a mean to describe the flow of activities in an organisation. It consists of the following elements:

1. C is a non-empty set of constructs that comprises nodes as well as edges,
2. R is the set of permitted relations between the constructs with $R \subseteq C \times C$,
3. V is a set of well-formedness rules that restricts the possible process models of the grammar,
4. G is a set of graphical symbols, and
5. Z is a mapping of symbols to constructs with $Z \subseteq C \times G$.

In order to exemplify this formal definition we use the fictitious process modelling grammar Basic Activity Diagram (PMG_{BAD}). PMG_{BAD} can be considered a simplified sub-set of BPMN. It contains the five constructs: *manual activities* ($\mathcal{MA}$), *automated activities* ($\mathcal{AA}$), *annotation objects* ($\mathcal{O}$), *control flows* ($\mathcal{CF}$), and *annotation object flows* ($\mathcal{OF}$). The difference between *manual activities* and

automated activities is that the latter are executed by a machine while the former are performed by a human being. *Annotation objects* are anything that can be attached to a manual or automated activity. Such annotation objects are for example: organisational units, documents, hardware, software, or comments. A *control flow* connects activities. An *annotation object flow* attaches annotation objects to activities. Control flows are directed. Annotation object flows are undirected and can only be drawn from an annotation object to an activity (note that the rules where a flow can be attached do not define its direction). Both kinds of flows are always connected to exactly two nodes, i.e. to its start node and to its end node. Control flows and annotation object flows cannot be further described by domain statements, i.e. it is not possible to label them.

Based on this verbal description $BPMG_{BAD}$ can be formalised. The process modelling grammar $BPMG_{BAD} = \langle C, R, V, G, Z \rangle$ has the following structure:

1. $C = \{\mathcal{MA}, \mathcal{AA}, \mathcal{O}, \mathcal{CF}, \mathcal{OF}\}$,
2. $R = \{(\mathcal{MA}, \mathcal{CF}), (\mathcal{CF}, \mathcal{MA}), (\mathcal{AA}, \mathcal{CF}), (\mathcal{CF}, \mathcal{AA}), (\mathcal{O}, \mathcal{OF}), (\mathcal{OF}, \mathcal{MA}), (\mathcal{OF}, \mathcal{AA})\}$,
3. $V = \emptyset$,
4. $G = \{$  ,  ,  ,→,  $\}$, and
5. $Z = \{(\mathcal{MA},$  $), (\mathcal{AA},$  $), (\mathcal{O},$  $), (\mathcal{CF}, \rightarrow), (\mathcal{OF},$  $)\}$.

$BPMG_{BAD}$ can be used to create BPMs. A $BPM = \langle E, F, S, A \rangle$ is a description of the temporal, logical order of activities in an organisation. Formally, it is defined as:

1. E is a non-empty set of model elements with $E \subseteq C \times \mathbb{N}$,
2. F is a set of relations between the model elements with $F \subseteq E \times E$,
3. S is a set of statements in an applications domain language, and
4. A is a mapping of statements to model elements with $A \subseteq E \times S$.

In order to provide a formal definition of the distributed modelling conflicts, it is helpful to introduce a couple of abbreviations (cf. Table 1).

Abbreviation	Description of the abbreviation
$\tau(e) = \tau((c,n)) = c$	Returns the type c of a model element e.
$E^c = \{e \mid e \in E \wedge \tau(e) = c\}$	Returns the set of model elements with the specific type c.
$E^{\mathcal{N}} = E^{\mathcal{MA}} \cup E^{\mathcal{AA}} \cup E^{\mathcal{O}}$	$E^{\mathcal{N}}$ is the set of nodes in a model BPM_{BAD}.
$\varsigma(e) = \begin{cases} s, & \text{iff } \exists s \in S: (e,s) \in A \\ \varepsilon, & \text{otherwise} \end{cases}$	Returns the domain statement s of a model element e if such a s exists or nothing otherwise.
$D_{\varsigma(e)}$	Represents an arbitrary textual or verbal definition of the domain statement ς(e) that is assigned to the model element e.
$M(D_{\varsigma(e)})$	Contains the set of all interpretations of the definition $D_{\varsigma(e)}$. It defines the extensional (real world) semantics of $D_{\varsigma(e)}$. The extension of $M(D_{\varsigma(e)})$ is specified by a linguistic community.
$e =_{sem} e' \Leftrightarrow M(D_{\varsigma(e)}) = M(D_{\varsigma(e')})$	The domain statements of e and e′ are semantically equivalent.
$e =_{syn} e' \Leftrightarrow \varsigma(e) = \varsigma(e')$	The domain statements of e and e′ are syntactically equivalent.
$e =_{tsem} e' \Leftrightarrow M(D_{\tau(e)}) = M(D_{\tau(e')})$	The types of e and e' are semantically equivalent.
$e =_{tsyn} e' \Leftrightarrow \tau(e) = \tau(e')$	The types of e and e' are syntactically equivalent
$\bullet^c e = \{e' \mid (e,e') \in F \wedge (c \neq \varepsilon \Rightarrow \tau(e') = c)\}$	Returns the set of preceding model elements of e of type c.
$e\bullet^c = \{e' \mid (e',e) \in F \wedge (c \neq \varepsilon \Rightarrow \tau(e') = c)\}$	Returns the set of succeeding model elements of e of type c.
$\odot(M) = \odot(\{m_1, \ldots, m_n\}) = m_1$	Selects an arbitrary element from a set M.
$\circ^c e = \odot(\bullet^c e)$	Returns an arbitrary preceding model element of e of type c.
$e\circ^c = \odot(e\bullet^c)$	Returns an arbitrary succeeding model element of e of type c.

Table 1. Some abbreviations to simplify the deviation of the conflicts.

In the next section, the distributed process modelling conflicts are explained based on a thought experiment. We assume that two modellers have the task to describe the same business process with the grammar $BPMG_{BAD}$. As a result they create the two models $BPM_{BAD} = \langle E, F, S, A \rangle$ and

$BPM'_{BAD} = \langle E', F', S', A' \rangle$. The setting of this thought experiment makes sure that the divergent decisions of the two modellers are made visible. As both of them refer to the same real world process, the resulting conflicts can be analysed ceteris paribus. Consequently, BPM_{BAD} and BPM'_{BAD} are used to derive the problems that in general can arise between two BPMs when multiple modellers are involved. The two models are described in Figure 2.

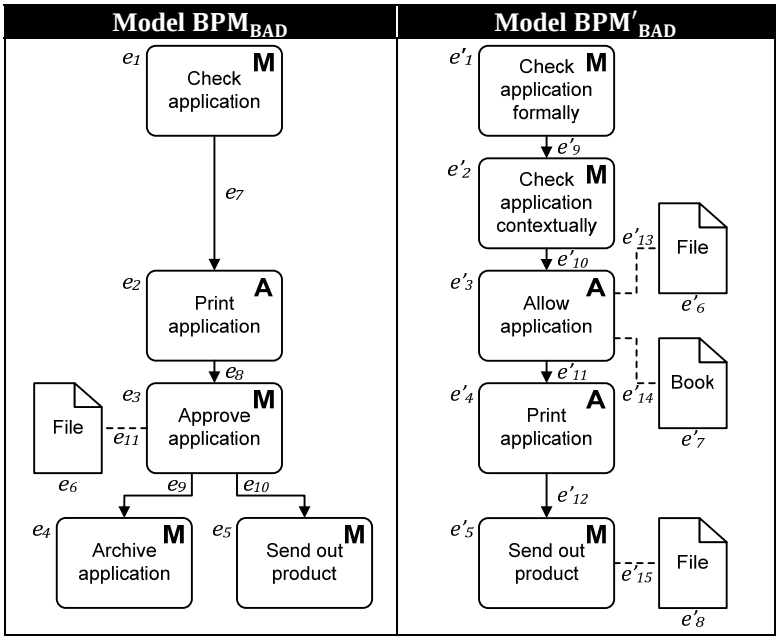

Figure 2. Process Models BPM_{BAD} and BPM'_{BAD}.

3 DISTRIBUTED PROCESS MODELLING CONFLICTS

A *conflict* is a semantic or syntactic deviation between different BPMs that refer to the same or a similar real world phenomenon. Conflicts can be due to two different reasons (Soffer & Hadar 2007). Firstly, they can be caused by a varying perception of the world. Based on intentions and pre-existing knowledge modellers can structure a domain differently or can consider other aspects of it as relevant. Secondly, the conflicts can be caused by deviating decisions during the construction of a BPM. Domain languages and modelling grammars offer certain degrees of freedom to express a given fact. Model creators can utilize this freedom in diverse ways which can lead to conflicts.

It is important to stress that these conflicts are not necessarily unwanted. In large modelling projects it is often helpful to start with an abstract model, to gradually decompose it, and, subsequently, to refine the emerging parts (Soffer & Golany & Dori 2003). Correspondingly, it can be sensible to avoid presenting the same aspects of a model to all target groups (Becker & Delfmann & Knackstedt 2007). However, although, the conflicts may serve a meaningful purpose they become problematic when the objective of a distributed modelling project is to provide uniform and comparable BPMs.

Conflicts between models have intensively been discussed in the database schema matching and integration literature (e.g., Hakimpour & Geppert 2001, Kashyap & Sheth 1996, Lawrence & Barker 2001) as well as in publications about meta modelling (e.g., Rosemann & zur Mühlen 1998) and ontology engineering (Davis et al. 2003). However, to the best of our knowledge there is no comprehensive analysis of the conflicts in the context of business process modelling. In contrast to data models, BPMs are mainly directed graphs with multiple edge types, often instance data is not available, and not all BPMs dispose of a formal semantics. These different characteristics of BPMs motivate to systematically derive their conflicts.

3.1 Semantic Inequality Conflicts

In order to derive the first three conflicts let us assume an arbitrary model element $e \in E^{\mathcal{N}}$ from BPM_{BAD}. Let us further suppose that there is no corresponding model element $e' \in E'^{\mathcal{N}}$ in the model BPM'_{BAD} with the same meaning. This assumption can be formally expressed as:

$$\exists e \in E^{\mathcal{N}}, \forall e' \in E'^{\mathcal{N}} : e \neq_{sem} e'. \quad \text{(A1)}$$

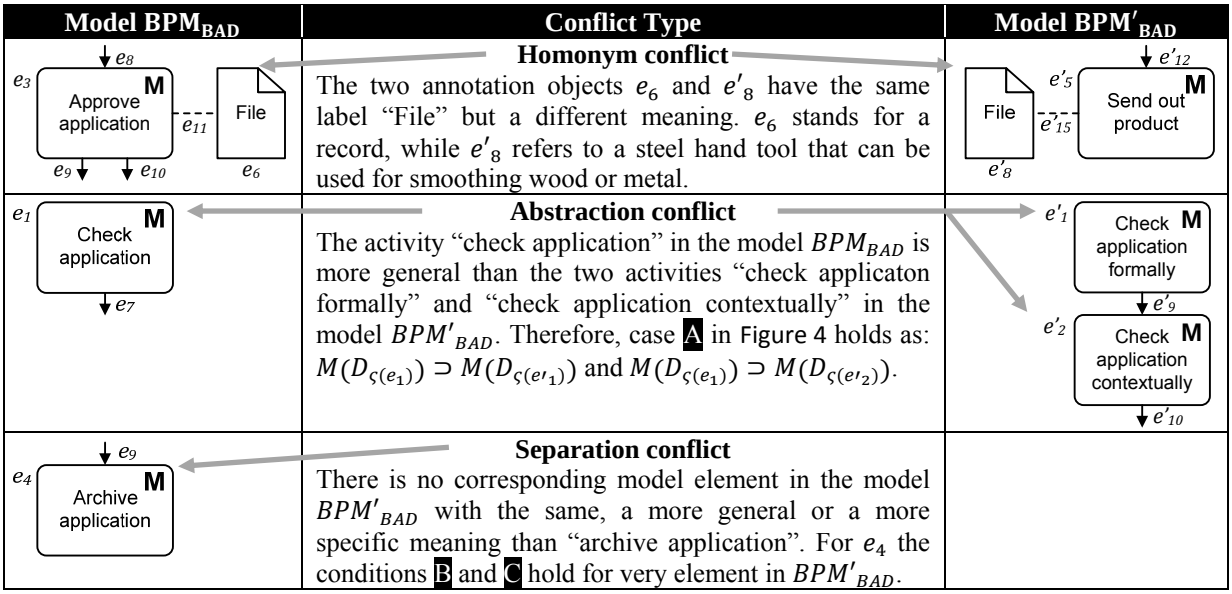

Figure 3. Examples for semantic inequality conflicts.

Firstly, let us consider a *syntactic* equivalence relation between e and an arbitrary $e' \in E'^{\mathcal{N}}$. We assume that: $e =_{syn} e'$. This means that both model elements have the same label while their semantics differs. This is a so called *homonym conflict*. An example for this conflict is given in Figure 3. There are no homonym conflicts between two models, if the following condition holds:

$$\forall e \in E^{\mathcal{N}}, \forall e' \in E'^{\mathcal{N}} : e =_{syn} e' \Rightarrow e =_{sem} e'. \quad \text{(C1)}$$

Secondly, we analyze the possible *semantic* relations between $e \in E^{\mathcal{N}}$ and $e' \in E'^{\mathcal{N}}$ with $e \neq_{sem} e'$. A semantic comparison of their corresponding domain statements can lead to three cases A, B, and C that are depicted in Figure 4. Firstly, it is possible that the statement $\varsigma(e')$ has a strictly more general or more specific meaning than statement $\varsigma(e)$ (cf. case A). For example "check application" and "check application formally". Secondly, there can be a semantic overlapping between the two corresponding statements (cf. case B). For example "check application" and "authorize application formally" as both of them may entail a formal evaluation. Thirdly, the statements can be semantically disjoint (cf. case C). For example "check application" and "send out product". Beneath these three cases there is a fourth one described in Figure 4. This fourth case can be derived based on a further analysis of the case B. Suppose there are two semantically overlapping model elements e and e'_1. However, there is no subset relation between them. Additionally, there is a second element e'_2 in $E'^{\mathcal{N}}$ that together with e'_1 covers the meaning of e. So, in the case D there is more than one model element from BPM'_{BAD} needed in order to describe the semantics of e. For example the activity "write and send postcard" can be explained with the two partially more general activities "write document" and "send document".

The cases A and D describe an *abstraction conflict* between the model elements involved. Abstraction conflicts result from the representation of the application domain at deviating levels of abstraction. They occur when different modellers use more general or more specific domain statements for the same fact. Consequently, two model elements with more general or more specific domain statements

attached indicate an abstraction conflict. The cases B and C represent a *separation conflict*. The model element e has no corresponding counterpart in model BPM'_{BAD} with the same, a more general, or a more specific meaning. Separation conflicts can occur when two modellers have a different scope of the process in mind or consider different model elements as relevant to describe the process. Figure 3 gives an example for each of the two conflicts.

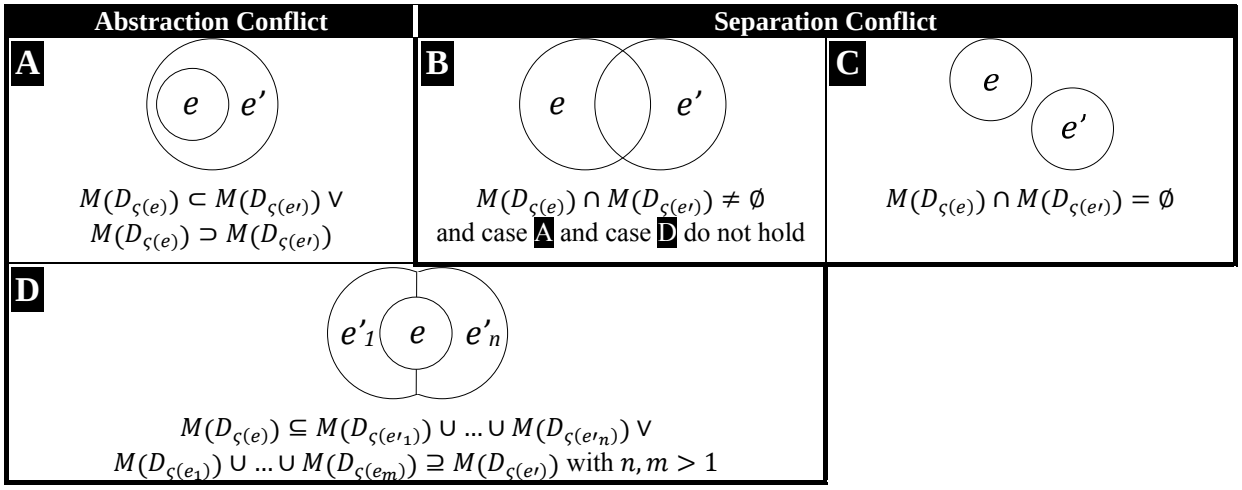

Figure 4. Set theoretic representation of the abstraction and separation conflict.

The example in Figure 3 also shows that *abstraction conflicts* and *separation conflicts* are closely related. Both conflicts can emerge together. Firstly, let us take the case where the meaning of "check application" is exactly covered by "check application formally" and "check application contextually". This is expressed by: $M(D_{\varsigma(e_1)}) = M(D_{\varsigma(e'_1)}) \cup M(D_{\varsigma(e'_2)})$. Under this condition there is only an abstraction conflict and no separation conflict. However, secondly, suppose there is an extra activity such as "check application ethically" that is part of the activity "check application". Consequently, the semantics of e_1 is not completely covered by e'_1 and e'_2. This can be formally described as: $M(D_{\varsigma(e_1)}) \supset M(D_{\varsigma(e'_1)}) \cup M(D_{\varsigma(e'_2)})$. In this situation beneath the abstraction conflict there is an additional separation conflict as some part of the semantics of e_1 is not available in BPM'_{BAD}.

Consequently, when *abstraction* and *separation conflicts* are removed between e and e', i.e. cases A, B, C, and D do not apply, we gain: $e =_{sem} e'$. This, in turn, implies for BPM_{BAD} and BPM'_{BAD} that these two conflicts are avoided for at least one e' for each e and vice versa when it holds that:

$$\left(\forall e \in E^{\mathcal{N}}, \exists e' \in E'^{\mathcal{N}} : e =_{sem} e'\right) \wedge \left(\forall e' \in E'^{\mathcal{N}}, \exists e \in E^{\mathcal{N}} : e' =_{sem} e\right). \quad \text{(C2)}$$

(C2) means that all model elements in $E^{\mathcal{N}}$ have at least one semantically equivalent counterpart in $E'^{\mathcal{N}}$ and vice versa. In the following, we suppose that all three conflicts have been removed.

3.2 Semantic Equality Conflicts

We again assume an arbitrary model element $e \in E^{\mathcal{N}}$ and a corresponding $e' \in E'^{\mathcal{N}}$. However, this time we suppose that these two model elements e and e' have the same meaning and refer to an identical fact of the application domain. This is expressed by: $e =_{sem} e'$. Figure 5 describes the conflicts that can arise, when the two model elements e and e' are compared.

Synonym conflicts arise when two modellers include two different domain statements with the same meaning in the models. *Type conflicts* are the result of choices in the modelling language about what construct to use to represent a certain fact. Type conflicts can occur when two modellers utilise their freedom in a different way. *Control flow* conflicts emerge when two modellers have a deviating structure of the process model in mind. They depend on the modeller's opinion about what activities are performed sequentially or in parallel, where are branchings or loops. *Annotation conflicts* occur

when two modellers describe an activity in a differently detailed form. That means they use a varying number of annotation object flows to characterise an activity. Table 6 provides a formal definition of the four semantic equality conflicts.

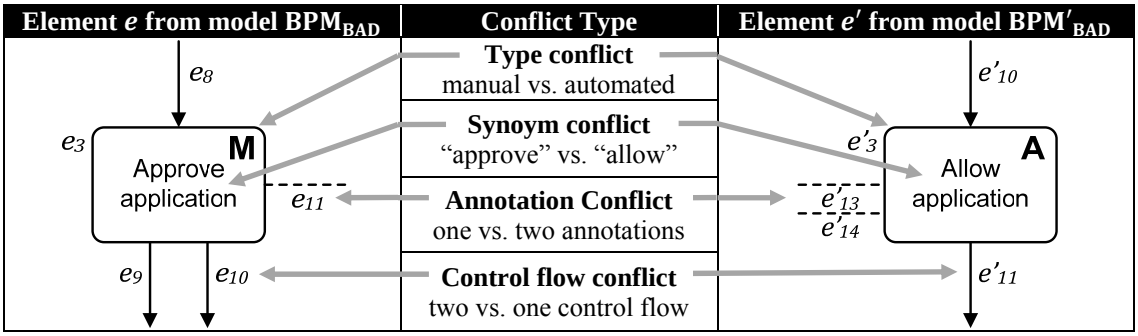

Figure 5. Examples for semantic equality conflicts between BPM_{BAD} and BPM'_{BAD}.

Conflict Type	Formal Definition	Description of the conflict
Synonym conflict	$e \neq_{syn} e'$	The textual statements (labels) connected to the model elements e and e' differ.
Type conflict	$e \neq_{tsyn} e' \vee e \neq_{tsem} e'$	The types of the model elements e and e'differ syntactically or semantically.
Control flow conflict	$\lvert\bullet^{\mathcal{CF}} e\rvert \neq \lvert\bullet^{\mathcal{CF}} e'\rvert \vee \lvert e\bullet^{\mathcal{CF}}\rvert \neq \lvert e'\bullet^{\mathcal{CF}}\rvert$	The number of outgoing or incoming *control flows* differs between e and e'.
Annotation conflict	$\lvert\bullet^{\mathcal{OF}} e\rvert \neq \lvert\bullet^{\mathcal{OF}} e'\rvert \vee \lvert e\bullet^{\mathcal{OF}}\rvert \neq \lvert e'\bullet^{\mathcal{OF}}\rvert$	The number of outgoing or incoming *annotation object flows* differs between e and e'.

Table 6. Formal definition of the semantic equality conflicts.

Based on Table 6 when these four conflicts are resolved, the following condition holds for a pair $e \in E^{\mathcal{N}}$ and $e' \in E'^{\mathcal{N}}$:

$$e =_{sem} e' \Rightarrow e =_{syn} e' \wedge e =_{tsyn} e' \wedge e =_{tsem} e' \wedge \left|\bullet^{\mathcal{CF}} e\right| = \left|\bullet^{\mathcal{CF}} e'\right| \wedge \left|e\bullet^{\mathcal{CF}}\right| = \left|e'\bullet^{\mathcal{CF}}\right| \wedge \left|\bullet^{\mathcal{OF}} e\right| = \left|\bullet^{\mathcal{OF}} e'\right| \wedge \left|e\bullet^{\mathcal{OF}}\right| = \left|e'\bullet^{\mathcal{OF}}\right|. \quad \text{(C3)}$$

Consequently, a pair of semantically equivalent model elements e and e' shares the same type, name, and number of incoming and outgoing edges. In the following we suppose that the models have been transformed in order to meet conditions (C1), (C2), and (C3) (cf. Figure 7).

3.3 Order Conflict

The control flow of BPMs spans an unidirectional graph. Consequently, based on the direction of the graph all activities in a process model have a specific order. This order is shaped by the modellers depending on their understanding of the process. Activities can be performed sequentially or in parallel (Scheer 2000). It is also possible that two activities can have an arbitrary order but are not allowed to be performed in parallel (Priemer 1995, van der Aalst et al. 2003). Assume for example a person who prints an application first and approves the application afterwards. Further, consider the application is printed for documentation purposes only. Hence, there is no contextual reason for this particular order. A different modellers could represent the same fact just the other way round. This can lead to an order conflict as depicted in Figure 7.

The order of a BPM depends on its incoming and outgoing *control* as well as *annotation object flows* To preserve a specific order the structure of BPM_{BAD} and BPM'_{BAD} must match. Partially, this has already been done by resolving the control flow and annotation conflicts. Hence, the control flow e_9 and the annotation flow e'_{14} have been removed from Figure 7. However, not only the number of the

flows is relevant but also what other model elements they are assigned to. Again, let us assume that there is an arbitrary model element $e \in E^{\mathcal{N}}$ and a corresponding $e' \in E'^{\mathcal{N}}$ with $e =_{sem} e'$. An *order conflict* can be defined as:

$$(\exists o \in e\bullet^{\mathcal{CF}}, \forall o' \in e'\bullet^{\mathcal{CF}}: o\circ \neq_{sem} o'\circ) \vee (\exists i \in \bullet^{\mathcal{CF}}e, \forall i' \in \bullet^{\mathcal{CF}}e': \circ i \neq_{sem} \circ i') \vee (\exists g \in \bullet^{\mathcal{OF}}e, \forall g' \in \bullet^{\mathcal{OF}}e': \circ g \neq_{sem} \circ g'). \quad \text{(C4)}$$

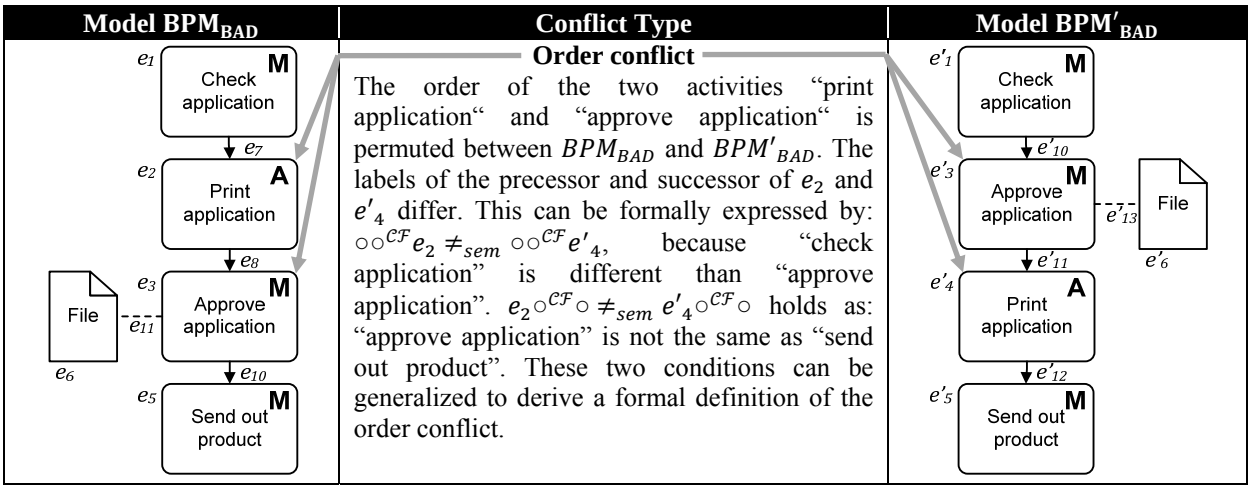

Figure 7. Example for an order conflict between BPM_{BAD} and BPM'_{BAD}.

This definition states that there is an *order conflict* when the incoming or outgoing *control flows* of e and e' are not connected to semantically equivalent model elements. There is also an order conflict when the annotation objects attached to an activity semantically differ between the two models. In other words, there is an order conflict if the environment of e and the environment of e' are not the same. Based on (C3) and (C4) the following implication for two model elements with a semantically equivalent domain statement can be derived:

$$\begin{aligned} e =_{sem} e' \Rightarrow\ & e =_{syn} e' && \text{[a]} \\ & \wedge\ e =_{tsyn} e' && \text{[b]} \\ & \wedge\ e =_{tsem} e' && \text{[c]} \\ & \wedge\ \left|\bullet^{\mathcal{CF}}e\right| = \left|\bullet^{\mathcal{CF}}e'\right| \wedge \left|e\bullet^{\mathcal{CF}}\right| = \left|e'\bullet^{\mathcal{CF}}\right| && \text{[d]} \\ & \wedge\ \left|\bullet^{\mathcal{OF}}e\right| = \left|\bullet^{\mathcal{OF}}e'\right| \wedge \left|e\bullet^{\mathcal{OF}}\right| = \left|e'\bullet^{\mathcal{OF}}\right| && \text{[e]} \\ & \wedge\ (\forall o \in e\bullet^{\mathcal{CF}}, \exists o' \in e'\bullet^{\mathcal{CF}}: o\circ =_{sem} o'^{\circ}) && \text{[f]} \\ & \wedge\ (\forall i \in \bullet^{\mathcal{CF}}e, \exists i' \in \bullet^{\mathcal{CF}}e': \circ i =_{sem} \circ i') && \text{[g]} \\ & \wedge\ (\forall g \in \bullet^{\mathcal{OF}}e, \exists g' \in \bullet^{\mathcal{OF}}e': \circ g =_{sem} \circ g') && \text{[h]} \end{aligned} \quad \text{(C5)}$$

Condition (C5) makes sure that two model elements e and e' with a semantically identical domain statement have: [a] the same label, [b, c] the syntactically and semantically same type, [d, e] the same number of incoming and outgoing flows, and [f, g, h] all activities and annotation objects that are connected to e have a semantically corresponding model element that is attached to e'. When [d, e] holds [f, g, h] holds vice versa also for the element e'. Due to (C1) and (C3) when synonym and homonym conflicts are resolved it holds that: $e =_{sem} e' \Leftrightarrow e =_{syn} e'$. Thus, based on (C5) in combination with (C2) a condition can be derived that is only true when *semantic inequality, equality*, and *order conflicts are* resolved for each $e \in E^{\mathcal{N}}$ and at least one $e' \in E'^{\mathcal{N}}$:

$$\left(\forall e \in E^{\mathcal{N}}, \exists e' \in E'^{\mathcal{N}}: e =_{syn} e'\right) \wedge \left(\forall e' \in E'^{\mathcal{N}}, \exists e \in E^{\mathcal{N}}: e' =_{syn} e\right) \quad \text{(C6)}$$

Expressions (C5) and (C6) can be considered a syntactic and semantic equivalence criterion for BPM_{BAD} and BPM'_{BAD}. Indeed it is closely related to similar criteria that can be found in literature (Pfeiffer 2007). Consequently, it can be assumed that if it is possible to resolve or prevent these eight

conflicts, semantically equivalent models arise. This is a strong argument in favour of the assumption that these eight conflicts are sufficient to describe deviations in a distributed business process modelling project. Thus, by systematically deriving the conflicts the thought experiment has led us to a condition for syntactically and semantically equivalent process models. As a consequence, when these conflicts can be avoided in the first place uniform and comparable BPMs can be achieved by different modellers. Possible approaches are discussed in the next section.

4 APPROACHES TO AVOID THE CONFLICTS

In recent years two different research streams have emerged to cope with distributed modelling conflicts. Both of them aim at reducing the degrees of freedom of the modeller and, thus, to decrease the subjectivity of the model construction. However, so far these two perspectives have mainly been discussed separately from each other in the IS literature. In the following we will analyse the advantages and drawbacks of the two approaches. Based on the conflicts we will show that the two approaches can fruitfully be combined.

The first perspective is taken by the *ontology-based approach* (e.g., Guizzardi & Pires & Sinderen 2002, He & Ling 2006, Höfferer 2007, Pfeiffer & Gehlert 2005, Simon & Mendling 2007). In this approach the issue of subjectivity of different modellers is addressed by offering a common terminological reference point in the form of a domain ontology. Domain ontologies are an intensively discussed measure in IS to capture the common knowledge of a certain part of reality (e.g., Chandrasekaran & Joesephson & Benjamins 1999, Guizzardi & Pires & Sinderen 2002, Uschold 1998, Wimmer & Wimmer 1992). They provide a set of shared concepts that describes what exists in this specific domain and formalise the relevant vocabulary (Evermann 2005). Therefore, they have been suggested as a mechanism to systematically guide the construction of process models and conceptual models in general (Guizzardi & Pires & Sinderen 2002, Mylopoulos 1998, Storey 2005). Through a semantic annotation with elements from an ontology, BPMs are underpinned with the shared conceptual vocabulary of a specific domain. For this purpose the selection of labels of the model elements is governed and restricted by the ontology. This means that the names of the labels cannot freely be chosen but have to be taken from the ontology or have to reference it. Figure 8 gives an example for a mapping between a domain ontology and two BPMs.

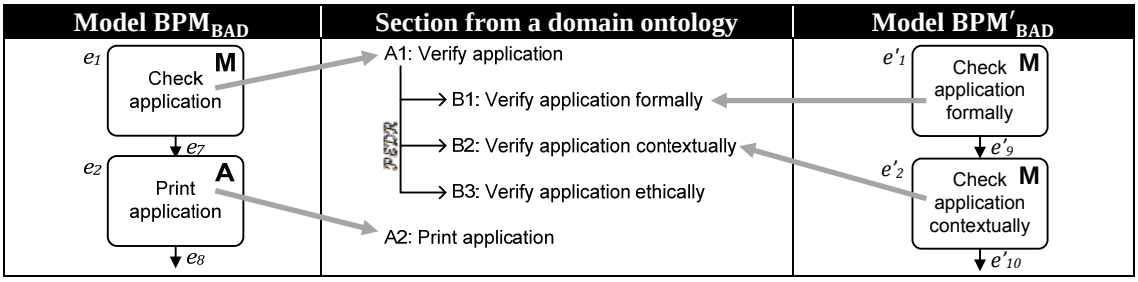

Figure 8. Mapping of a domain ontology to BPM_{BAD} and BPM'_{BAD}.

Depending on their degree of formalisation and expressiveness different representational means for (domain) ontologies have been proposed (McGuinness 2003, Uschold & Gruninger 2004). Ontologies can for example be described in the form of glossaries, thesauri, taxonomies, technical term models, data models, or by using a logic-based language. For the purpose of this paper it is sufficient to assume that an ontology can at least represent homonym ($\mathcal{HOR}$) and synonym ($\mathcal{SYR}$) relations, specialisation relations ($\mathcal{SR}$), as well as a hierarchical decomposition of domain statements based on a object-oriented ($\mathcal{OBDR}$), performance-oriented ($\mathcal{PEDR}$), or process-oriented ($\mathcal{PRDR}$) criterion (Scheer 2000).

A *domain ontology* $DO = \langle OS, OR \rangle$ can, therefore, formally be defined as:

1. OS is a non-empty set of domain statements and

2. OR is a set of typed relations between the domain statements with $OR \subseteq OS \times OS \times T$ and $T = \{\mathcal{SYR}, \mathcal{HOR}, \mathcal{SR}, \mathcal{OBDR}, \mathcal{PEDR}, \mathcal{PRDR}\}$.

Consequently, the annotation of a BPM with an ontology leads to the condition: $S \subseteq OS$. That means that the vocabulary of the process model S can be entirely explained by the domain statements OS of the ontology. Labels that are not known in the ontology cannot be used. Therefore, with the *ontology-based approach* mainly four types of conflicts can be addressed:

- *Homonym and synonym conflicts*: Homonym ($\mathcal{HOR}$) and synonym ($\mathcal{SYR}$) relations are made explicit in the domain ontology. This makes it feasible to resolve the corresponding conflicts between two process models BPM_{BAD} and BPM'_{BAD} (e.g., Hakimpour & Geppert 2001). By inferring from the ontology and guiding the modeller it can be assured that: $e =_{syn} e' \Leftrightarrow e =_{sem} e'$. Consequently, in (C5) condition part [a] and condition (C1) can be addressed.
- *Abstraction conflicts:* Based on the different decomposition relations ($\mathcal{OBDR}$, $\mathcal{PEDR}$, $\mathcal{PRDR}$) more specific or more general elements in the process models can be identified (cf. case A in Figure 4) (Kashyap & Sheth 1996). Therefore, by inferring from the ontology it can be enforced that all models are created at the same level of abstraction. For example it can be ensured that only statements from level A in Figure 8 such as "verify application" or "print application" are used. Hence, the abstraction grade of two process models can be fixed and the identification of semantically matching model elements is simplified. Consequently, condition (C6) is partially addressed by the use of an ontology. In order to discover only incompletely overlapping process parts (cf. case B and D in Figure 3) a more expressive ontology as defined here is needed (e.g., Weber et al. 2007).
- *Separation conflicts:* Separation conflicts can be handled by using an ontology in at least two different ways. Firstly, the ontology restricts the vocabulary of the modeller. Hence, statements that are not part of the ontology cannot be made in the models. Therefore, the ontology can rule out the specification of statements that are not relevant for a specific process. Secondly, an ontology can help to consistently specify the borders of a process. By offering a process catalogue the general structure of the processes can be provided to the modellers. Based on a definition of the process and the specification of its interfaces, different process modellers can come closer to common understanding what a process comprises. Therefore, also here condition (C6) is partially addressed.

A critical issue of this approach is the availability of domain ontologies. In recent years there have been intensive research efforts to develop domain ontologies (Weinberger & Teeni & Frank 2003). However, still, there are only a very few specific domains covered. Furthermore, most existing ontologies do not specifically focus the problems of business process modelling. Only some of them provide means (for example in the form of $\mathcal{OBDR}, \mathcal{PEDR}, \mathcal{PRDR}$) to derive a hierarchy of processes and activities in order to determine their level of abstraction. Consequently, in most cases a new ontology has to be built in order to guide modelling. This, of course, implies significant efforts.

The second perspective is adopted by the *modelling language-based approach* (Becker et al. 2007, Becker & Pfeiffer 2007, Pfeiffer 2007). It focuses on specifically designed modelling languages that support distributed modelling. The modelling language-based approach addresses the problem of subjectivity by offering language constructs that limit the choices of the modeller. For this purpose the set of constructs is carefully selected and restrictive meta-models and grammars are defined. This can mainly be done with the help of the well-formedness rules and a comprehensive and unambiguous definition for each construct. The work on *modelling conventions* (Rosemann 2003) is closely related to the modelling language-based approach.

The following four conflicts are mainly addressed by the *language-based approach*:

- *Type conflicts*: There are at least two possibilities to reduce the number of type conflicts with the aid of modelling languages. Firstly, based on the meta-model or the grammar of the

modelling language, mapping rules can be defined that reveal semantically corresponding structures of constructs (Höfferer 2007). In the case of BPG_{BAD} it could for example be defined that *automated* and *manual activities* share the same meaning whenever a paper file is attached to both of them. Secondly, semantically overlapping language constructs can be avoided in the first place, i.e. during the construction of a language (Wand 1996). In the example of BPG_{BAD} this means that there can be only one modelling construct *activity* and not two of them with a similar meaning. Whether, this activity is performed automatically or manually can be described by an additional construct such as annotation object. Semantically disjoint modelling language constructs assure that when a certain fact of the domain has to be represented, there is only one modelling language construct that fits its meaning (Pfeiffer 2007). Formally expressed this situation means: $e =_{tsyn} e' \Leftrightarrow e =_{tsem} e'$. Consequently, modelling rules as well as semantically disjoint constructs address part [b-c] in condition (C5).

- *Control flow and annotation conflicts*: In order to handle these kind of conflicts, the meta-model or grammar must restrict the number of *control* and *annotation object flows*. Consequently, obligatory and optional constructs as well as their allowed number of instances have to be stated explicitly. For example in the language BPM_{BAD} it could be defined that every activity has to have exactly one annotation object attached. This well-formedness rule $v \in V$ can be specified in the following way: $\forall e \in E: \left|\bullet^{\mathcal{OF}} e\right| = 1$. A similar condition can also be formulated for all *control flows*. The result would be a modelling language that only supports sequential modelling. The feasibility of such a modelling language has been discussed in Becker et al. (2007). Consequently, based on the well-formedness rules V the number of incoming and outgoing edges can be fixed and, so, harmonised for different modellers. Hence, parts [d] and [e] in condition (C5) are addressed.
- *Order conflicts:* The sequence of its elements has an important impact on the semantics of a process model. However, there are also cases where an arbitrary order of certain elements does not influence the meaning of a process. This problem has been described in Figure 7 based on the activities "print application" and "approve application". In IS literature a modelling language construct has been proposed to address this issue. In the EPC it is called the *SEQ-operator*. As part of the workflow patterns (van der Aalst et al. 2003) *interleaved parallel routing* has been suggested as a solution. This construct makes sure that a set of activities is executed in a random order, but not in parallel. Consequently, areas in the process model can be defined where the order of the elements does not matter. Thus, in (C5) condition part [f], [g], and [h] are partially addressed by this construct.

The main problem of the modelling language-based approach is that it supposes the modelling language to be freely modifiable. However, many companies and public administrations have defined an organisation wide standard for process modelling. The modelling tools that are available and licensed are tailored to this specific language. Therefore, modifications on the a process modelling language can lead to high organisational efforts when standards have to be changed. Additionally, investments in new modelling tools might be necessary. From an overall perspective the modelling language based approach can entail an uncontrolled growth of different variants of a process modelling grammar. The result can be process models that are described in many different languages. Such a situation significantly increases the efforts of an integration project as all models have to be transformed in a common language first.

This literature analysis has revealed that the ontology-based and the modelling language-based approach are complementary to each other. Both of them refer to different kind of conflicts. The ontology-based approach is mainly concerned with *homonym, synonym, abstraction,* and *separation conflicts*. The modelling language-based approach provides measures to handle *type*, c*ontrol flow, annotation,* and *order conflicts*. However, together both of them are able to address all of the distributed modelling conflicts. Therefore, an integrated approach that deals with the conflicts has to apply a specifically designed modelling language in combination with a domain ontology.

5 SUMMARY AND OUTLOOK

This paper has been based on the assumption that modelling projects of a practically relevant size are distributed. This means that multiple modellers are involved to collect the process information. Current modelling languages and methods, however, seem to implicitly suppose a single modeller who captures the entire organisation. To address this issue we have suggested a formal framework as a starting point. With the help of this framework and a thought experiment we were able to derive the main conflicts of distributed process modelling (cf. R1). Based on the conflicts we could define a structural equivalence criterion for BPMs. This finding is a strong theoretical argument that the conflicts are correct and complete. However, further empirical research is necessary to evaluate whether these eight conflicts are sufficient to explain all deviations that can actually be found in the business process modelling practice (Soffer & Hadar 2007).

Based on these theoretical results we have conducted a literature review that has revealed the *language-based* and the *ontology-based approach* as two possible ways to handle the conflicts. The analysis of the literature has also shown that a combination of both perspectives can lead to an integrated approach that considers all conflicts (cf. R2). Empirical results that such an attempt is in general feasible do exist in form of the business process modelling language PICTURE. This language has been specifically designed for collaborative modelling and has successfully been applied in multiple projects with more than ten modellers each (Becker & Pfeiffer & Räckers 2007). Thus, PICTURE offers first empirical evidence that an integrated approach can reduce the distributed modelling conflicts. However, further empirical studies are necessary to fortify this preliminary result.

It is also open to further research to elaborate in more detail on the integrated approach. Especially interesting is the question whether a combination of the *language-based* and the *ontology-based approach* can yield additional mechanisms to address the conflicts. *The ontology-based approach* suggests to use a domain ontology to annotate the process models. It has been proposed in literature (Evermann 2005, Pfeiffer 2007) to obtain the constructs of the modelling language mainly from the ontology as well. The resulting domain specific process modelling language might be able to simplify the determination of the relevant number of flows that can be attached to a model element. Thus, the rules that avoid control flow and annotation conflicts could easier be defined. The proposal could also enable the definition of semantic rules that indicate whether certain elements should be part of the model or are in the right order. Therefore, also separation conflicts and order conflicts could further be reduced. This example indicates that there is potential for further research that arises from a combination of the approaches.

Acknowledgements

The work published in this paper is partly funded by the European Commission through the STREP PICTURE. It does not represent the view of European Commission or the PICTURE consortium and the authors are solely responsible for the paper's content.

References

Balzer, W., Moulines, C. U. and Sneed, J. D. (1987). An Architectonic for Science - The Structuralist Program. D. Reidel Publishing Company. Dordrecht.

Becker, J., Algermissen, L., Falk, T., Pfeiffer, D. and Fuchs, P. (2006). Model Based Identification and Measurement of Reorganization Potential in Public Administrations – the PICTURE-Approach. In Proceedings of the 10th Pacific Asia Conference on Information Systems (PACIS 2006), p. 860-875, Kuala Lumpur, Malaysia.

Becker, J., Algermissen, L., Pfeiffer, D. and Räckers, M. (2007). Local, Participative Process Modelling - The PICTURE-Approach. In Proceedings of the 1st International Workshop on Management of Business Processes in Government, Brisbane, Australia.

Becker, J., Delfmann, P. and Knackstedt, R. (2007). Adaptive reference modeling: integrating configurative and generic adaptation techniques for information models. In Reference modeling: efficient information systems design through reuse of information models (J. Becker and P. Delfmann Ed.), p. 23-49, Physica, Heidelberg.

Becker, J. and Pfeiffer, D. (2007). Automated Semantic Analyses of Conceptual Models. In Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007) Forum, p. 65-68, Trondheim, Norway.

Becker, J., Pfeiffer, D. and Räckers, M. (2007). Domain Specific Process Modelling in Public Administrations - The PICTURE-Approach. In Proceedings of the 6th International EGOV Conference, LNCS 4656, p. 68-79, Regensburg.

Chandrasekaran, B., Joesephson, J. and Benjamins, R. (1999). What are Ontologies and Why Do We Need Them? IEEE Intelligent Systems, 14 (1), p. 20-26.

Dalal, N. P., Kamath, M., Kolarik, W. J. and Sivaraman, E. (2004). Toward an integrated framework for modeling enterprise processes. Communications of the ACM, 47 (3), p. 83-87.

Davenport, T. H. and Beers, M. (1995). Managing information about processes. Journal of Management Information Systems, 12 (1), p. 57-80.

Davis, I., Green, P., Milton, S. and Rosemann, M. (2003). Using meta models for the comparison of ontologies. In Proceedings of the 8th International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD 2003) at 15th International Conference on Advanced Information Systems Engineering (CAiSE '03), p. 1-10, Velden, Austria.

Evermann, J. (2005). Towards cognitive foundation for knowledge representation. Information Systems Journal, 15 (2), p. 147-178.

Fox, M. S. and Gruninger, M. (1998). Enterprise Modeling. AI Magazine, 19 (3), p. 109-121.

Grudin, J. (1994). Computer-supported cooperative work: Its history and participation. IEEE Computer, 27 (5), p. 19-26.

Guizzardi, G., Pires, L. F. and Sinderen, M. J. v. (2002). On the role of Domain Ontologies in the design of Domain-Specific Visual Modeling Languages. In Proceedings of the 2nd Workshop on Domain-Specific Visual Languages, 17th ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002), Seattle,WA.

Hakimpour, F. and Geppert, A. (2001). Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach. In Proceedings of the International conference on Formal Ontologies in Information Systems (FOIS 2001) (C. Welty and B. Smith Ed.), p. 297-308, Ogunquit, Maine.

He, Q. and Ling, T. W. (2006). An ontology based approach to the integration of entity–relationship schemas. Data & Knowledge Engineering, 58 (3), p. 299–326.

Höfferer, P. (2007). chieving Business Process Model Interoperability Using Metamodels and Ontologies. In Proceedings of the 15th European Conference on Information Systems (ECIS 2007), p. 1620-1631, St. Gallen, Switzerland.

Hoppenbrouwers, S. J. B. A., Proper, E. and van der Weide, T. P. (2005). Towards explicit strategies for modeling. In Proceedings of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD 2005) (T. A. Halpin, K. Siau and J. Krogstie Ed.), p. 485-492, Porto, Portugal.

Kashyap, V. and Sheth, A. (1996). Semantic and schematic similarities between database objects: a context-based approach. The International Journal on Very Large Data Bases (VLDB), 5 (4), p. 276-304.

Lawrence, R. and Barker, K. (2001). Integrating relational database schemas using a standardized dictionary. In Proceedings of the 16th ACM Symposium on Applied Computing, Las Vegas, USA.

Lindsay, A., Downs, D. and Lunn, K. (2003). Business processes — attempts to find a definition. Information and Software Technology, 45 (15), p. 1015-1019.

McGuinness, D. L. (2003). Ontologies Come of Age. In Spinning the Semantic Web - Bringing the World Wide Web to its Full Potential (D. Fensel, J. Hendler, H. Lieberman and W. Wahlster Ed.), MIT Press, Cambridge, MA.

Mylopoulos, J. (1998). Information modeling in the time of the revolution. Information Systems, 23 (3-4), p. 127-155.

Object Management Group (2006). BPMN Final Adopted Specification 1.0. Downloaded from http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf on 2006-Apr-30.

Patig, S. (2004). Measuring Expressiveness in Conceptual Modeling. In Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004), Riga, Latvia.

Pfeiffer, D. (2007). Constructing Comparable Conceptual Models With Domain Specific Languages. In Proceedings of the 15th European Conference on Information Systems (ECIS 2007), St. Gallen.

Pfeiffer, D. and Gehlert, A. (2005). A framework for comparing conceptual models. In Proceedings of the Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2005), p. 108-122, Klagenfurt, Austria.
Priemer, J. (1995). Entscheidungen über die Einsetzbarkeit von Software anhand formaler Modelle. Pro Universitate. Sinzheim.
Quartel, D. and van Sinderen, M. (2007). On interoperability and conformance assessment in service composition. In Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), p. 229-240, Annapolis, MD.
Rosemann, M. (2003). Preparation of Process Modeling. In Process Management (J. Becker, M. Kugeler and M. Rosemann Ed.), p. 41-78, Berlin et al.
Rosemann, M. and zur Mühlen, M. (1998). Evaluation of workflow management systems: a meta model approach. The Australian Journal of Information Systems, 6 (1), p. 103-116.
Scheer, A.-W. (2000). ARIS - Business Process Modeling. 3 Edition. Springer Publishing. Heidelberg et al.
Simon, C. and Mendling, J. (2007). Integration of Conceptual Process Models by the Example of Event-driven Process Chains. In Proceedings of the Wirtschaftsinformatik (WI 2007), p. 677-694, Karlsruhe, Germany.
Soffer, P., Golany, B. and Dori, D. (2003). ERP modeling: a comprehensive approach. Information Systems, 28 (6), p. 673-690.
Soffer, P. and Hadar, I. (2007). Applying ontology-based rules to conceptual modeling: a reflection on modeling decision making. European Journal of Information Systems, 16 (5), p. 599-611.
Speck, M. and Schnetgöke, N. (2003). To-be Modelling and Process Optimization. In Process Management (J. Becker, M. Kugeler and M. Rosemann Ed.), p. 135-164, Springer, Berlin, Heidelberg, New York.
Storey, V. C. (2005). Comparing Relationships in Conceptual Modeling: Mapping to Semantic Classifications. IEEE Transactions on Knowledge and Data Engineering, 17 (11), p. 1478-1489.
Thomas, O. and Scheer, A.-W. (2006). Tool Support for the Collaborative Design of Reference Models — A Business Engineering Perspective. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS 2006), Kauai, HI.
Turetken, O. and Demirors, O. (2007). An Approach for Decentralized Process Modeling. In Proceedings of the International Conference on Software Process (ICSP 2007), LNCS 4470, p. 195-207, Minneapolis, MN.
Uschold, M. (1998). Knowledge level modelling: concepts and terminology. The Knowledge Engineering Review, 13 (1), p. 5-29.
Uschold, M. and Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. ACM SIGMOD Record, 33 (4), p. 58-64.
van der Aalst, W., ter Hofstede, A. H. M., Kiepuszewski, B. and Barros, A. P. (2003). Workflow Patterns. Distributed and Parallel Databases, 14 (1), p. 5-51.
Wand, Y. (1996). Ontology as a foundation for meta-modelling and method engineering. Information and Software Technology, 38 (1996), p. 281-287.
Weber, I., Hoffmann, J., Mendling, J. and Nitzsche, J. (2007). Towards a Methodology for Semantic Business Process Modeling and Configuration. In Proceedings of the 2nd International SeMSoC Workshop "Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing", Vienna, Austria.
Weinberger, H., Teeni, D. and Frank, A. J. (2003). Ontologies of Organizational Memory as a Basis for Evaluation. In Proceedings of the 11th European Conference on Information Systems (ECIS 2003), Naples, Italy.
Werth, D., Walter, P. and Loos, P. (2007). Conceiving an environment for managing the lifecycle of collaborative business processes. In Proceedings of the Wirtschaftsinformatik, p. 805-822, Karlsruhe, Germany.
Wimmer, K. and Wimmer, N. (1992). Conceptual modeling based on ontological principles. Knowledge Acquisition, 4, p. 387-406.