

An Approach for Maintaining Models of an E-Commerce Collaboration

Lianne Bodenstaff^{1*}, Andreas Wombacher², Roel Wieringa¹
IS Group¹, DB group²
University of Twente, The Netherlands
{l.bodenstaff,a.wombacher,r.j.wieringa}@utwente.nl

Manfred Reichert
DBIS Institute
University of Ulm, Germany
manfred.reichert@uni-ulm.de

Abstract

To keep an overview on complex E-Commerce collaborations several models are used to describe them. When models overlap in describing a collaboration, the overlapping information should not contradict. Models are of different nature and maintained by different people. Therefore, keeping model-overlap contradiction-free is challenging. In this paper we propose a novel approach for maintaining models representing an E-Commerce collaboration. Applying this approach supports avoiding contradictions in models during evolution of E-Commerce collaborations.

1 Introduction

Nowadays, model-based implementation approaches are widely used. Many of these approaches allow specifying several models, each emphasizing one specific aspect of the system. Due to the complex nature of E-Commerce collaborations often a variety of models is used to specify the system. For example, financial benefits are captured in a business model while coordination details are captured in a process model. Although using several models to represent one complex system enhances usability when developing a specific model, new challenges arise when combining models for implementation. Different models representing one collaboration should be consistent with each other (inter-model consistency) and have to be maintained during runtime since they should reflect the behavior of the system.

To illustrate the importance of inter-model consistency, we consider two fundamental perspectives which are of high relevance for modelling E-Commerce collaborations: the *business* and the *process perspective* [5][13]. At a business level expectations, e.g., agreements on the number of transferred products between partners, are modelled. At a process level coordination of interorganizational processes

is modelled. Both perspectives describe necessary transfers between partners although focusing on different aspects (financial versus coordination). When, e.g., a payment is depicted in the business model while this is omitted in the process model then the models are *inconsistent*. If an implementation is based on these models payment is expected to occur (due to the business model) while occurrence of this payment is not enforced by the implementation (due to the coordination model). In other words, business and process model do not describe the same system (i.e., they are inconsistent). Since the models have a different level of abstraction, use different modelling notations, and have a different purpose, determining consistency is a challenge.

Our goal is to provide a *model independent approach* for ensuring inter-model consistency at design time as well as for maintaining inter-model consistency at the operational level. In this paper we present an approach which supports the user in efficiently maintaining consistency of his models (cf. Fig. 1). During runtime we use data from the *event log* to analyze behavior of the system. The contribution of this paper is as follows. We identify overlap between models of the collaboration to overcome model differences (cf. Sect. 4), we define consistency constraints between each pair of models based on the overlap (cf. Sect. 5), and after detecting an inconsistency, models might have to be adapted to regain consistency. To support decision making on the necessary type of change, we categorize for each model the possible changes (cf. Sect. 6). Based on this categorization it is possible to determine an efficient model change to solve specific violations of consistency constraints (cf. Sect. 7). Sect. 2 introduces our running example and the models we use for illustration purpose. Sect. 3 describes our overall approach after which we elaborate on each step in the following sections. Sect. 8 discusses related work and we conclude with summary and outlook.

2 Basics

For representation purposes we simplify the business case and omit repetitive behavior in this paper. How to deal

*This research has been supported by the Dutch Organization for Scientific Research (NWO) under contract number 612.063.409

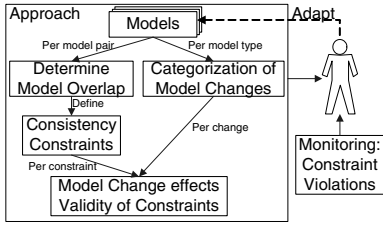


Figure 1. Maintaining Consistency

with this and other issues is described in [3]. Our business case consists of a *copier company* which *sells* and *leases* copiers to customer companies. When leasing a copier, it is mandatory to purchase *maintenance* on a yearly basis. Before implementation the company wants to evaluate financial consequences (business perspective) as well as coordination requirements (process perspective). For this purpose, it develops a *value model* denoting value exchanges and a *coordination model* describing how interaction between partners is arranged. To validate its models, information on the interactions with partners is gathered from the *event log*.

2.1 Business Perspective

The company reasons about *value transfers* to and from companies to *estimate* financial benefits. The *value model* (cf. Fig. 2) depicts estimations on who gets what from whom and for how much in e^3 -value notation [6].¹ In our running example, actor *copier company* has a group of *customer* companies, and three types of *value exchanges* are exchanged between them (all depicted in Fig. 2): money for leasing a copier (*Lease*, *Copier L*), money for maintenance of the copier (*Service*, *Maintenance*), and money for purchasing a copier (*Purchase*, *Copier P*). Interdependent value transfers (i.e., transfers exchanged in one business transaction) are connected through dotted and solid lines in Fig. 2, representing two possible *business transactions*. One is highlighted through a thick line representing the *customer need* for having a copier, starting at the customer side. The *XOR-split* indicates customers either lease or purchase. The *AND-split* indicates that for every lease a maintenance contract is purchased.

To obtain financial estimations, several quantifications are done for a specified *time frame*. In Fig. 2 estimations on the number of customers (=36), the number of customer needs (=2), and the purchase-lease ratio (33%-67%) are made. These quantifications result in an estimation on the *number* of leases and purchases. Together with an *average value* of each transfer, this gives an indication on the income for this

¹Other value-based modelling techniques (e.g. REA [10] and Business Modelling Ontology [12]) can be used as well. We select e^3 -value in this paper due to its graphical notation.

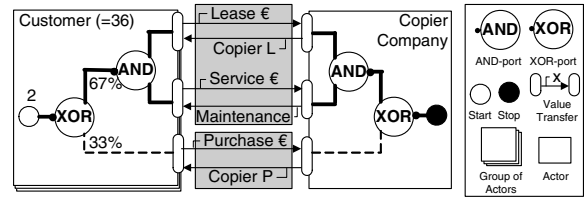


Figure 2. Value Model in e^3 -value notation

Value Transfer	Average Value	Occurrences
Lease /Copier L	1200 €	48
Service /Maintenance	700 €	48
Purchase /Copier P	7500 €	24

Table 1. Estimations Value Model Fig. 2

business activity in the specified time frame (one year in our case). Although these quantifications are part of the model, we represent them for clarification in Tab. 1.

2.2 Process Perspective

Besides financial validation the company needs to agree on *how* to implement the business. For example, should the customer pay before receiving the copier, or the other way around? The coordination model describes which messages are to be exchanged between partners and in which order. Such an *inter-organizational* process model is basis for implementation. We use Business Process Modeling Notation (BPMN) [14] to represent the coordination model (cf. Fig. 3).² The customer *chooses* to purchase or lease. Based on this *message* the copier company makes an offer after which the customer pays and receives the copier.

2.3 Event Log

To evaluate the operational system, data on its execution is gathered. The event log of the IS contains such data. Furthermore, *timestamps* show the order in which data are exchanged. As example take Fig. 4 in which parts of an XML based event log (i.e., one business transaction) are shown. It depicts data being exchanged between customer and copier company of payment for leasing a copier. Each message is annotated with a *timestamp*, *issuer*, *recipient* and *name*. A message contains information on the value of a transfer (*Amount*), the type of a transfer (*Good*), and a *contract number*. Messages with same contract number belong to one business transaction while one specific customer can have multiple business transactions.

²Other modelling techniques like Activity Diagrams and Petri Nets are applicable as well. Here we select BPMN due to its graphical notation.

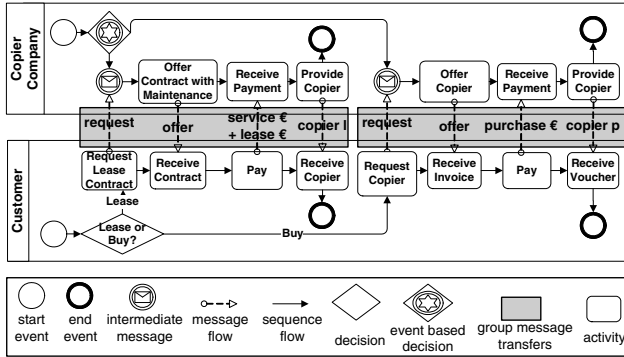


Figure 3. Coordination Model, BPMN notation

State	Time	Sender	Receiver	Message
Done	2007_08_17 09:13:33	customer_a	copier_company	request
Done	2007_08_17 09:15:30	copier_company	customer_a	offer
Done	2007_08_17 09:23:12	customer_a	copier_company	copier payment
Done	2007_08_17 09:25:14	copier_company	customer_a	copier l

Date: Fri, 17 Aug 2007 09:23:12
 Sender: customer_a
 <?xml version='1.0' encoding='UTF-8?'>
 <soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
 <soapenv:Header />
 <soapenv:Body>
 <copier_payment xmlns='http://www.utwente.nl/consistency'>
 <Process_payment>
 <Good>Service</Good>
 <Amount>700</Amount>
 </Process_payment>
 <Process_payment>
 <Good>Lease</Good>
 <Amount>1200</Amount>
 </Process_payment>
 <contract_number>NL_TWENTE_98267854</contract_number>
 </copier_payment>
 </soapenv:Body>
 </soapenv:Envelope>

Figure 4. Event Log in XML

3 Approach

When several models, each describing part of the business, form the basis for implementation, it is necessary to ensure consistency between them. Furthermore, after implementation it is important to maintain consistency between models and running system. When an inconsistency is detected, restoring it can be done by adapting models or implementation. Our approach enables comparison of different models and maintaining consistency between models and running system. It is suitable for E-Commerce models which focus on interorganizational collaborations. Therefore, the minimum requirement is that exchanges between partners are described since our approach relies on message transfers. Consistency checks are done for a single actor, i.e., from a local viewpoint. Comparison of models is challenging because of: *different levels of granularity*, *different purpose of the models*, and *different modelling notation*.

Different models of a system can have a different level of *granularity*. For example, a business model might state

that company A pays x Euro to company B, while a coordination model might specify a spread payment of five transfers. This problem can be overcome by making *hierarchical* models where the model with the highest level of granularity determines to which level other models are aggregated. As a result sets of entities (in our example, sets of messages) can be linked to one higher level entity (message) of the other model. For the remainder of this paper we assume that this granularity difference is overcome by aggregation. To compare models with a different *purpose* their commonalities have to be identified. Part of our approach (cf. Fig. 1) is to handle different modelling *notations* by representing their commonalities *independent* from any formalism (cf. Sect. 4). Another part is to find consistency constraints between all pairs of models (cf. Sect. 5). If an inconsistency occurs, one or more models can be changed to restore it. Therefore, we determine which *types* of changes can be distinguished in each model (cf. Sect. 6). For each type of change, we determine which consistency constraints it influences (cf. Sect. 7).

4 Overcoming Model Differences

Different viewpoints (e.g. value and coordination viewpoint), modelled using different notations (e.g. e³-value and BPMN), can not be checked in a straightforward manner for inter-model consistency. Information present in both models (i.e., overlap) has to be identified since inconsistencies can occur there. Furthermore, during runtime data in the event log are compared with information in the models. To investigate this, we propose to *abstract* from the modelling techniques and represent overlapping parts of the models and information checked during runtime independent from a particular formalism or notion. We refer to this as *abstraction* of models and use *sets and tuples* for representation.

For identifying overlap *commonalities* between models have to be found. Therefore, we check which *entities* and *relations* they have in common. We focus in our approach on *transfers* exchanged between different partners (i.e., entities) and *dependencies* between these transfers (i.e., relations). Transfers are represented as elements of sets and tuples, and relations between transfers are depicted by grouping interdependent transfers in one set. Dependency between transfers in case of the value model means that the transfers are realized in one business transaction.

Two possible business transactions are depicted as highlighted grey areas in Fig. 2). Important in overlap with the coordination model are value transfers resulting in message transfers in the coordination model. Furthermore, during runtime the value model is validated by checking the number of value transfers and their value. Note that value transfers which do neither appear in the coordination model nor in the event log are also not captured in the abstract-

tion. Therefore, only value transfers representing *products* or *money* are captured. A value transfer in a value model has an *issuer*, *recipient*, unique *name*, estimated average *value*, and estimation on the number of *occurrences*, which we represent as quintuple $x=(a,b,c,d,e)$ where $issuer(x)=a$, $recipient(x)=b$, $name(x)=c$, $value(x)=d$ and $occurrences(x)=e$. For example, in Tab. 1 value transfer *Copier P* is expected to be issued by the copier company (represented as *cc*), to be received by the customer (represented as *c*), to have an average value of 7500 Euro, and to occur 24 times. This is represented as: *Copier P*=(*cc*,*c*,*copierp*,7500,24). The abstraction of the value model from Fig. 2 contains two sets (the two grey areas) as well as the accompanying values and number of occurrences in Tab. 1.

$$\mathcal{V} = \left\{ \begin{aligned} & \{(c, cc, lease, 1200, 48), (cc, c, copier1, 1200, 48), \\ & (c, cc, service, 700, 48)\}, \\ & \{(c, cc, purchase, 7500, 24), (cc, c, copierp, 7500, 24)\} \end{aligned} \right\}$$

Essential in the coordination model in turn, are the message transfers, the order in which they appear, and the different actors involved. We use sets of message transfers performed in a single business transaction to represent the coordination model (see highlighted grey areas in Fig. 3). A message transfer is represented by *issuer*, *recipient*, and unique *name* as a triplet $x=(issuer, recipient, name)$. For example, message transfer *Copier P* in Fig. 3 is represented as (*cc*,*c*,*copierp*) with $issuer(Copier P)=cc$, $recipient(Copier P)=c$, and $name(Copier P)=copierp$. Furthermore, a *strict partial order* $<$ is defined between the messages based on the order in which they appear in the model. The coordination model in Fig. 3 can be represented as a set containing two sets.

$$\mathcal{W} = \left\{ \begin{aligned} & \{(c, cc, request), (cc, c, offer), (c, cc, lease), (cc, c, copier1), \\ & (c, cc, service)\}, \\ & (c, cc, request) < (cc, c, offer), (cc, c, offer) < (c, cc, lease), \\ & (c, cc, lease) < (cc, c, copier1), (c, cc, service) < (cc, c, copier1), \\ & \{(c, cc, purchase), (cc, c, copierp)\}, (c, cc, request) < (cc, c, offer), \\ & (cc, c, offer) < (c, cc, purchase), (c, cc, purchase) < (cc, c, copierp) \} \end{aligned} \right\}$$

The abstraction of the event log contains sets of entries which are performed in a single business transaction. Each entry in an event log is modelled with a *timestamp*, *issuer*, *recipient*, unique *name*, and specific *value*. For example entry *Service* in transfer *copier_payment* (cf. Fig. 4) is represented as: (3,*c*,*cc*,*service*,700). We use a simplified notation for timestamps, a higher integer means a later point in time. The following example shows the set abstraction of an event log for one month.

$$\mathcal{E} = \left\{ \begin{aligned} & \{(1, c, cc, request, 0), (2, cc, c, offer, 0), (3, c, cc, lease, 1200), \\ & (4, cc, c, copier1, 1200), (3, c, cc, service, 700)\}, \\ & \{(5, c, cc, request, 0), (6, cc, c, offer, 0), (7, c, cc, lease, 1400), \\ & (8, cc, c, copier1, 1400), (7, c, cc, service, 700)\}, \\ & \{(9, c, cc, request, 0), (10, cc, c, offer, 0), (11, c, cc, purchase, 6000), \\ & (14, cc, c, copierp, 6000)\}, \\ & \{(15, c, cc, request, 0), (16, cc, c, offer, 0), (17, c, cc, lease, 2500), \\ & (20, cc, c, copier1, 2500), (17, c, cc, service, 1000)\}, \\ & \{(21, c, cc, request, 0), (22, cc, c, offer, 0), (23, c, cc, purchase, 10000), \\ & (24, cc, c, copierp, 10000)\}, \\ & \{(25, c, cc, request, 0), (26, cc, c, offer, 0), (27, c, cc, lease, 1000), \\ & (28, cc, c, copier1, 1000), (27, c, cc, service, 700)\} \end{aligned} \right\}$$

5 Inter-Model Consistency Constraints

Models used for implementing an E-Commerce collaboration should not contradict (i.e., be consistent). Therefore, consistency constraints between each pair of models have to be defined. Since these models differ in notation, granularity, and purpose, we propose to use the abstraction of each model (cf. Sect. 4) to explicate these constraints. We consider two models to be consistent if they facilitate the same business transactions (e.g. one option for purchase and one for leasing a product). Furthermore, each business transaction has to be facilitated in the same way (e.g. purchase over the internet versus purchase in a store). Therefore, not only consistency constraints on the sets representing the business transaction are defined but also on the transfers in these sets. Since the abstractions capture the overlap between the models by representing each transfer as a tuple, we can define these consistency constraints by *matching* sets and matching elements (tuples) in the sets. In general we define consistency between models during design time as follows.

Consistency 1 (Design Time) *Each set in the abstraction of model A has exactly one matching set in the abstraction of model B, and vice versa. These sets match if for each transfer in model A there exists a transfer in model B, and vice versa. Only transfers which are part of the overlap between model A and model B are considered.*

In complex E-Commerce collaborations where repetitions in a business transaction occur, also *relations between* sets and between elements in sets have to be checked. In this case it should be checked whether certain transfers are executed an equal number of times (e.g. are as many payments done by the customer as services offered by the company?). Due to space limitations we can not provide a formal definition on how to handle this but this can be found in a technical report extending this paper [3].

After implementation (i.e., during runtime) we check consistency between models and event log. Data from the running system abstracted from the event log have to be

compared with the models. It is checked whether the realized business transactions present in the event log are indeed instantiations of business transactions in the models and whether each modelled business transaction occurs at least once on the event log. In general we define consistency between models and event log as follows.

Consistency 2 (Runtime) *Each set (representing a business transaction) in the event log for the specified time frame should match a set in the abstraction of model A. Each business transaction in model A has to occur as a business transaction in the event log for that time frame.*

Furthermore, additional *model specific properties* are checked, e.g. whether messages are transferred in the order specified in the coordination model. These constraints are different for each type of model. We illustrate how to define these constraints for our running example. A complete formalization of these constraints, including definitions, can be found in the extended version of this paper [3].

5.1 Design Time

During design time value and coordination model have to be consistent (i.e., both facilitate purchasing and leasing a copier). To check this, we *match* the set representing purchasing a copier in the value model with the set representing this in the coordination model. Furthermore, each product or money transfer in the value model must have a matching message transfer in the coordination model. Each valuable message (we refer to a message as valuable when the content concerns money or products) in the coordination model must have a matching value transfer in the value model. In the abstraction, elements match if they have the same *issuer*, *recipient*, and *name*.

Constraint 3 (Value and Coordination Model) *Each set in the abstraction of a value model has exactly one matching set in the abstraction of a coordination model, and vice versa. Sets match if for each product/money transfer in the value model there exists a message transfer in the coordination model and for each valuable message in the coordination model there exists a value transfer in the value model.*

Both sets in value (\mathcal{V}), and coordination (\mathcal{W}) model of our running example match. As well as the elements in these sets. For example, value transfer $(c, cc, lease, 1200, 48)$ matches message transfer $(c, cc, lease)$ since both have the same issuer, recipient and name. Therefore, the value model in Fig. 2 is consistent with the coordination model in Fig. 3 (i.e., Constraint 3 is met).

5.2 Runtime

Value Model and Event Log. Each business transaction in the event log should match one of the business transactions in the value model (e.g., each business transaction in the event log has to be a purchase or lease transaction). Furthermore, each business transaction in the value model has to occur at least once in the event log. When, for example, no purchases are done but only lease transactions, this will not be consistent with the value model. More specifically, each product and money transfer in the value model has to be represented as an entry in the event log and each value entry in the event log (i.e., money or product transfer) has to be represented as a value transfer in the value model.

Constraint 4 (Business Transaction) *Each set (representing one business transaction) in the event log for the specified time frame should match a set in the abstraction of the value model. Furthermore, each business transaction in the value model has to occur as a business transaction in the event log for the specified time frame.*

Since the value model denotes financial benefits of the collaboration over a specified period of time we also check whether estimated profits are achieved. This is done by checking whether the *number* of transfers and their *value* is equal to the estimations.

Constraint 5 (Number of Occurrences) *For each business transaction in the value model the estimated number of occurrences has to be equal to the realized number of business transactions in the event log during the specified time frame.*

Constraint 6 (Average Value) *The estimated average value of the transfers in each business transaction of the value model have to be equal to the realized average value of this transfer in the event log for the specified time frame.*

Each business transaction in event log \mathcal{E} is a realization of a business transaction in value model \mathcal{V} (cf. Constraint 4). Each valuable entry in the event log matches a value transfer in the value model. Furthermore, both business transactions modelled in the value model are present in the event log. Furthermore, the number of occurrences should be equal to estimations in the value model (cf. Constraint 5). Estimations in the value model were for one year while event log \mathcal{E} represents activities over one month. Therefore, we divide the estimated number of occurrences by twelve. For example, the realized number of *Purchase* transfers in the event log is 2 and the estimated number of occurrences *Purchase* in the value model is $24 \frac{1}{12} = 2$. Furthermore, the realized average value should be equal to the estimated average value (cf. Constraint 6). For example, the realized

average value of *Purchase* is $\frac{6000}{2} + \frac{10000}{2} = 8000$ Euro and therefore *not* equal to estimated average value in the value model of 7500 Euro. Therefore, value model and event log are *not* consistent at September 15, 2007.

Coordination Model and Event Log. Each business transaction in the event log should match a business transaction in the coordination model. Furthermore, each business transaction in the coordination model has to occur at least once in the event log. Each message in the coordination model has to occur as an entry in the event log and each entry in the event log has to occur as a message in the coordination model.

Constraint 7 (Business Transaction) *Each set (representing one business transaction) in the event log for the specified time frame should match a set in the abstraction of the coordination model. Furthermore, each business transaction in the coordination model has to occur as a business transaction in the event log for the specified time frame.*

Since the coordination model explicates in which order messages are to be exchanged, it is checked whether this prescribed order matches the order of entries in the event log. The strict partial order in the abstraction of the coordination model can be checked in a straightforward manner with the timestamps occurring in event log entries.

Constraint 8 (Ordering) *Messages in each business transaction of the event log for the specified time frame must be ordered as prescribed in the coordination model.*

To check consistency between event log \mathcal{E} and coordination model \mathcal{M} , we check both constraints. Each business transaction in the event log matches a set in the abstraction of the coordination model (cf. Constraint 7). Each entry in the event log matches a message transfer in the coordination model. For example, event log entry $\{(1, c, cc, request, 0), (2, cc, c, offer, 0), (3, c, cc, lease, 1200), (4, cc, c, copier1, 1200), (3, c, cc, service, 700)\}$ matches $\{(c, cc, request), (cc, c, offer), (c, cc, lease), (cc, c, copier1), (c, cc, service)\}$. Furthermore, the *order* of messages in the coordination model should be equal to the order in the event log (cf. Constraint 8). The coordination model prescribes that payments have to occur before delivery, which they are in the event log. In our example, event log and coordination model are consistent since both constraints are met.

6 Model Changes

To maintain consistency between running system and models, changes might be necessary. For example, when the event log shows a lower number of sales than expected it makes sense to adapt estimations in the value model, or

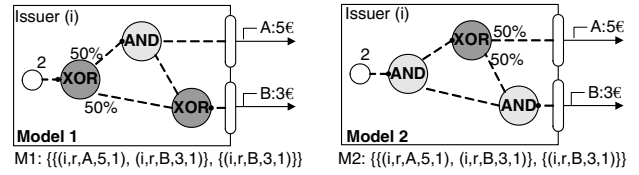


Figure 5. Same Business Transaction

to change implementation to enforce more sales. Handling *impact* of changes within one model on inter-model relations for complex collaborations is tedious work. Each consistency relation a changed model has with other models, must be reevaluated and, if necessary, updated. To allow more efficient and structured checking and maintaining of consistency, we propose to determine *upfront* effects certain changes have. In this way, a well-informed decision on the type of change can be made. Furthermore, it is possible to oversee which other relations are affected. We demonstrate our approach by illustrating how to categorize changes in value and coordination models (cf. Tab. 2).

Change 1. *Non-observable changes* in a model do not influence the abstraction of the model, i.e., the change does not influence the possible business transactions. (i) Typically, there is more than one way to structure a model while preserving the same possible business transactions. For example, though *Model 1* and *Model 2* in Fig. 5 are different, they facilitate the same transactions since abstraction *M1* and *M2* are equal. Since a change from *Model 1* to *Model 2* does not change the possible business transactions, we refer to this as a non-observable change. (ii) The other way is to change part of the model with no connection to the abstraction. For example, adapting transfer *Maintenance* in Fig. 2 does not influence the abstraction of the model since it is not a product or money exchange (it is a service).

As opposed to non-observable changes, *observable changes* do change the possible business transactions. Depending on the type of model, several categories of change can be identified (cf. Tab. 2). The abstraction of a model consists of *sets*, each set representing one possible business transaction. Each business transaction consists of one or more exchanges.

Change 2. *Observable structural changes* add or remove (part of) a business transaction by adding or removing *constructs* (e.g. XOR-splits and transfers) while preserving a valid model. Fig. 6 depicts three observable structural changes represented as an e^3 -value model. Model *M* with set $\{A, B, C\}$ is the original model of our running example where we renamed transfers for the sake of simplicity. Models *M'*, *M''*, and *M'''* are adapted models of *M* where (sets of) transfers are removed or added.

Other observable changes are *model-specific*. Changes

| Type | Subtype | Change | VM | CM |
|----------------|---------------|----------|----|----|
| Non-observable | | Change 1 | x | x |
| Observable | Structural | Change 2 | x | x |
| | # Occurrences | Change 3 | x | |
| | Average value | Change 4 | x | |
| | Message order | Change 5 | | x |

Table 2. Categorizing Value Model Changes

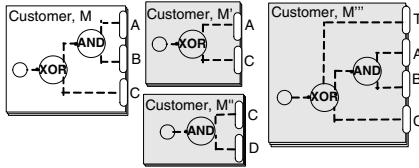


Figure 6. Observable Structural Change

in the estimated number of occurrences (Change 3 in Tab. 2) and in the estimated average value of a transfer (Change 4) are value model specific. Changes in the order of messages (Change 5) are coordination model specific. In general, model-specific changes are related to these parts of the abstraction which are not captured by observable structural change. By going through the abstraction and addressing each part not influenced by an observable structural change, model-specific changes are identified.

Change 3. An *observable change in the number of occurrences* can be achieved by adapting the last element of a quintuple (i.e., number of occurrences). For example, Model 1 in Fig. 5 represents a single actor with 2 customer needs. Adapting this from 2 to 4 results in doubling the last element in the quintuple from 1 to 2.

Change 4. An *observable change in the average value* changes the fourth element of a quintuple (i.e., the average value). As opposed to Change 3 the average value is not the result of other estimations, but of combining information outside the model. For example, information on production costs determine, partially, the average value of a product.

Change 5. An *observable change in the order* reorganizes when exchanges occur. Here, the coordination model depicts payment before delivery. The company might decide to deliver prior to payment as a service to its customers. Changing this order does not affect other parts of the abstraction, it only affects the strict partial order.

7 Maintaining Inter-Model Consistency

In Sect. 5.2 an inconsistency between value model and event log is detected. The estimated average value of a copier is 8000 Euro while data in the event log show 7500

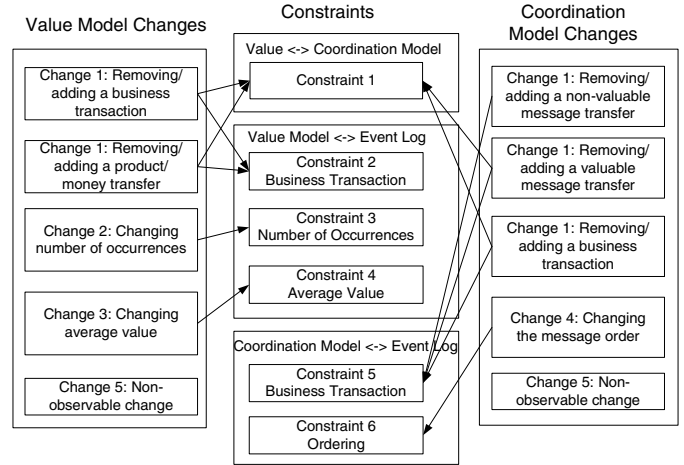


Figure 7. Relating Constraints and Changes

Euro, a violation of Constraint 6. Since models should reflect runtime behavior of the system the value model has to be updated to regain consistency. Here, we propose a structured approach in determining which type of model change is most suitable. Therefore, we identify which changes affect which constraints (cf. Fig. 7). In general, it is determined for each type of model change (cf. Tab. 2) which constraints (cf. Sect. 5) are affected. Each change affects a specific part of the abstraction of a model (e.g., changing the estimated average value of a transfer affects the fourth element of the quintuple representing that transfer in the value model abstraction (cf. Change 4)). This specific part of the abstraction appears in one or more consistency constraints as defined in Sect. 5. For example, the fourth element of a value model tuple (i.e., estimated average value) is part of Constraint 6. Fig. 7 depicts these relations graphically by pointing an arrow from each type of change to the constraints it affects. Here, we can see that in the example case there is one possible type of model adaptation to solve the inconsistency in Constraint 6, namely type Change 4.

In complex models constraint violations could be solved by multiple types of changes (e.g. Constraint 3 can be solved by type Change 1 and 2). However, each type of change might affect more than one constraint (e.g. type Change 2 affects Constraint 3 and Constraint 4). Visualizing these complex relations enables the user to determine the most suitable type of change for a specific constraint violation. The most suitable one will be the change having the least negative effects on other not-violated constraints.

8 Related Work

Several approaches for ensuring consistency between different models at an operational level exist. For exam-

ple, *Business Process Intelligence* (BPI) aims at supporting business and its users in managing process execution quality [7] and acknowledge the importance of inter-model alignment. Recently efforts are made to focus on the analysis of costs related to the use of BPI [11]. Here, Mutschler et al. introduce two cost models. One model analyzes the Total Cost of Ownership while the other model analyzes the impact of BPI on Software Development Efforts. Although in BPI quantifications are made and data is related to process models, BPI focuses on execution quality instead of on the overall performance of the collaboration. Another example is the *Astro-project* [8] where business requirements and business processes are integrated into one framework to enable flexibility. Formal verification of, for example, consistency within the framework can be checked.

Besides checking consistency between different models, there exist constructive approaches guaranteeing consistency of the model derived from another model. For example in [1] an approach is proposed to use an intermediate model as a bridge between a business model and a process model. The approach is based on identifying tasks needed to accomplish the consumer need and to derive the interdependencies of these tasks. [2] propose a *chaining method* to derive from a business model a corresponding process model. Another approach is by Koehler et al. [9] who propose a *pattern based* approach to come from a business process model to a consistent implementation. Model checking techniques are used to automatically verify, for example, consistency. However, these constructive approaches focus only on static consistency.

A well known approach for assessing business models is using *Key Performance Indicators* (KPI). In these approaches, KPI are chosen as evaluation criteria for business models. In [4] KPI are used to overcome the problem of measuring a priori the benefits of E-Commerce investments. The e-business is assessed by business process simulation where users can experiment with different configurations. The resulting simulated values of the KPI are compared with the estimated values in the process models. A business decision is made based on this comparison. In our mechanism, the profitability evaluation criterium can be considered a KPI.

9 Conclusion and Outlook

We describe an approach for defining consistency relations between E-Commerce models. Furthermore, we describe important research in maintaining consistency at the operational level by using model abstractions. As a result, efforts for maintaining consistency between such models are reduced and inter-model consistency can be guaranteed.

We plan to investigate *whether* and *how* an inconsistency should be addressed. For example, if there is a deviation of

only 1 Euro in the average value of a transfer this is in our definition an inconsistency but might not be perceived as such by the stakeholder. Therefore, we plan to investigate *degrees of inconsistency* in different collaborations.

References

- [1] B. Andersson, M. Bergholtz, A. Edirisuriya, T. Ilayperuma, and P. Johannesson. A declarative foundation of process models. In *Proc. of the 17th Int. Conf. on Advanced Inf. Systems Engineering*, pages 233–247, 2005.
- [2] B. Andersson, M. Bergholtz, B. Grégoire, P. Johannesson, M. Schmitt, and J. Zdravkovic. From business to process models - a chaining methodology. In *CAiSE2006*, pages 211–218, Luxembourg, 2006.
- [3] L. Bodenstag, A. Wombacher, and M. Reichert. On formal consistency between value and coordination models. Technical Report TR-CTIT-07-91, Univ. of Twente.
- [4] G. Giaglis, R. Paul, and G. Doukidis. Dynamic modelling to assess the business value of electronic commerce. In *Proc. of the 11th Int. Electronic Commerce Conference*, 1998.
- [5] J. Gordijn, H. Akkermans, and H. van Vliet. Business modelling is not process modelling. In *Proc. of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling*, pages 40–51, London, 2000. Springer-Verlag.
- [6] J. Gordijn and J. M. Akkermans. Value-based requirements engineering: Exploring innovative e-commerce ideas. *Requirements Engineering*, 8(2):114–134, 2003.
- [7] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.-C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004.
- [8] R. Kazhamiak, M. Pistore, and M. Roveri. A framework for integrating business processes and business requirements. In *Proceedings of the Enterprise Distributed Object Computing Conference (EDOC'04)*, pages 9–20, Washington, DC, USA, 2004. IEEE Computer Society.
- [9] J. Koehler, G. Tirenni, and S. Kumaran. From business process model to consistent implementation: A case for formal verification methods. In *Proc. Enterprise Distributed Object Computing Conference (EDOC'02)*, Washington, USA, 2002. IEEE Computer Society.
- [10] W. E. McCarthy. The REA accounting model: a generalized framework for accounting systems in a shared data environment. In *Acc. Review*, volume 57, pages 554–578, 1982.
- [11] B. Mutschler, M. Reichert, and J. Bumiller. An approach to quantify the costs of business process intelligence. In *International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 05)*, pages 152–165, 2005.
- [12] A. Osterwalder and Y. Pigneur. An e-business model ontology for modeling e-business. In *Proc. of the 15th Bled E-Commerce Conference - Constructing the eEconomy*, 2002.
- [13] W. van der Aalst. Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries. *Inf. and Manag.*, 37(2):67–75, 2000.
- [14] S. A. White. Business Process Modelling Notation (BPMN). <http://www.bpmn.org/Documents/OMG-02-01.pdf>, 2006. Visited: May 2, 2007.