**Noname manuscript No.**
(will be inserted by the editor)

# On Formal Consistency between Value and Coordination Models

**Lianne Bodenstaff[1]⋆, Andreas Wombacher[2], Manfred Reichert[1]**

[1] Information Systems Group, Department of Computer Science,
University of Twente, The Netherlands
e-mail: {l.bodenstaff,m.u.reichert}@utwente.nl
[2] School of Computer and Communication Sciences,
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
e-mail: andreas.wombacher@epfl.ch

The date of receipt and acceptance will be inserted by the editor

**Abstract** In information systems (IS) engineering different techniques for modeling inter-organizational collaborations are applied. In particular, value models estimate the profitability for involved stakeholders, whereas coordination models are used to agree upon the inter-organizational processes before implementing them. During the execution of inter-organizational collaboration, in addition, event logs are collected by the individual organizations representing another view of the IS. The combination of the two models and the event log represent the IS and they should therefore be consistent, i.e., not contradict each other. Since these models are provided by different user groups during design time and the event log is collected during run-time consistency is not straight forward. Inconsistency occurs when models contain a conflicting description of the same information, i.e., there exists a conflicting *overlap* between the models. In this paper we introduce an *abstraction* of value models, coordination models and event logs which allows ensuring and maintaining alignment between models and event log. We demonstrate its use by outlining a proof of an inconsistency resolution result based on this abstraction. Thus, the introduction of abstractions allows to explore formal inter-model relations based on consistency.

## 1 Introduction

In information systems (IS) engineering a variety of modelling techniques can be used to specify a system. Each of the resulting models emphasizes a specific aspect

⋆ This research has been supported by the Netherlands Organisation for Scientific Research (NWO) under contract number 612.063.409

of the IS and has its own perspective on it. In this paper we focus on two fundamental perspectives which are of high relevance for modelling collaborations, namely the *value perspective* [5] and the *coordination perspective* [14].

For assessing the collaboration at a business level, *value models* describe *what* is exchanged between participating stakeholders and help to clarify *expectations* of each stakeholder in the collaboration. Value models provide estimations on profitability for companies participating in a collaboration. For example, agreements on the number of transferred products or payments between stakeholders can be modelled. Related to this is value-based software engineering [3]. Here value considerations are integrated with software design. In value-based business modelling, these value considerations are used to evaluate a collaboration.

*Coordination models*, in turn, describe *how* to coordinate interorganizational business processes. More precisely, at the IS level coordination is achieved by describing the order in which messages are exchanged between the stakeholders. This ordering of message control flow is of high importance. For example, whether products are sent out before or after corresponding payments are made influences the degree of risk for stakeholders. A coordination model is used as a basis for implementing a collaboration. When executing it by an IS, in addition, *event logs* are produced containing data about *how* messages are exchanged and whether each intended exchange is *realized*.

The business strategy level and the business process level of an interorganizational collaboration are closely related and highly depend on each other. In particular, the business process level describes how the transfer of objects (e.g. products) specified on the business strategy level is expected to be realized. Transferring several of these objects complete a business transaction. For example, transferring money in return for transferring a product together constitutes a complete business transaction. Both models now describe both the money and product transfer where the value model focusses on the monetary aspect and the coordination model focusses on the order of these transfers. If the overlap in modelled information is conflicting inter-model inconsistencies occur. For example, when the value model depicts transfer of money while this is not depicted in the coordination model.

Fig. 1 depicts the formal relations between value model, coordination model and event log as used in this paper. During *design time* of the models expectations about the behavior of the system are modelled. We refer to *static consistency* (cf. Figure 1) when ensuring inter-model consistency of the overlapping information during design time. Second, we relate data from the event log, which reflects realization of the collaboration, to the value model and the coordination model at *operational level*. This is referred to as *dynamic consistency* (cf. Figure 1) of the overlapping information. Though the relation between value and coordination model is evident, not much work has been done in this area so far. As we will show in this paper such an integration is far from being trivial, particularly when being confronted with the evolving nature of interorganizational collaborations. More precise, the goal is to provide a *model independent method* for ensuring inter-model consistency at design time as well as maintaining inter-model consistency and consistency between models and real life behavior of the system at an operational level. In this paper we present an *abstraction* method for models at business

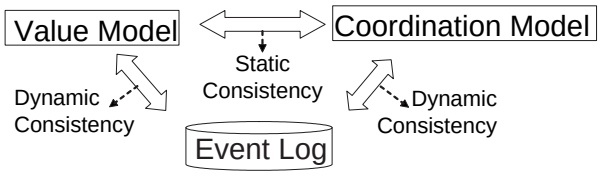


**Fig. 1** Consistency Relations

strategy level and business process level as well as for event logs. We abstract from graphical representations and focus on the formal properties of the models using *set-theory*. This representation enables formal consistency checking between models and running system. When an inconsistency is detected, consistency can be regained by adapting one of the models. However, new inconsistencies might be introduced concerning other consistency relations of this model. Here, we show that in some circumstances, inconsistencies at an operational level between model and event log can be solved while maintaining the other two consistency relations mentioned in Figure 1.

In Section 2 background information on value and coordination modelling as well as event logs is illustrated by means of a running example. Section 3 describes the notion independent abstractions. Section 4 relates value model and coordination model during design time while Section 5 relates event logs with value and coordination model at the operational level. In Section 6 we use these relations to prove that a need for structural changes in the value model does not occur. Section 7 discusses related work and we conclude with a summary and outlook in Section 8.

## 2 Running Example

We introduce basics of value models, coordination models and event logs through a simplified example. Real life collaborations represented as value or coordination model are highly complex and are therefore not suitable for our illustration purpose. Our business case consists of a *online photo service* shop for online ordering of *photos* and *photo albums*. When ordering a product *delivery costs* are charged.

### *2.1 Value Model*

To reason about value transfers and to estimate the income of the online photo service the company develops a *value model*. Fig. 2 depicts the value model of our running example using the e$^3$-value modelling formalism [6] [1]. The actor *online photo service* has a group of *customers*. The example depicts four *value exchanges* between company and customers: money for printing photos (*money1*,

[1] Other value-based modelling techniques (e.g. REA [11] and Business Modelling Ontology [13]) can be used as well. We selected e$^3$-value in this paper due to its graphical notation.

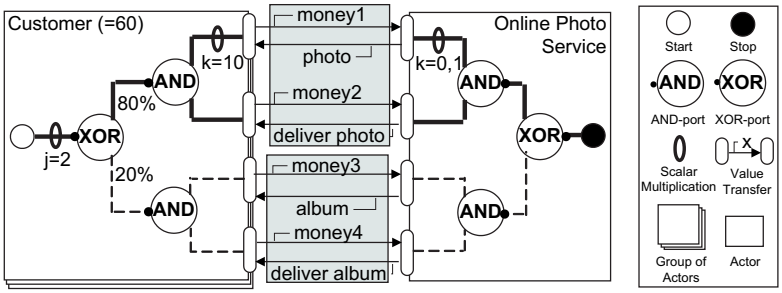


**Fig. 2** Example of a Value Model ($e^3$-value)

*photo*), money for delivering the photos (*money2, deliver_photo*), money for printing a photo album (*money3, album*), and money for delivering the album (*money4, deliver_album*). Interdependent value transfers (i.e., transfers exchanged in one *instance* of the model) are connected through the dotted and solid lines in Fig. 2.

Fig. 2 contains two possible instances. One possible instance in the model is highlighted by a thick line. An instance starts within the customer as a *start stimulus* representing the consumer need for ordering photo products, and ends with an *end stimulus* of the company. To indicate that certain exchanges appear more than once in a single business transaction, *scalar multiplication* is used. Scalar multiplication element *j* for example indicates that on average customers buy 2 photo products per business transaction. Scalar multiplication element *k* within actor online photo service is added for calculation purposes, i.e., it is the counterpart of element *k* in the group of customers indicating that customers order on average 10 photos at once. The *XOR-split* in the dependency path indicates that customers either buy an album or photos. Both *AND-splits* indicate that for every photo or album purchase also delivery costs are paid.

To make an estimation of the income, several quantifications are made for a specified time frame. In Fig. 2, for example, an estimation on how many customers the company expects (=60), how many purchases on average are made in a business transaction (scalar multiplication $j = 2$), and on what the ratio between photo and album purchases is (XOR-split `80%-20%`). These quantifications result in an estimation on the *number* of sold photos, albums and delivery costs for both products. Together with an *average value* of each transfer, this gives an indication on the income for this business activity in the specified time frame, in this case a year. Although these quantifications are part of the model, we represent them for clarification separately (cf. Table 1).

### *2.2 Coordination Model*

The coordination model describes how the collaboration shall be implemented, i.e., which messages are to be exchanged between the different stakeholders. Fur-

| Value Transfer | Average Value, Euros | Number of Occurrences per year |
|---|---|---|
| *photo/money1* | 0.20 | 960 |
| *deliver_photo/money2* | 2 | 96 |
| *album/money3* | 40 | 24 |
| *deliver_album/money4* | 4 | 24 |

**Table 1** Estimations Value Model

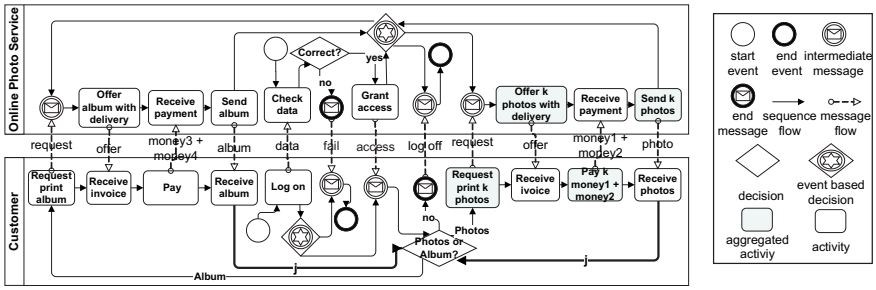


**Fig. 3** Example Coordination Model in BPMN notation

thermore, the coordination model denotes in which order these messages are exchanged. By executing the coordination model and by transferring messages between stakeholders, the value transfers modelled in the value model should be accomplished. The coordination model forms the basis for implementing the collaboration. For representing the coordination model, we use Business Process Modeling Notation (BPMN) [15][2]. Fig. 3 depicts the coordination model of our business case in BPMN notation. The customer first has to *log on* for accessing the upload pages of the online company. Then the customer has a *choice* to order photos or an album. Based on this *message* the company makes an offer after which the customer pays and receives the product. Here, *scalar multiplication elements*, indicating possible repetition of activities (e.g. ordering another set of photos or album), are marked with *j* and *k*, in analogy to the scalar multiplications in the value model in Fig. 2. Note that *k* is included in *request print photos* task since a customer will order *k* photos instead of *k* times a photo. Such an *aggregated activity* is colored grey in the coordination model.

### *2.3 Event Log*

The event log contains realized data gathered from the running system. This data contains real life information on messages exchanged between stakeholders. Furthermore, using *timestamps* shows the order in which data is exchanged. The event log enables traceability of execution processes during collaboration. Furthermore,

[2] Other modelling techniques like Activity Diagrams and Petri Nets are applicable as well. In this paper we selected BPMN due to its graphical 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        copierl
================================================
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>
```

**Fig. 4** Example Event Log in XML

it enables checking whether profitability estimates made in the value model are realized. As an example, take Fig. 4 in which part of an XML based event log (i.e., one business transaction) is shown. It depicts data being exchanged between customer and online photo service company for ordering photos. Here, each message is annotated with a *timestamp*, *sender*, *receiver* and *name*. A message contains a *contract number* and information on the transferred goods. This information entails the value of a good (*Item_amount*), the number of ordered goods (*Number*) and the type of good (*Good*). In Fig. 4 two goods are transferred: *money1* 10 times for 0.20 euro, and *money2* 1 time for 2 euro. Messages with the same contract number belong to one business transaction while one specific customer can have multiple business transactions with the rental company.

## 3 Abstracting Models and Event Log

For relating models and event log, *commonalities* between transfers in models and entries in event log need to be defined. To investigate this, we propose to *abstract* from the used modelling techniques and represent models and event log independent from a particular formalism or notion. Each model-specific abstraction is obtained by focussing on *formal properties* of the model as well as identifying its *overlap* with the other models. Each model-abstraction abstracts from graphical notation and all constructs that fall outside the overlap.

### *3.1 Abstraction of the Value Model*

In our abstraction we abstract from information in the value model which has no overlap with information in the coordination model or event log, and we focus on

- Value model
  - Instance
    - Value Transfer
    - Value Transfer
    - …
  - …

the commonalities. Essential in the value model are the value transfers, with estimations on the number of occurrences, average value, and involved stakeholders. For example, value transfer *photo* is estimated to occur 960 times with an average value of 0.20 euro between customer and online photo service. Furthermore, the value transfers are *grouped* according to the instance they belong to. Each value model consists of a set of instances (cf. the grey areas in Fig. 2) and each instance consists of a set of value transfers (cf. Table 3.1). Also the relations between the different sets and elements (i.e., instances and value transfers), are important. In the set representation of the value model we abstract from the graphical notation as well as from reciprocity (i.e., if an actor receives a value transfer, he also will have to offer one) as well as from any constructs modelled inside an actor.

Now, each value transfer in a value model is represented as a quintuple `x=(a,b,c,d,e)`, where `issuer(x)=a`, `recipient(x)=b` `name(x)=c`, `value(x)=d` and `occurrences(x)=e` represent *issuer*, *recipient*, unique *name*, average *value*, and estimation on the number of *occurrences*, respectively. For example, in Table 1 value transfer *album* is expected to be issued by the online photo service (represented as `op` in the abstraction), received by the customer (represented as `c` in the abstraction), has an average value of 40 euros and occurs 24 times. This is represented as: `Album=(op,c,album,40,24)`. These value transfers are *grouped* into *multisets* according to the instances they belong to. An abstraction of a value model now consists several multisets of instances, which contain value transfers. We use multisets since two identical value transfers may occur in one instance (e.g. two times the transfer of an album with the same value). In Fig. 2 these multisets are depicted as highlighted grey areas. We adopt the most common definition of multisets as used by Jensen [8], where $\mathbb{N}$ denotes the set of all natural numbers and $[A \rightarrow B]$ denotes the set of all functions from $A$ to $B$:

**Definition 1 (Multiset)** *A multiset $M$, over a non-empty set $S$, is a function $M \in [S \rightarrow \mathbb{N}]$. The non-negative integer $M(x) \in \mathbb{N}$ is the multiplicity of element $x$ in multiset $M$. An element $x$ belongs to multiset $M$ (i.e. $x \in M$) iff $M(x) \neq 0$. When we write explicitly the elements of multiset $M$, we write $M = \{x_1^{n_1}, x_2^{n_2}, \ldots\}$, where $n_i = M(x_i)$. Furthermore, the cardinality of a multiset is the sum of the multiplicities of its elements (i.e. $card(M) = \sum M(x_i)$).*

To represent *relations* between different value transfers and instances, we identify *scalar multiplication elements* in the value model. For example, scalar multiplication element *k* in Fig. 2 indicates that in one instance order *several* photos can be purchased while only *one* time delivery costs are paid, i.e., there is an *occurrence difference* between photos and delivery costs. To identify transfers related to a scalar multiplication, their name is annotated with the name of the scalar

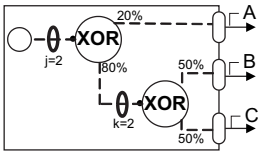


**Fig. 5** Complex Scalar Multiplication Example

multiplication structure. For example, transfer *money1* in Fig. 2 is represented as $(c,op,money1^{jk},0.20,960)$ where *jk* indicates that value transfer *money1* is influenced by scalar multiplication elements *j* and *k*. Annotating these names enables us to identify occurrence relations between transfers, where the same annotation indicates the same number of estimated occurrences. If there exists such an occurrence relation between to value transfers then there should occur a similar relation in event log and coordination model. Therefore, this information is added to the abstraction by annotating the name of each value transfer in the multisets with the scalar multiplication element it is influenced by (cf. Def. 2).

**Definition 2 (Scalar Multiplication)** *Scalar multiplication $\ell$ in value or coordination model, realizing iterations of value or message transfer* x*, is represented as* $name(x)=c^{\ell}$ *in the abstraction of the model. Nested scalar multiplications are represented as* $c^{jk}$.

The two instances (i.e., the two multisets) can occur together in one business transaction. For example, in one business transaction photos as well as an album can be ordered. This is modelled in e$^3$-value by a multiplication element before an XOR-split. For example, scalar multiplication element *j* indicates that in one business transaction on average two orders are done. This can be two photo orders, two album orders or a photo and an album order. To indicate in the abstraction that two multisets can co-occur, they are annotated with the name of the scalar multiplication element which influences the XOR-split. In more complex models, multisets are annotated with the name of the scalar multiplication element before the XOR-split and closest to the start element. For example, the abstraction of the value model in Fig. 5 consists of three multisets which are all annotated with scalar multiplication element *j*. $\left\{ \{A\}_j, \{B\}_j, \{C\}_j \right\}$. The algorithm for creating the abstraction of a value model is now as follows:

1. Annotate value transfers with the scalar multiplications they are influenced by (cf. Def. 2).
2. Create a multiset for each path (instance) through the value model (cf. thick line in Fig. 2).
3. Identify each XOR-split preceded by one or more scalar multiplications in the value model.
4. Annotate each multiset with a superscript resulting from the XOR-split with the scalar multiplication element closest to the start stimulus.

The abstraction of the value model from Fig. 2 is as follows:

$$\mathscr{V} = \Big\{ \{(c,op,money1^{jk},0.20,960),(c,op,money2^{j},2,96),(op,c,photo^{jk},0.20,960)\}^{j}, \{(c,op,money3^{j},40,24),(c,op,money4^{j},4,24),(op,c,album^{j},40,24)\}^{j} \Big\}$$

### *3.2 Abstraction of the Coordination Model*

Also in the coordination model we focus on the commonalities. Essential are the message transfers with information on involved stakeholders, and the *order* in which they appear. Analogue to the value model abstraction, we group transfers according to the instance they belong to (cf. the grey areas in Fig. 6). Again also the relations between sets and elements are important. We abstract from the graphical notation as well as from any constructs inside an actor. Furthermore, message transfers that do not represent a value transfer in the value model (e.g. *request* in Fig. 3) are also not represented. Now, a message transfer in a coordination model is represented as a quadruplet `x=(index,issuer,recipient,name)` with an initial *index* of zero, *issuer*, *recipient*, and unique *name*. For example, message transfer *album* in Fig. 3 is represented as `(0,op,c,album)` with `index(album)=0`, `issuer(album)=op`, `recipient(album)=c`, and `name(album)=album`. The index will later on be used to identify an *ordering* of messages in the coordination model.

In the coordination model in Fig. 3 scalar multiplication elements are depicted as *choices* for repetition or as *aggregated* messages. For example, the choice to start *Photos or Album?* in the customer or aggregated message *Pay k times money1 + money2* in Fig. 3 depict scalar multiplications. If message transfers are influenced by the same scalar multiplication they occur the same number of times. To create the abstraction of a coordination model, the names of message transfers are annotated with the scalar multiplication elements they are influenced by. Then each scalar multiplication edge in the coordination model is removed, and each aggregated activity (cf. the grey messages in Fig. 3) is transformed into a normal one. Fig. 6 shows the result as a *reduced model*. Now, for each instance in the coordination model a multiset is created and annotated with the scalar multiplication it is influenced by to indicate its relations. Furthermore, each multiset of transfers has a *strict partial order*. The order is partial since there exist aggregated transfers e.g. ordering 10 photos at once, and the ordering is strict since we do not allow the occurrence of two messages at the same time. The algorithm for creating the abstraction of a coordination model is now as follows.

1. Annotate names of message transfers with the scalar multiplications they are influenced by (cf. Def. 2).
2. Create a reduced model by removing the scalar multiplication edges and the scalar multiplication part of activities (cf. Fig. 6).
3. Create a multiset for each instance in the reduced model (cf. Fig. 6).
4. Annotate each multiset with the scalar multiplication element it is influenced by.

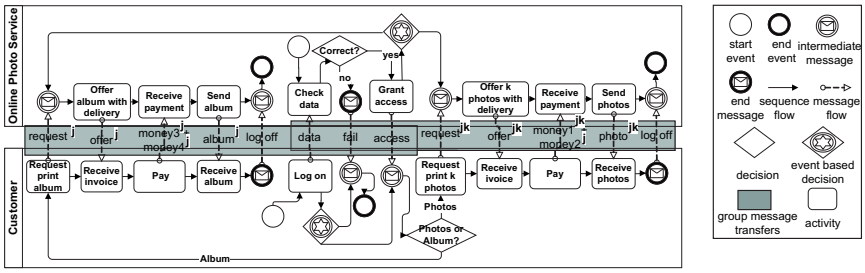


**Fig. 6** Example Reduced Coordination Model

5. Define the strict partial order on the elements in the multisets based on their occurrence in the coordination model.

Using this algorithm, the abstraction of the coordination model in Fig. 3 is as follows.

$$
\begin{aligned}
\mathscr{W} = \Big\{ & \\
& \big(\{(0,c,op,money1^{jk}),(0,c,op,money2^{j}),(0,op,c,photo^{jk})\}_j, \\
& \{(0,c,op,money1^{jk}) \prec (0,op,c,photo^{jk}),\ (0,c,op,money2^{j}) \prec (0,op,c,photo^{jk})\}\big), \\
& \big(\{(0,c,op,money3^{j}),(0,c,op,money4^{j}),(0,op,c,album^{j})\}_j, \\
& \{(0,c,op,money3^{j}) \prec (0,op,c,album^{j}),\ (0,c,op,money4^{j}) \prec (0,op,c,album^{j})\}\big)\Big\}
\end{aligned}
$$

### *3.3 Abstraction of the Event Log*

Essential in the event log are the message entries which contain information on when the message happened (i.e., *timestamp*), who was the *issuer*, *recipient*, what the content of the message is (i.e., *name* of the message), and what the economical *value* is. Furthermore, the relations between the entries are of importance, i.e., to which instance they belong and to which business transaction. An event log entry is now represented as a quintuple containing information on the timestamp, issuer, recipient, name, and value of the entry. For example, 10 entries of *money1* in transfer *photo_payment*, all with value 0.20 (cf. Fig. 4) is represented as: $\texttt{money1} = (1,c,op,money1,0.20)^{10}$. For abbreviation, we represent timestamp "Thu, 17 July 2007 09:23:12" in Fig. 4 in the abstraction as integer 1. Higher integers in the abstraction indicate a later timestamp in the event log.

In the value and coordination model we abstracted information on the relations between the transfers by analyzing the structure of the models. However, in the event log we need a different approach. For example, we need to recognize transfers of ordering photos as transfers belonging to one instance and one business transaction in the event log. This is a widely known problem for which different solutions exist. For this paper we choose to use *identifier messages*. Log on and log off messages are used to identify separate business transactions. Different business transactions can also be identified through *contract numbers*. Each

business transaction, in this case, has a unique contract number. A multiset is created for every business transaction. Furthermore, a *request* message for either a photo order or for an album indicates a new instance within a business transaction. We add information on different instances by annotating elements with a subscript integer. Instances occurring later in time have a higher number. The algorithm to create the abstraction from an event log is as follows.

1. Group messages that have the same contract number into multisets, which can also be identified through a *log on* and *log off* message surrounding the messages.
2. Annotate within the multisets elements (i.e., transfers) which are part of the first instance with a subscript 1, elements part of the second instance with 2, etc.
3. Represent each entry in an event log as an element in a multiset with timestamp, issuer, recipient, name, and value.

The following example shows the abstraction of the event log for one month using the algorithm. Recall that estimations in the value model were made for one year.

$$
\begin{aligned}
\mathscr{E} = \Big\{ & \\
& \{(1,c,op,money1,0.20)^{10}_{2},(1,c,op,money2,2)_{2},(2,op,c,photo,0.20)^{10}_{2}\}, \\
& (3,c,op,money3,40)_{1},(3,c,op,money4,4)_{1},(4,op,c,album,40)_{1}, \\
& \{(5,c,op,money3,42)_{1},(5,c,op,money4,4)_{1},(8,op,c,album,42)_{1}, \\
& (9,c,op,money1,0.20)^{8}_{2},(9,c,op,money2,2)_{2},(12,op,c,photo,0.20)^{8}_{2}, \\
& (13,c,op,money1,0.20)^{12}_{3},(13,c,op,money2,2)_{3},(14,op,c,photo,0.20)^{12}_{3}\}, \\
& \{(6,c,op,money3,50)_{1},(6,c,op,money4,4)_{1},(7,op,c,album,50)_{1}\}, \\
& \{(10,c,op,money1,0.15)^{14}_{1},(10,c,op,money2,2)_{1},(11,op,c,photo,0.15)^{14}_{1}, \\
& (14,c,op,money1,0.22)^{6}_{2},(14,c,op,money2,2)_{2},(15,op,c,photo,0.22)^{6}_{2}\}, \\
& \{(16,c,op,money1,0.20)^{10}_{1},(16,c,op,money2,2)_{1},(17,op,c,photo,0.20)^{10}_{1}\}\Big\}
\end{aligned}
$$

## 4 Relating Models Statically

We introduce a consistency definition to relate value model and coordination model *statically* (cf. Fig. 1), i.e., we investigate their relation during design time based on our abstraction. We define this notion based on auxiliary definitions and previous work by Zlatev et al. [16]. To check whether the models are consistent, we need to *match the multisets* in the two abstractions with each other. Furthermore, we need to establish whether the *relations* between the *elements* and the relations between the *multisets* are equal in both models. Based on these constraints we check whether we can *map* one abstraction to the other. If there exists such a mapping, the models are said to be consistent. Fig. 7 depicts the structure of this section.

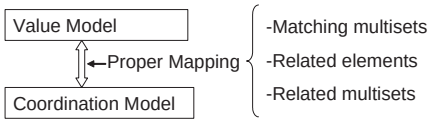


**Fig. 7** Overview of Static Consistency

### *4.1 Matching Multisets*

To check whether the sets representing these models (i.e., the abstractions of the models) are semantically equal (i.e, represent the same multisets of transfers) we need to *match* multisets. Two multisets match if for each element in the multisets there is *exactly* one element in the other multiset that represents the same transfer (i.e., issuer and recipient are the same (i.e., actors match (cf. Def. 3)), and the name is the same). Furthermore, we check whether they both can only appear once in an instance (i.e., not influenced by a scalar multiplication), or that both can appear multiple times in a single instance (i.e., both are influenced by a scalar multiplication). Two elements, i.e., value or message transfers, *match* if either of them is influenced by the *same* scalar multiplication element or none of them is (cf. Def. 4)[3]. Two elements *partially match* if both elements are annotated but not with the same scalar multiplications (cf. Def. 5). For example, in the following set only $\texttt{(i,r,money}^{\ell}\texttt{,c,d)}$ and $\texttt{(i,r,money}^{mn}\texttt{,e,f)}$ partially match because both elements are influenced by scalar multiplication elements: $\{\texttt{(i,r,money,a,b)},\texttt{(i,r,money}^{\ell}\texttt{,c,d)},\texttt{(i,r,money}^{mn}\texttt{,e,f)}\}$. For matching multisets both matching and partially matching elements are sufficient since we only check whether it is possible for the elements to occur more than one time.

**Definition 3 (Matching Actors)** *Let x and y be elements of a multiset. Then x and y have matching actors, actor(x,y), iff:*

1. `issuer(x)=issuer(y)`, *and*
2. `recipient(x)=recipient(y)`.

**Definition 4 (Matching Elements)** *Let x and y be elements of a multiset. Then x and y are matching elements, match(x,y), iff actor(x,y) and* `name(x)=name(y)`.

**Definition 5 (Partially Matching Elements)** *Let x and y be elements of a multiset, and let $\ell$ and k represent one or more scalar multiplications. Then x and y are partially matching elements, pmatch(x,y), iff actor(x,y)* $\wedge$ $\texttt{name(x)=a}^{\ell}$ $\wedge$ $\texttt{name(y)=a}^{k}$.

**Definition 6 (Matching Multisets)** *Multiset M and N match ($M \cong N$) iff*

1. $\forall x \in M: \exists! y \in N \ \wedge (match(x,y) \vee pmatch(x,y))$, *and*
2. $\forall y \in N: \exists! x \in M \ \wedge (match(x,y) \vee pmatch(x,y))$.

[3] We use this simplified matching definition realizing the semantic dimension of the problem [16]. However, for the course of this paper the actual matching is not affecting our results since it is an equivalence relation.

### *4.2 Related Elements*

Elements in one multiset that are influenced by the same scalar multiplication have the *same degree of freedom*, i.e., the are occurring the same number of times within one business transaction. For example, in Fig. 2 *money3* and *money4* are both influenced by scalar multiplication element $_{\mathtt{j}}$. This means that if you pay *x* times *money3*, you also pay *x* times *money4*. Transfers which are influenced by the same scalar multiplication, or not influenced by any scalar multiplication, are said to be *related*, i.e., their *estimated number of occurrences* is equal. Identifying these transfers is important because related transfers in the value model also have to be related, i.e., have the *same degree of freedom*, in the coordination model and event log.

**Definition 7 (Related Elements)** *Let x and y be elements of multisets, and let $\ell$ represent one or more scalar multiplications. Then x and y related elements, $rel(x, y)$, iff actor(x,y)* $\wedge$ $\mathtt{name(x)=a}^{\ell}$ $\wedge$ $\mathtt{name(y)=b}^{\ell}$.

**Definition 8 (Related Element Pairs)** *Let M be a multiset. Then the set of names of related element pairs of transfers is defined as follows.* $S_M := \{(\mathtt{name(x)},\mathtt{name(y)}) | x, y \in M \wedge rel(x, y)\}$.

### *4.3 Related Multisets*

In the running example, two different instances are identified, i.e., ordering photos and ordering an album. In one business transaction a customer can order a combination of these instances. For example, a customer can place two photo orders and an album order in one business transaction. This *co-occurrence* of instances in one business transaction can in the value model be identified by an XOR-split, indicating two different instances, with *before* the XOR-split a multiplication element, indicating the possibility of more than one occurrence of an instance in one business transaction. In the coordination model this can be identified by a *back edge* at the end of an instance back to a decision element. To identify which instances (i.e., multisets) can co-occur in one business transaction, multisets in value and coordination model are annotated with the multiplication element closest to the start stimulus they are influenced by. For consistency both models should have the same multisets that can co-occur.

**Definition 9 (Related Multisets)** *Let M and N be multisets. Then M and N are related multisets, $M =_{\ell} N$, if they are annotated with the same scalar multiplication element subscript.*

**Definition 10 (Related Multiset Pairs)** *Let set $\mathscr{V}$ be the abstraction of value or coordination model. Then the set of related multiset pairs is defined as follows.* $S_{\mathscr{V}} := \{(\mathtt{M},\mathtt{N}) | M, N \in \mathscr{V} \wedge M =_{\ell} N\}$.

### *4.4 Proper Mapping*

We define a mapping from one abstraction to another by matching the multisets of the two different abstractions to each other (cf. Def. 6), by matching for all multisets related multiset pairs (cf. Def. 9), and by matching sets of related pairs of elements (cf. Def. 10). Two models are *statically consistent* if their abstractions have a proper mapping.

**Definition 11 (Mapping Abstractions)** *Let sets $\mathscr{V}$ and $\mathscr{W}$ be abstractions of value or coordination models. Let $T_{\mathscr{V}}$ be the set of related multiset pairs in $\mathscr{V}$, and let $T_{\mathscr{W}}$ be the set of related multiset pairs in $\mathscr{W}$. Then mapping $\nu$: $\mathscr{V} \rightarrow \mathscr{W}$ is defined as:*

1. $\forall M \in \mathscr{V}: \exists! N \in \mathscr{W}: \nu(M,N) \wedge N \cong M$, *and*
2. $\forall N \in \mathscr{W}: \exists! M \in \mathscr{V}: \nu(M,N) \wedge M \cong N$.

*Mapping $\nu$: $\mathscr{V} \rightarrow \mathscr{W}$ is a proper mapping iff:*

1. $\forall \nu(M,N): \forall (M,M') \in T_{\mathscr{V}}: \exists (N,N') \in T_{\mathscr{W}}$, *and*
2. $\forall \nu(M,N): \forall (N,N') \in T_{\mathscr{W}}: \exists (M,M') \in T_{\mathscr{V}}$, *and*
3. *Let $S_M$ be the set of related pairs of elements in M and let $T_N$ be the set of related pairs of elements in N: $\forall \nu(M,N): S_M = T_N$.*

**Definition 12 (Static Consistency)** *Let set $\mathscr{V}$ be the abstraction of value model $\tilde{V}$ and let set $\mathscr{W}$ be the abstraction of coordination model $\tilde{W}$. Then $\tilde{V}$ and $\tilde{W}$ are considered to be statically consistent iff there exists a proper mapping $\nu$: $\mathscr{V} \rightarrow \mathscr{W}$.*

We check *static consistency* between the value and coordination model of our running example. According to Definition 12 these models are statically consistent if their abstractions, i.e., the sets representing the models, have a proper mapping. Recall the abstraction of the value model from Fig. 2 ($\mathscr{V}$), and the abstraction of the coordination model in Fig. 3 ($\mathscr{W}$):

$$\mathscr{V} = \Big\{ \{(\mathtt{c},\mathtt{op},\mathtt{money1}^{jk},\mathtt{0.20},\mathtt{960}),(\mathtt{c},\mathtt{op},\mathtt{money2}^{j},\mathtt{2},\mathtt{96}),(\mathtt{op},\mathtt{c},\mathtt{photo}^{jk},\mathtt{0.20},\mathtt{960})\}^{j}, \{(\mathtt{c},\mathtt{op},\mathtt{money3}^{j},\mathtt{40},\mathtt{24}),(\mathtt{c},\mathtt{op},\mathtt{money4}^{j},\mathtt{4},\mathtt{24}),(\mathtt{op},\mathtt{c},\mathtt{album}^{j},\mathtt{40},\mathtt{24})\}^{j} \Big\}$$

$$\mathscr{W} = \Big\{ (\{(\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money1}^{jk}),(\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money2}^{j}),(\mathtt{0},\mathtt{op},\mathtt{c},\mathtt{photo}^{jk})\}_j, \{(\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money1}^{jk}) < (\mathtt{0},\mathtt{op},\mathtt{c},\mathtt{photo}^{jk}), (\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money2}^{j}) < (\mathtt{0},\mathtt{op},\mathtt{c},\mathtt{photo}^{jk})\}), (\{(\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money3}^{j}),(\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money4}^{j}),(\mathtt{0},\mathtt{op},\mathtt{c},\mathtt{album}^{j})\}_j, \{(\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money3}^{j}) < (\mathtt{0},\mathtt{op},\mathtt{c},\mathtt{album}^{j}), (\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money4}^{j}) < (\mathtt{0},\mathtt{op},\mathtt{c},\mathtt{album}^{j})\}) \Big\}$$

Based on the first two items of Definition 11, each multiset contained in the sets must have a matching multiset in the other set. The first multiset of the abstraction of the value model ($\mathscr{P}_{vm}$ for ordering Photos), matches the first multiset of the abstraction of the coordination model ($\mathscr{P}_{cm}$ for ordering Photos) since each transfer in these multisets has a matching transfer in the other multiset. For example, value transfer $(\mathtt{c},\mathtt{op},\mathtt{money1}^{jk},\mathtt{0.20},\mathtt{960})$ matches message transfer $(\mathtt{0},\mathtt{c},\mathtt{op},\mathtt{money1}^{jk})$

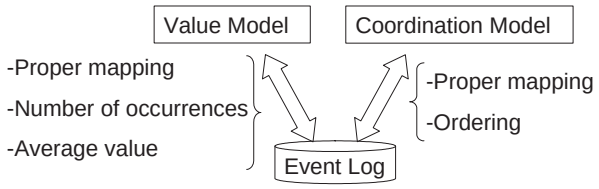


**Fig. 8** Overview of Dynamic Consistency

since both have the same issuer, recipient, name, and equal name annotation (cf. Definition 6). Furthermore, if there is a mapping between multiset A and B then related multisets of A should match related multisets of B (cf. item 1 and 2 of the constraints for a proper mapping in Def. 11). For example, multiset $\mathscr{P}_{vm}$ is mapped to $\mathscr{P}_{cm}$. $\mathscr{P}_{vm}$ is related to the second multiset in the abstraction of the value model $\mathscr{A}_{vm}$ (for ordering an album). Multiset $\mathscr{P}_{cm}$ is related to multiset $\mathscr{A}_{cm}$ (for ordering an album). Therefore, multisets $\mathscr{A}_{vm}$ and $\mathscr{A}_{cm}$ should be part of mapping $v$, i.e. they should match. Based on item 3 for the constraints of a proper mapping in Def. 11, the sets of related pairs of elements of matching multisets have to be equal. We define set $S1$ and $S2$ as the sets of related pairs of the two multisets representing the value model and set $T1$ and $T2$ as the sets of related pairs of the two multisets representing the coordination model. $S1 = T1 = \{(\texttt{money1}^{jk}, \texttt{photo}^{jk})\}$ and $S2 = T2 = \{(\texttt{money3}^{j},\texttt{album}^{j}),(\texttt{money3}^{j},\texttt{money4}^{j}),(\texttt{album}^{j},\texttt{money4}^{j})\}$. Therefore, we conclude that the value model in Fig. 2 is statically consistent with the coordination model in Fig. 3.

## 5 Relating Models and Event Log Dynamically

At the operational level we relate value model and event log, as well as coordination model and event log (cf. Fig. 1). An overview of the structure of this section is depicted in Fig. 8For both dynamic relations we first need to define a *proper mapping* by matching multisets of the event log with the two models. In addition for relating value model and event log, the estimated *number of occurrences* has to be compared with the realized number of occurrences. Furthermore, the estimated *average value* of a transfer has to be compared with the realized average value of a transfer. In addition for relating coordination model and event log, the *ordering* in the coordination model should be equal to the realized order in the event log. First we discuss the mapping from event log to the model after which we relate event log and coordination model, depicted as a *dynamic consistency* in Fig. 1. Finally, Section 5.3 relates event log and value model also denoted as *dynamic consistency* in Fig. 1.

### *5.1 Relating Multisets - A Proper Mapping*

To find a proper mapping between event log and value or coordination model, we need to relate every *occurred* business transaction (i.e., every multiset) in the

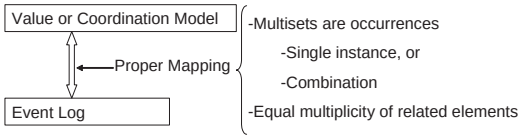


**Fig. 9** Overview of Proper Mapping

event log to a business transaction in value or coordination model. An overview of the proper mapping is given in Fig. 9 One multiset in the event log can be mapped to a *single* instance (i.e., multiset) in value or coordination model or to a *combination* of instances (i.e., combination of multisets). For example, if a multiset in the event log represents purchasing one album, it can be mapped to one multiset in the abstraction of the value model. However, when a customer orders an album and photos in one business transaction then this is mapped to a combination of both multisets in the value model. Furthermore, if two elements in a multiset in the value or coordination model are related (i.e., occur an equal number of times) then they should occur also with the same multiplicity in the event log entry. An element in an event log is said to be an *occurrence* of an element in value or coordination model if actors and name are equal.

**Definition 13 (Element Occurrence)** *Let x and y be elements of multisets, and let $\ell$ represent one or more scalar multiplications. Then x is considered to be an occurrence of y, occ*$(x, y)$*, iff match(x,y)* $\vee$*(actor(x,y)* $\wedge$ `name(x)=a` $\wedge$ `name(y)=a`$^{\ell}$*).*

A multiset in the event log is an occurrence of a multiset in value or coordination model if each element in the event log multiset is an occurrence of an element in the value or coordination model multiset. Furthermore, each element in the multiset of the value or coordination model should also occur in the multiset of the event log. As a third constraint, if two elements in value or coordination model are related (i.e., they should occur the same number of times), their multiplicity in the multiset of the event log should be equal. The multiplicity of element $x$ in multiset $M$ is denoted as: $M(x)$.

**Definition 14 (Set Occurrence)** *Let M and N be multisets, let $\ell$ represent one or more scalar multiplications, and let set $S_N$ be the set of related pairs of elements (c.f. Definition 10) in N. Then M is considered to be an occurrence of N, occ*$(M, N)$*, iff*

1. $\forall y \in N: \exists x \in M \wedge occ(x, y)$*, and*
2. $\forall x \in M: \exists y \in N \wedge occ(x, y)$*, and*
3. $\forall (x, y) \in S_N, \rightarrow M(x) = M(y)$*.*

Since a multiset in the event log can be mapped to a combination of value or coordination model multisets, we formalize how two related multisets in value or coordination model are united (cf. Def. 16 and Def. 17). First we define two union operators (cf. Def. 15). The first joins two multisets where the multiplicity

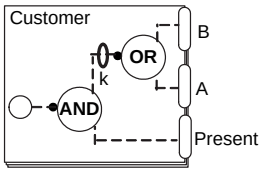


**Fig. 10** Single Element Occurrence

of each element the maximum is of the multiplicity of that element in the united multisets. The second joins two multisets where the multiplicity of each element the summation is of the multiplicity of the element in both multisets.

**Definition 15 (Multiset Union Operators)** *Let M be a multiset of M', and let N be a multiset over N'. The set union of multiset M and N resulting in multiset P, $P = M \cup N$, is defined as follows. $S := M' \cup N'$, with P over set S: $\forall x \in S : P(x) = max(M(x), N(x))$. The multiset union of multiset M and N resulting in multiset P, $P = M \uplus N$, is defined as follows. $S := M' \cup N'$, with P over set S: $\forall x \in S : P(x) = M(u) + N(u)$.*

Recall that related elements have an equal number of occurrences. To create correct related elements in the united multiset we have to rename the annotation of the elements. Elements of different multisets can have a different number of occurrences and are therefore not related. For example, although *money2* and *money3* (cf. Fig. 2) are both influenced by scalar multiplication *j*, if they occur in the same business transaction they occur a different number of times. Furthermore, elements that do not have an annotation (i.e., occur only once), will appear also once in the united multiset. For example, Fig. 10 depicts that each customer receives a present at the end of a business transaction. It does not matter how many A's or B's the customer orders in a single business transaction, he receives always one present. The two instances (i.e., multisets) of this value model both contain the *present* transfer without annotation since it is not influenced by any loop. Now, the union of these two instances should also contain one element representing the present.

**Definition 16 (Union of Value Model Multisets)** *Let M and N be multisets in the abstraction of a value model. Let $A_M := \{x \mid x \in M \wedge \texttt{name(x)=a}^{\ell}\}$, let $A_N := \{x \mid x \in N \wedge \texttt{name(x)=a}^{\ell}\}$, let $B_M := \{x \mid x \in M \wedge \texttt{name(x)=a}\}$, and let $B_M := \{x \mid x \in N \wedge \texttt{name(x)=a}\}$. Then the union of M and N is constructed as follows: $M \uplus\!\!\uplus N = (A_M \uplus A_N) \uplus (B_M \cup B_N)$.*

The union of coordination model multisets is equal to the union of value model multisets and in addition the *order* in which messages appear is united. If a customer first orders a set of photos, then an album, and then another album, then this matches with the union of first the photo-multiset with the album-multiset, after which this is united with another photo-multiset. Now, not only the order *within* a multiset is important but also the order *between* elements of the different multisets.

Therefore, the multiset which is added to the union, gets an *index number* which is one higher than the highest index number of the current multiset. This index number is used to define the order between the elements of the different multisets ($<_{new}$).

**Definition 17 (Union of Coordination Model Multisets)** *Let $M$ and $N$ be multisets with a strict partial order $<_M$ and $<_N$ in the abstraction of a coordination model. Let multisets $A_M$, $B_M$, $A_N$, and $B_N$ be defined as follows.*
$A_M := \{x \mid x \in M \wedge \texttt{name(x)=a}^{\ell}\}$.
$B_M := \{x \mid x \in M \wedge \texttt{name(x)=a}\}$.
$A_N := \{x \mid x \in N \wedge \texttt{name(x)=a}^{\ell}\}$.
$B_N := \{x \mid x \in N \wedge \texttt{name(x)=a}\}$.
*Let $k$ be the highest index number in multiset $M$.*
$k := max_{x \in M}(\texttt{index(x)})$.
*Let multiset $A_N^{'}$ and $B_N^{'}$ be defined as follows.*
$A_N^{'} := \{(k+1, \texttt{issuer(x)}, \texttt{recipient(x)}, \texttt{name(x)}) \mid x \in A_N\}$.
$B_N^{'} := \{(k+1, \texttt{issuer(x)}, \texttt{recipient(x)}, \texttt{name(x)}) \mid x \in B_N\}$.
*Let the strict partial order on $A_N^{'}$ and $B_N^{'}$ be based on $N$ and defined as follows.*
$<_{N'} := \{(x' < y') \mid x', y' \in (A_N^{'} \cup B_N^{'}) \wedge x, y \in N \wedge match(x', x) \wedge match(y', y) \wedge (x < y) \in <_N\}$.
*Let the strict partial order between multisets $A_M$, $B_M$, $A_N^{'}$, and $B_N^{'}$ be defined as follows.*
$<_{new} := \{(x' < y') \mid x' \in (A_M \cup B_M) \wedge y' \in (A_N^{'} \cup B_N^{'})\}$.
*Then the union of $M$ and $N$ is as follows. $M \uplus N = (A \uplus A') \uplus (B \cup B')$ with a strict partial order: $<_M \cup <_{N'} \cup <_{new}$*

To reason about the multisets which are used to create a union, we define an *element of a union* as a multiset which is used to create the union multiset.

**Definition 18 (Elements and Sets of Unions)** *Let $\mathscr{V}$ be the abstraction of a value model. Then set of all possible unions multisets in $\mathscr{V}$, $\mathscr{U}_V$ is defined as follows. $\mathscr{U}_V := \{U \mid U = \circledast_{i=1}^{n} U_{\ell}^{(i)} \wedge U_{\ell}^{(i)} \in \mathscr{V}\}$. Let $U$ be a union of multisets with $n \in \mathbb{N}^+$: $U = \circledast_{i=1}^{n} U_{\ell}^{(i)}$. If $\exists U_{\ell}^{(i)} = M$ then $M$ is said to be element of the union $U$ denoted as: $M \in^* U$.*

In the value and coordination model value and message transfers agreed upon by the stakeholders are modelled. Therefore, *only* these transfers should occur during runtime and appear in the event log. Furthermore, value and message transfers agreed upon should *indeed* occur during runtime. Therefore, the event log should contain data about every agreed upon value and message transfer. This relation between event log and models is defined as a *proper mapping* $\mu$ from event log to model. Based on Definition 14 and the union definitions:

**Definition 19 (Proper Mapping)** *Let $\mathscr{E}$ be the abstraction of an event log, let $\mathscr{V}$ be the abstraction of a value or coordination model, and let $\mathscr{U}_V$ be the set of all possible unions of multisets in $\mathscr{V}$. Then mapping $\mu : \mathscr{E} \rightarrow \mathscr{U}_V$ is denoted as:*

– $\forall N \in \mathscr{E} : \exists M \in \mathscr{U}_V : \mu(N, M) : occ(N, M)$.

*Mapping $\mu$ is a proper mapping iff:*

– $\forall M \in \mathscr{V}: \exists N \in \mathscr{E}: \mu(N, M') \wedge M \in^* M'$.

### *5.2 Relating Event Log and Coordination Model*

For defining *dynamic consistency* between coordination model and event log we use the constraint of a proper mapping. Furthermore, the order in which the event log entries occur should match the order of the message transfers as specified in the coordination model. We define a strict partial order for event log multisets using their timestamps. Then, we define an *equivalence* relation between the order in the event log multiset and the order in the coordination model multiset based on a proper mapping between them.

**Definition 20 (Order Equivalence)** *Let $E$ be a multiset in event log abstraction $\mathscr{E}$. Then strict partial order $<_E$ over $E$ is defined as follows. $<_E := \{(x < y) \mid x, y \in E \wedge time(x) < time(y)\}$. Let $M$ be a multiset with strict partial order $<_N$. Let $\mu(E, M)$ be a proper mapping. Then $<_E = <_N$ if $\forall (m < m') \in <_M: \exists (e < e') \in <_E : occ(e, m) \wedge occ(e', m')$*

**Definition 21 (Dynamic Consistency)** *Let set $\mathscr{E}$ represent event log $\tilde{E}$, let set $\mathscr{W}$ represent coordination model $\tilde{W}$, let $\mathscr{U}_W$ be set of all possible unions of multisets in $\mathscr{W}$. Then $\tilde{E}$ and $\tilde{W}$ are dynamically consistent iff*

1. *$\exists$ proper mapping $\mu : \mathscr{E} \rightarrow \mathscr{U}_W$ (cf. Definition 19), and*
2. *$\forall \mu(M, N): <_M = <_N$.*

To check dynamic consistency between event log and coordination model of our running example, we first find a proper mapping between the abstraction of the coordination model ($\mathscr{W}$) and the abstraction of the event log ($\mathscr{E}$) after which we check whether the order in this mapping is equal:

$$\mathscr{W} = \Big\{ \Big( \big\{ (0,c,op,money1^{jk}), (0,c,op,money2^{j}), (0,op,c,photo^{jk}) \big\}_j, \big\{ (0,c,op,money1^{jk}) < (0,op,c,photo^{jk}), (0,c,op,money2^{j}) < (0,op,c,photo^{jk}) \big\} \Big), \Big( \big\{ (0,c,op,money3^{j}), (0,c,op,money4^{j}), (0,op,c,album^{j}) \big\}_j, \big\{ (0,c,op,money3^{j}) < (0,op,c,album^{j}), (0,c,op,money4^{j}) < (0,op,c,album^{j}) \big\} \Big) \Big\}$$

$$\mathscr{E} = \Big\{ \big\{ (1,c,op,money1,0.20)_2^{10}, (1,c,op,money2,2)_2, (2,op,c,photo,0.20)_2^{10} \big\}, (3,c,op,money3,40)_1, (3,c,op,money4,4)_1, (4,op,c,album,40)_1, \big\{ (5,c,op,money3,42)_1, (5,c,op,money4,4)_1, (8,op,c,album,42)_1, (9,c,op,money1,0.20)_2^{8}, (9,c,op,money2,2)_2, (12,op,c,photo,0.20)_2^{8}, (13,c,op,money1,0.20)_3^{12}, (13,c,op,money2,2)_3, (14,op,c,photo,0.20)_3^{12} \big\}, \big\{ (6,c,op,money3,50)_1, (6,c,op,money4,4)_1, (7,op,c,album,50)_1 \big\}, \big\{ (10,c,op,money1,0.15)_1^{14}, (10,c,op,money2,2)_1, (11,op,c,photo,0.15)_1^{14}, (14,c,op,money1,0.22)_2^{6}, (14,c,op,money2,2)_2, (15,op,c,photo,0.22)_2^{6} \big\}, \big\{ (16,c,op,money1,0.20)_1^{10}, (16,c,op,money2,2)_1, (17,op,c,photo,0.20)_1^{10} \big\} \Big\}$$

Each multiset in the event log is an occurrence of a union of multisets in the coordination model. For example, the first multiset in the event log ($\mathscr{E}_1$) is an occurrence of the union of the second multiset $\mathscr{P}$ (for photo order) in the coordination model with the first $\mathscr{A}$ (for album order). This union is allowed since both multisets are annotated with the same subscript: $j$. To create this union we first rename the annotation of the elements in the second multiset:

$$\begin{array}{l}\mathscr{A} = \\ (\{(0,c,op,money3^{j}),(0,c,op,money4^{j}),(0,op,c,album^{j})\}, \\ \{(0,c,op,money3^{j}) < (0,op,c,album^{j}),\ (0,c,op,money4^{j}) < (0,op,c,album^{j})\}) \\ \rightarrow \\ (\{(0,c,op,money3^{i}),(0,c,op,money4^{i}),(0,op,c,album^{i})\}, \\ \{(0,c,op,money3^{i}) < (0,op,c,album^{i}),\ (0,c,op,money4^{i}) < (0,op,c,album^{i})\})\end{array}$$

Now, multisets $A_P$ and $B_P$ from the photo order, and multisets $A_A$, $B_A$, $A_A^{'}$ and $B_A^{'}$ of the album order are created according to Def. 17. Since each element is annotated in both multisets, multisets $B_P$, $B_A$ and $B_A^{'}$ are empty. The highest index of multiset $\mathscr{P}$ is 0, therefore the index of the elements in multiset $A_A^{'}$ is 1, indicating that the album is ordered later than the photos:

$$\begin{array}{l}A_P = \{(0,c,op,money1^{jk}),(0,c,op,money2^{j}),(0,op,c,photo^{jk})\} \\ A_A = \{(0,c,op,money3^{i}),(0,c,op,money4^{i}),(0,op,c,album^{i})\} \\ A_A^{'} = \{(1,c,op,money3^{i}),(1,c,op,money4^{i}),(1,op,c,album^{i})\}\end{array}$$

The strict partial order of the elements in multiset $A_A^{'}$ (and $B_A^{'}$, which is empty), $<_{A'}$ is equal to the order of the original set $\mathscr{A}$ where only the annotation and index number is different.

$$<_{A'} = \{(1,c,op,money3^{i}) < (1,op,c,album^{i}),\ (1,c,op,money4^{i}) < (1,op,c,album^{i})\}$$

Each element part of ordering photos occurs before any element part of ordering the album. This is the new ordering $<_{new}$ between the elements of the different multisets:

$$\begin{array}{l}<_{new} = \{ \\ (0,c,op,money1^{jk}) < (1,op,c,album^{i}),\ (0,c,op,money1^{jk}) < (1,c,op,money3^{i}), \\ (0,c,op,money1^{jk}) < (1,c,op,money4^{i}),\ (0,c,op,money2^{j}) < (1,op,c,album^{i}), \\ (0,c,op,money2^{j}) < (1,c,op,money3^{i}),\ (0,c,op,money2^{j}) < (1,c,op,money4^{i}), \\ (0,op,c,photo^{jk}) < (1,op,c,album^{i}),\ (0,op,c,photo^{jk}) < (1,c,op,money3^{i}), \\ (0,op,c,photo^{jk}) < (1,c,op,money4^{i})\}.\end{array}$$

As a last step, we create union $U$ by $A_P \uplus A_A^{'}$ with a new strict partial ordering $<_P \cup <_{A'} \cup <_{new}$:

$A_P \uplus A'_A = \{$(0,c,op,money1$^{jk}$),(0,c,op,money2$^{j}$),(0,op,c,photo$^{jk}$),
(1,c,op,money3$^{i}$),(1,c,op,money4$^{i}$),(1,op,c,album$^{i}$)$\}$
$<_P \cup <_{A'} \cup <_{new} = \{$
(0,c,op,money1$^{jk}$) $<$ (0,op,c,photo$^{jk}$), (0,c,op,money2$^{j}$) $<$ (0,op,c,photo$^{jk}$),
(1,c,op,money3$^{i}$) $<$ (1,op,c,album$^{i}$), (1,c,op,money4$^{i}$) $<$ (1,op,c,album$^{i}$),
(0,c,op,money1$^{jk}$) $<$ (1,op,c,album$^{i}$), (0,c,op,money1$^{jk}$) $<$ (1,c,op,money3$^{i}$),
(0,c,op,money1$^{jk}$) $<$ (1,c,op,money4$^{i}$), (0,c,op,money2$^{j}$) $<$ (1,op,c,album$^{i}$),
(0,c,op,money2$^{j}$) $<$ (1,c,op,money3$^{i}$), (0,c,op,money2$^{j}$) $<$ (1,c,op,money4$^{i}$),
(0,op,c,photo$^{jk}$) $<$ (1,op,c,album$^{i}$), (0,op,c,photo$^{jk}$) $<$ (1,c,op,money3$^{i}$),
(0,op,c,photo$^{jk}$) $<$ (1,c,op,money4$^{i}$)$\}$.

The first multiset in the event log, $\mathscr{E}_1$, is an occurrence of union $U$ since each element in $\mathscr{E}_1$ is an occurrence of an element in $U$. For example, (1,c,op,money3,40) is an occurrence of (1,c,op,money3$^{i}$) since the actors as well as the names are equal (Def. 13). In this way, we can find a proper mapping from the event log to the coordination model. Furthermore, the ordering of the messages in coordination model and event log is equal for the mapping. Using Def. 20 we define strict partial order $<_{\mathscr{E}_1}$:

$<_{\mathscr{E}_1} = \{$
(1,c,op,money1,0.20)$_2$ $<$ (2,op,c,photo,0.20)$_2$, (1,c,op,money1,0.20)$_2$ $<$ (3,c,op,money3,40)$_1$,
(1,c,op,money1,0.20)$_2$ $<$ (3,c,op,money4,4)$_1$, (1,c,op,money1,0.20)$_2$ $<$ (4,op,c,album,40)$_1$,
(1,c,op,money2,2)$_2$ $<$ (2,op,c,photo,0.20)$_2$, (1,c,op,money2,2)$_2$ $<$ (3,c,op,money3,40)$_1$,
(1,c,op,money2,2)$_2$ $<$ (3,c,op,money4,4)$_1$, (1,c,op,money2,2)$_2$ $<$ (4,op,c,album,40)$_1$,
(2,op,c,photo,0.20)$_2$ $<$ (3,c,op,money3,40)$_1$, (2,op,c,photo,0.20)$_2$ $<$ (3,c,op,money4,4)$_1$,
(2,op,c,photo,0.20)$_2$ $<$ (4,op,c,album,40)$_1$, (3,c,op,money3,40)$_1$ $<$ (4,op,c,album,40)$_1$,
(3,c,op,money4,4)$_1$ $<$ (4,op,c,album,40)$_1$ $\}$.

The order of $U$ is equal to the order of $\mathscr{E}_1$ since $<_{\mathscr{E}_1} = <_U$ according to Def. 20. Each element in strict partial order $<_U$ is present in strict partial order $<_{\mathscr{E}_1}$ where the elements in $<_{\mathscr{E}_1}$ are occurrences of elements in $<_U$. For example, the elements of ((1,c,op,money1,0.20)$_2$ $<$ (2,op,c,photo,0.20)$_2$) $\in <_{\mathscr{E}_1}$ are occurrences of the elements in ((0,c,op,money1$^{jk}$) $<$ (0,op,c,photo$^{jk}$)) $\in <_U$.

### *5.3 Relating Event Log and Value Model*

To relate event log and value model at operational level we check the existence of a proper mapping between event log and value model (cf. Item 1 Def. 24). Furthermore, we check whether the estimated number of *occurrences* is equal to the realized number of occurrences and whether the estimated *average value* of a transfer is equal to the realized average value of a transfer. The realized number of occurrences is calculated by counting all occurrences of a specific transfer in the different multisets of the event log. For example, we count the number of occurrences of value transfer *photo* in the different multisets in the event log. The average value is the total value of all these transfers divided by the total number of occurrences. However, in Fig. 10 transfer *present* is part of both instances

of the value model, namely {A, Present} and {B, Present}. If business transaction {A,B,Present} occurs in the event log, the occurrence and value of this transfer is *divided* between multiset {A,Present} and {B,Present} since both multisets contributed to the occurrence of transfer Present. To calculate what ratio of a transfer is awarded to a multiset, we need to calculate the total number of multisets containing the transfer that where used to make the mapping. This *ratio* of an element is calculated in Def. 22.

**Definition 22 (Element Ratio)** *Let $U$ and $N$ be multisets with proper mapping $\mu(N, U)$. Let $M \in^* U$. Let $x \in M$. Define $S := \{M' \mid M' \in^* U \wedge x \in M'\}$. Then $ratio(x, M, N, U) = \frac{1}{|S|}$.*

The value model is defined over time period $t_1$ (e.g. a year) and consistency is checked using data gathered in time period $\delta$ (e.g. one month). The number of realized occurrences in the event log should now be equal to $\frac{\delta}{t_1}$ (e.g. $\frac{1}{12}$) times the estimated number of occurrences of that transfer in the value model. The first subset of the abstraction of the event log, $\mathcal{S}$ in Def. 23 contains all multisets which are an occurrence of multiset $M$. Subset $\mathcal{S}'$ contains all multisets which are an occurrence of a *union* of value model multisets and $M$ is part of this union. When an event log entry is an occurrence of $M$ we simply calculate all occurrences of the elements and what their average value is. However, when multiset $M$ occurs in a union and it contains elements which are not influenced by a scalar multiplication element (e.g. *Present* in Fig. 10) then these elements should only be counted for a certain percentage. Namely, the ratio calculated in Def. 22. Furthermore, for each multiset we require that the realized average value of each transfer is equal to the estimated average value. For example, we check whether the total number of occurrences of transfer *photo* in the event log is equal to the estimated number of occurrences *photo* in the value model. Again, elements which are not influenced by a scalar multiplication element are only accounted for a certain percentage. The multiplicity of element $x$ in multiset $M$ is denoted as: $M(x)$.

**Definition 23 (Dynamically Consistent Multisets)** *Let $\mathscr{V}$ be an abstraction of a value model, let $\mathscr{U}_V$ be the set of possible unions of multisets in $\mathscr{V}$, let $M \in \mathscr{V}$, let $\mathscr{E}$ be an abstraction of an event log, let $\mathcal{S} \subseteq \mathscr{E}$, let $\mathcal{S}' \subseteq \mathscr{E}$, let $\mu: \mathscr{E} \rightarrow \mathscr{U}_V$ be a proper mapping, and let $t_1$ be the time period used in the value model. $M$ is $\delta$-dynamically consistent with $(\mathcal{S}, \mathcal{S}', \mu)$ iff $\forall x \in M \wedge name(x) = a^{\ell}$:*

1. $\sum_{N \in (\mathcal{S} \cup \mathcal{S}')} \sum_{y \in N \wedge occ(y,x)} N(y) = occurrences(x)\frac{\delta}{t_1}$, *and*
2. $\sum_{N \in (\mathcal{S} \cup \mathcal{S}')} \sum_{y \in N \wedge occ(y,x)} \frac{value(y)}{N(y)} = value(x)$,

*and iff $\forall x \in M \wedge name(x) = a$:*

1. $\sum_{N \in \mathcal{S}} \sum_{y \in N \wedge occ(y,x)} N(y) + \sum_{N \in \mathcal{S}'} \sum_{y \in N \wedge occ(y,x) \wedge \mu(N,M')} ratio(x, M, N, M') \times N(y) = occurrences(x)\frac{\delta}{t_1}$, *and*
2. $\frac{\sum_{N \in \mathcal{S}} \sum_{y \in N \wedge occ(y,x)} value(y) + \sum_{N \in \mathcal{S}'} \sum_{y \in N \wedge occ(y,x) \wedge \mu(N,M')} ratio(x,M,N,M') \times value(y)}{\sum_{N \in \mathcal{S}} \sum_{y \in N \wedge occ(y,x)} N(y) + \sum_{N \in \mathcal{S}'} \sum_{y \in N \wedge occ(y,x) \wedge \mu(N,M')} ratio(x,M,N,M') \times N(y)} = value(x)$.

Based on these constraints we define $\delta$-dynamic consistency between event log and value model. There should be a proper mapping and each multiset of the value model, representing an instance, should be consistent with all its occurrences in the event log.

**Definition 24 (Dynamically Consistent Models)** *Let set $\mathcal{V}$ represent value model $\tilde{V}$, let $U = \circledast_{i=1}^{n} U_{\ell}^{(i)}, U_{\ell}^{(i)} \in \mathcal{V}$, let set $\mathcal{E}$ represent event log $\tilde{E}$, let $\mathcal{S}_{t,M} := \{N \mid N \in \mathcal{E} \wedge occ(N, M) \wedge \mu(N, M) \wedge N$ was realized between $t - \delta$ and $t$ $(t > \delta > 0)\}$, and let $\mathcal{S}'_{t,M} := \{(N \mid N \in \mathcal{E} \wedge M \in^{*} U \wedge occ(N, U) \wedge \mu(N, U) \wedge N$ was realized between $t - \delta$ and $t$ $(t > \delta > 0)\}$. Then $\tilde{V}$ and $\tilde{E}$ are $\delta$-dynamically consistent at time $t$ iff*

1. $\exists$ *proper mapping* $\mu: \mathcal{E} \rightarrow U$ *(Definition 19),*
2. $\forall M \in \mathcal{V}$: *M is* $\delta$*-dynamically consistent at time t with* $(\mathcal{S}_{t,M}, \mathcal{S}'_{t,M}, \mu)$.

For checking dynamic consistency between value model and event log of our running example, we first confirm a proper mapping. Recall the abstraction of the value model ($\mathcal{V}$) and the abstraction of the event log ($\mathcal{E}$):

$$\mathcal{V} = \Big\{ \{(\mathtt{c,op,money1}^{jk},\mathtt{0.20,960}),(\mathtt{c,op,money2}^{j},\mathtt{2,96}),(\mathtt{op,c,photo}^{jk},\mathtt{0.20,960})\}^{j}, \{(\mathtt{c,op,money3}^{j},\mathtt{40,24}),(\mathtt{c,op,money4}^{j},\mathtt{4,24}),(\mathtt{op,c,album}^{j},\mathtt{40,24})\}^{j} \Big\}$$

$$\mathcal{E} = \Big\{ \{(\mathtt{1,c,op,money1,0.20})_2^{10},(\mathtt{1,c,op,money2,2})_2,(\mathtt{2,op,c,photo,0.20})_2^{10}\}, (\mathtt{3,c,op,money3,40})_1,(\mathtt{3,c,op,money4,4})_1,(\mathtt{4,op,c,album,40})_1, \{(\mathtt{5,c,op,money3,42})_1,(\mathtt{5,c,op,money4,4})_1,(\mathtt{8,op,c,album,42})_1, (\mathtt{9,c,op,money1,0.20})_2^{8},(\mathtt{9,c,op,money2,2})_2,(\mathtt{12,op,c,photo,0.20})_2^{8}, (\mathtt{13,c,op,money1,0.20})_3^{12},(\mathtt{13,c,op,money2,2})_3,(\mathtt{14,op,c,photo,0.20})_3^{12}\}, \{(\mathtt{6,c,op,money3,50})_1,(\mathtt{6,c,op,money4,4})_1,(\mathtt{7,op,c,album,50})_1\}, \{(\mathtt{10,c,op,money1,0.15})_1^{14},(\mathtt{10,c,op,money2,2})_1,(\mathtt{11,op,c,photo,0.15})_1^{14}, (\mathtt{14,c,op,money1,0.22})_2^{6},(\mathtt{14,c,op,money2,2})_2,(\mathtt{15,op,c,photo,0.22})_2^{6}\}, \{(\mathtt{16,c,op,money1,0.20})_1^{10},(\mathtt{16,c,op,money2,2})_1,(\mathtt{17,op,c,photo,0.20})_1^{10}\}\Big\}$$

Every multiset in the abstraction of the event log is an occurrence of a multiset or union of multisets in the value model (Item 1, Definition 24). Furthermore, all occurrences of each multiset in the value model should be $\delta$-dynamically consistent at time $t$ (Item 2, Definition 24) with the multisets in the event log it is mapped to or to which a union containing it is mapped to. In our example $t$ = July 1, 2007, the current date, and $\delta$ = 1 month, the time captured in the event log. Recall the simplified timestamp notation in the abstraction. The second multiset in the abstraction of the event log is an occurrence of the union of the second (album, *A*) with the first (photo, *P*1) and the first (photo, *P*2) multiset in the abstraction of the value model. According to Def. 16, multiset $A_A = \{(\mathtt{c,op,money3}^{j},\mathtt{40,24}),(\mathtt{c,op,money4}^{j},\mathtt{4,24}),(\mathtt{op,c,album}^{j},\mathtt{40,24})\}$, multiset $A_{P1} = \{(\mathtt{c,op,money1}^{lm},\mathtt{0.20,960}),(\mathtt{c,op,money2}^{l},\mathtt{2,96}),(\mathtt{op,c,photo}^{lm},\mathtt{0.20,960})\}$, multiset $A_{P2} = \{(\mathtt{c,op,money1}^{no},\mathtt{0.20,960}),(\mathtt{c,op,money2}^{n},\mathtt{2,96}),(\mathtt{op,c,photo}^{no},\mathtt{0.20,960})\}$, and multisets $B_A, B_{P1}, B_{P2} = \emptyset$ since all elements are annotated.

$$A \uplus P1 \uplus P2 = \{(\mathtt{c,op,money3}^{j},\mathtt{40,24}),(\mathtt{c,op,money4}^{j},\mathtt{4,24}),(\mathtt{op,c,album}^{j},\mathtt{40,24}), (\mathtt{c,op,money1}^{lm},\mathtt{0.20,960}),(\mathtt{c,op,money2}^{l},\mathtt{2,96}),(\mathtt{op,c,photo}^{lm},\mathtt{0.20,960}), (\mathtt{c,op,money1}^{no},\mathtt{0.20,960}),(\mathtt{c,op,money2}^{n},\mathtt{2,96}),(\mathtt{op,c,photo}^{no},\mathtt{0.20,960})\}$$

| Mapping Constraint | Additional Constraint | Cause |
|---|---|---|
| Proper mapping exists | Mismatch number of occurrences | Cause 1 |
| | Mismatch average value | Cause 2 |
| No proper mapping | Missing multiset in value model | Cause 3 |
| | Missing occurrence multiset in event log | Cause 4 |

**Table 2** Categorization of Dynamic Inconsistency between Value Model and Event Log

For each element in this multiset, the realized number of occurrences should be equal to the estimated number of occurrences (Item 1, Definition 23). Timeframe $t_1$ in the value model is one year. For example, the realized number of *money1* transfers is 2 in the event log and the estimated number of occurrences for *money1* in the value model is $24\frac{1}{12} = 2$. Furthermore, the realized average value should be equal to the estimated average value (Item 2, Definition 23). For instance, the realized average value of *money1* is $\frac{38}{2} + \frac{42}{2} = 19 + 21 = 40$ and is therefore equal to the estimated average value in the value model. When conducting the complete calculations, value model and event log prove to be 1 month-dynamically consistent at July 1, 2007.

## 6 Addressing the Necessity for Structural Change

When a dynamic inconsistency between value model and event log occurs, this can be addressed by changing part of the value model to regain consistency. The different possible *changes* in a value model are categorized (cf. Section 6.2). Here, we prove that some important causes for inconsistencies will not occur if coordination model and event log are strictly related (cf. Section 6.3). As a result, some changes possible in the value model to regain consistency will never be required. By ruling out necessity of certain changes in the value model, the process of adapting the value model to regain consistency becomes more efficient. Furthermore, investigating formal properties improves our understanding of relations between value model, coordination model and event log.

### *6.1 Categorization of Causes for Dynamic Inconsistency between Value Model and Event Log*

An inconsistency between value model and event log may arise due to violation of constraints for dynamic consistency between value model and event log (cf. Definition 23 and 24). A corresponding categorization is depicted in Table 2. Of course, these causes can appear in combination.

**Cause 1** *The number of estimated occurrences of a value transfer in a specific multiset is not equal to the realized number of occurrences of that value transfer in multisets of the event log* (see Item 1, Definition 23). For example, assume that the value model estimates that 24 scooters will be rented. If the event log shows that only 20 scooters have been rented, an inconsistency will occur.

**Cause 2.** *The estimated average value of a value transfer in a specific multiset is not equal to the realized average value of transfers in multisets of the event log* (see Item 2, Definition 23). For example, in the value model it is estimated a scooter is rented for an average of 40 euros. If the event log shows an average of only 35 euros, an inconsistency will occur.

**Cause 3.** *There is an occurrence in the event log which does not match with any multiset in the abstraction of the value model.* This means that there is a business transaction not represented in the value model. This results in the nonexistence of a proper mapping. There is either a missing combination of value transfers, for example, when the event log contains entries of a car and scooter rented together with one insurance then this will not match with a multiset in the abstraction of the value model depicted in Fig. 2 where cars and scooters have separate insurances. Or there will be a complete value transfer missing in the value model, for example, if the event log contains entries of bike rentals while there is no value transfer *Bike* present.

**Cause 4.** *A multiset in the abstraction of the value model has no occurrence in the event log.* This means that one business transaction is never executed. This results in the nonexistence of a proper mapping. For example, if the event log has not shown any entries of scooter rentals this will be inconsistent with Fig. 2.

### *6.2 Categorization of Value Model Changes*

We show how to restore consistency between value model and event log. When an inconsistency emerges we focus on *adaptations of the value model*. We start with a definition for value models belonging to the same class. Then we introduce the four changes illustrated with $e^3$-value examples. Typically, there is more than one way to structure a value model while preserving the same options for business transactions, i.e., facilitating the same set of multisets. For example, though *Model 1* and *Model 2* in Fig. 11 are different models, they facilitate the same set of multisets. The abstraction M1 and M2 of the models are equal. Value transfers *A* and *B* are estimated to be transferred once in a single business transaction from issuer (i) to recipient (r) for the value of 5 and 3 euros, respectively, and value transfer *B* is also estimated to be transferred once in a single transaction for 3 euros. We introduce *classes of value models* to aggregate differently structured value models with the same functionality, i.e., facilitating the same set of multisets, and differently structured models with different functionality. When two models facilitate the same functionality (e.g. Models 1 and 2 in Fig. 11) they are considered to belong to the same *class*. This definition is based on Definition 11 where models are statically equivalent if they facilitate the same set of multisets.

**Definition 25 (Class)** *Let set $\mathscr{V}$ represent value model $\tilde{V}$ and set $\mathscr{W}$ represent value model $\tilde{W}$. Then: $\tilde{V}$ and $\tilde{W}$ belong to the same class if $\mathscr{V}$ and $\mathscr{W}$ are statically equivalent according to Definition 11.*

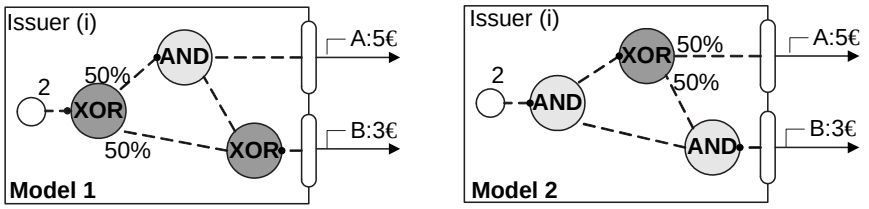


**Fig. 11** Same Class Models in e$^3$-value Notation

| Type | Subtype | Sub-subtype | Change |
|---|---|---|---|
| Observable | Structural | | Change 1 |
| | Non-structural | Number of occurrences | Change 2 |
| | | Average value | Change 3 |
| Non-observable | | | Change 4 |

**Table 3** Categorization of Value Model Changes

An overview of the changes is given in Table 3. The first distinction is between *observable* and *non-observable* changes. An observable change adapts the abstraction of the value model while a non-observable change does not. The abstraction of the value model consists of *multisets*, each multiset consists of *quintuples* (issuer, recipient, name, value, occurrences). Changing the set either means adding or deleting a multiset, adding or deleting a quintuple, or changing a value in the quintuple. The first two options, adding or deleting a multiset or quintuple, are considered to be *structural* changes and the last one, changing a value in the quintuple, is considered to be a *non-structural* change. Changing a value in the quintuple means adapting the *number of occurrences* or adapting the *average value*.

**Change 1.** An *observable structural change* adds or removes a multiset or part of a multiset of value transfers. In e$^3$-value observable structural changes are the result of adding or deleting one or more *constructs* (e.g. XOR-port, AND-port and value transfer) preserving a valid value model. Fig. 12 depicts three observable structural changes represented as an e$^3$-value model. Model $M$ with set $\{\{A,B\},\{C,D\}\}$ is the original model of our running example where we renamed the transfers for the sake of simplicity. Model $M'$ with set $\{\{A,B\},\{C\}\}$ is an adapted model of $M$ where part of a multiset is removed, namely $D$, by deleting an *AND-port*. Model $M''$ with set $\{\{C,D\}\}$ is another adapted model of $M$ where a multiset is removed by deleting an *XOR-port* and all constructs following. Finally, model $M'''$ with set $\{\{A,B\},\{C,D\},\{T\}\}$ is an adapted model of $M$ where a multiset is added by adding an exit to the *XOR-port* and adding transfer $T$.

**Lemma 1 (Observable Structural Change)** *An observable structural change in a value model changes its class.*

*Proof* According to Definition 11, two sets will be statically equivalent if all multisets *match*. If they are statically equivalent then they will belong to the same

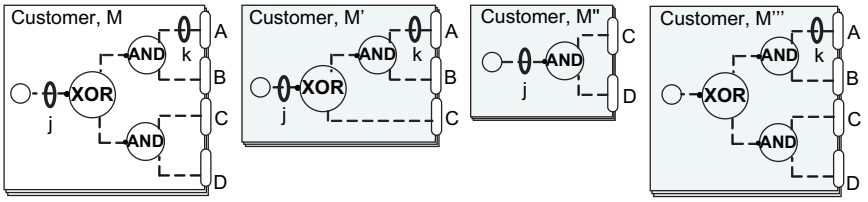


**Fig. 12** Changes Category 1

*class*. By adding or deleting a multiset or part of a multiset, the new set and the original one are not statically equivalent. As a consequence, the new abstraction of the value model belongs to a different class than the original set. Also changes in *issuer, recipient* or *name* results in a class change since such a change influences the equivalence relation (Definition 11). □

**Change 2.** An *observable non-structural change of the number of occurrences* can be achieved by adapting the value of the last element of a quintuple (i.e., the number of occurrences). This number of occurrences results from combining other estimations (e.g. number of business transactions and size of the group of actors). For example, Model 1 in Fig. 11, represents a single actor, with 2 consumer needs (specified at the start stimulus); 50 percent is expected to be fulfilled by transferring *A* and *B* and 50 percent is expected to be fulfilled by transferring *B*. This is depicted in abstraction M1 (cf. Fig. 11). Adapting the estimated number of occurrences means related estimations also get adjusted. For example, if we assume 4 consumer needs instead of 2, the expected number of occurrences of the value transfer (i.e., the value of the last element in the quintuple) will double.

**Change 3.** An *observable non-structural change of the average value* changes the fourth element of a quintuple (i.e., the average value). As opposed to Change 2 the average value of a transfer is not the result of other estimations in the value model, but is estimated by combining information outside the model. For example, information on production and material costs determine, partially, the average value of a product. When the estimated rental price of a scooter is higher than the realized average value, this could be the result of discounts given to specific customers. Adapting the average value does not affect other parts of the abstraction or model.

**Change 4.** *Non-observable changes* in a value model do not influence parts of the model that have a representation in the coordination model or event log. The abstraction of the value model only depicts parts present in both models and event log. Either (i) such a change does not affect the semantics of the model or (ii) part of the model, with no connection to the abstraction of it, is changed. In case (i) let, for example, in Fig. 11 Model 2 be the adapted and Model 1 the original model. Then this is a non-observable change since the sets representing the models are the same for Model 1 and Model 2. In (ii), for example, adapting transfer *Insurance1* in Fig. 2 does not influence the abstraction of the model since there is no matching message in coordination model or entry in the event log.

| Assumption | Inconsistency Type | Change Type | Effects |
|---|---|---|---|
| Dynamic consistency (coordination model and event log), and Static consistency | Proper Mapping: Cause 1 or 2 | Observable Non-structural: Change 2 or 3 | No introduction of static inconsistency |

**Table 4** Line of Argumentation

### *6.3 Evolution of Value Models through Adaptation*

We aim at maintaining consistency while *minimizing changes* in the models and *without introducing static inconsistencies*. We assume that the implementation is based on statically consistent value and coordination models (cf. Definition 12). We show that when coordination model and event log are dynamically consistent (cf. Definition 21), maintaining dynamic consistency (cf. Definition 24) between value model and event log can be achieved *without structural changes* (Changes 2 and 3) and without introducing new inconsistencies (see Table 4). We use the definitions and categorizations from the previous sections.

**Theorem 1** *Assume event log and coordination model are dynamically consistent. Assume further that value and coordination model are statically consistent. Then it is always possible to solve dynamic inconsistency between value model and event log while preserving static consistency between value model and coordination model.*

*Proof* When assuming dynamically consistent event log and coordination model, and static consistency, there is always a proper mapping (see Lemma 2). Dynamic inconsistency between value model and event log can then be solved through a non-structural change (see Lemma 3). A non-structural change does not influence static consistency (see Lemma 4). By transitivity: If event log and coordination model are dynamically consistent then dynamic inconsistency between value model and event log can be solved without influencing static consistency. □

**Lemma 2** *Assume event log and coordination model are dynamically consistent. Assume further that value and coordination model are statically consistent. Then: There is always a proper mapping between the abstraction of the event log and the abstraction of the value model.*

*Proof* If there is no proper mapping there must either be a multiset in the value model not present in the event log (Item 1, Definition 19), or there is a multiset in the event log not present in the value model (Item 2, Definition 19). Violation of Item 1 means that this multiset in the value model has to match with a multiset in the coordination model due to static consistency (cf. Definition 12). Because of dynamic consistency between coordination model and event log, this multiset must have an occurrence in the event log (cf. Definition 21). If this multiset has an occurrence in the event log it will be an occurrence of the value model. However, this contradicts our initial assumption $\perp$.

Violation of Item 2 means that the multiset in the event log has to be an occurrence of a multiset in the coordination model due to dynamic consistency between coordination model and event log (Definition 21). This multiset in the coordination model must have a matching multiset in the value model because of static consistency (Definition 12). However, the multiset in the coordination model, violating Item 2, is an occurrence of this multiset in the value model $\perp$. □

**Lemma 3** *Assume event log and coordination model are dynamically consistent. Assume further that value and coordination model are statically consistent. Then: It is always possible to resolve dynamic inconsistency between value model and event log by adapting the value model with an observable non-structural change.*

*Proof* If event log and coordination model are dynamically consistent there will be a proper mapping (cf. Lemma 2). If there is a proper mapping and a dynamic inconsistency between value model and event log this will be either due to Cause 1 (i.e., a mismatch in the estimated number of occurrences) or due to Cause 2 (i.e., a mismatch in the average value of a transfer) (Table 2). By transitivity, if event log and coordination model are dynamically consistent then dynamic inconsistency between value model and event log can be solved through an observable non-structural change. □

**Lemma 4** *An observable non-structural change in the value model does not influence static consistency between value model and coordination model.*

*Proof* If a non-structural change would influence static consistency between value model and coordination model then it would influence static equivalence between the abstraction of the value model as well as the abstraction of the coordination model (Definition 12). To influence static equivalence, a non-structural change has to add or remove a multiset from the abstraction of the value model (Definition 11). A non-structural change (see Change 2 or Change 3 in Section 6.2) only adapts values of elements in a quintuple and never influences multisets. Therefore, a non-structural change will never influence static consistency. □

## 7 Related Work

Several approaches for ensuring consistency between different models at an operational level exist. For example, *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 [12]. 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 focusses on execution quality instead of on the overall performance of the collaboration. Another example is the *Astro-project* [9] 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. However, this constructive approach focusses only on static consistency. Another approach is by Koehler et al. [10] 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.

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.

## 8 Conclusion and Outlook

We describe important research concerning consistency relations between models and system at the operational level by using abstractions. Using these abstractions allows formal property checking of the relations between value model, coordination model, and event log. Furthermore, we propose a categorization of dynamic inconsistencies between value model and event log, and a categorization of changes in value models. Using these categorizations and our abstraction method, we prove that several changes in the value model can be discarded when solving dynamic inconsistency if certain circumstances are met. Furthermore, we prove that by using these changes no new inconsistencies are introduced. As a result, efforts for maintaining consistency at the operational level are reduced and consistency between value model, coordination model, and event log are guaranteed.

We plan to investigate *whether* and *how* an inconsistency should be addressed. For example, if there is a derivation of only one euro in the average value of a transfer this is in our definition an inconsistency but might not be perceived by the stakeholder as an inconsistent result. Furthermore, we plan to investigate *automated consistency checking* based on our abstraction.

## 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. B. Boehm. Value-based software engineering: reinventing. *ACM SIGSOFT Software Engineering Notes*, 28(2):1–7, 2003.
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. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Springer, 1997. Three Volumes.
9. R. Kazhamiakin, 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.
10. 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.
11. W. E. McCarthy. The REA accounting model: a generalized framework for accounting systems in a shared data environment. In *Accounting Review*, volume 57, pages 554–578, 1982.
12. 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.
13. 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.
14. W. van der Aalst. Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries. *Inf. and Management*, 37(2):67–75, 2000.
15. S. A. White. Business Process Modelling Notation (BPMN). http://www.bpmn.org/Documents/OMG-02-01.pdf, 2006. Visited: May 2, 2007.
16. Z. Zlatev and A. Wombacher. Consistency between $e^3$-value models and activity diagrams in a multi-perspective development method. In *OTM Conferences (1)*, pages 520–538, 2005.