Consistency Between *e*^{3}*-value* Models and Activity Diagrams in a Multi-perspective Development Method

Zlatko Zlatev and Andreas Wombacher

University of Twente, Department of Computer Science, Information System Group,
P.O. Box 217, 7500 AE Enschede, The Netherlands
Z.V.Zlatev@ewi.utwente.nl
A.Wombacher@utwente.nl

Abstract. Multi-perspective approaches to analysis and design of businesses information systems are used to manage the complexity of the development process. A perspective contains a partial specification of the system from a particular stakeholder's standpoint. This separation of concerns leads to potential inconsistencies between specifications from different perspectives, resulting in non-implementable systems. In this paper, a consistency relationship between the *economic value* and *business processes* perspectives of a design framework for networked businesses is proposed based on an equivalence of a common semantic model.

1 Introduction

The development of an information system to support a business is a complex process. Many stakeholders with distinctive interests are involved in the alignment of information system capabilities and business objectives. A standard approach to manage the complexity of such a process is the adoption of a multi-perspective development method, where responsibilities for design and analysis of distinctive aspects of the system are localized in separate perspectives. We use a framework with three perspectives, including: (i) *economic value*, in which we model the creation of value among networked businesses and analyze the incentives for them to take part in such a network; (ii) *business processes*, in which we model the coordination of activities realizing the exchanges of economic values; and (iii) *application communication*, in which we model the data exchange among the components of the information system that supports the business.

A multi-perspective approach presupposes a decentralized development process, the benefits of which come at a price of potential inconsistencies between models from different perspectives. In particular, specifications are inconsistent if it is not possible to build a single system that correctly implements the specification of each perspective. As a consequence, specifications are consistent, if an implementation of the specifications exists. Therefore, specifications need to be checked for consistency to identify required changes in the final design.

The inconsistencies occur due to different design methodologies, opposing stakeholders' goals, conflicting knowledge and incompatible modeling notations. In particular, inconsistencies emerge from the redundant information in different

R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760, pp. 520 – 538, 2005.

models; i.e., perspectives overlap in their responsibilities for modeling certain aspects of the system. Therefore, we base our consistency check on the common concepts and relations of two modeling notations. We call such a limited modeling notation a *reduced model* as it contains only constructs present in both models. We define consistency between pairs of models, which is a necessary condition for global consistency. The common concepts and relations of the three models are too limited, with regard to the precision of a consistency check involving all three perspectives, resulting in unusable consistency decisions.

We operationalize our consistency definition by transforming the specifications from different perspectives to reduced models and by defining an equivalence relationship between reduced models. Hence, a specification may represent different scenarios and because the concepts of separating these scenarios are incomparable in the different perspectives, a single reduced model represents a single alternative scenario of the original specification. (An example of an alternative in a process model is an execution sequence without decision point.) Thus, a single specification may produce more than one reduced model and the equivalence relationship is defined between sets of reduced models.

To illustrate the proposed approach, we use exemplary modeling notations, although, other notations could have been used. We represent the value perspective with the e^3-*value* modeling notation [7], the business process perspective with UML Activity diagram [10 pages 3-155—3-169], and the application communication perspective with, e.g., communication diagram [14 pages 201—211] or UML Component diagram [10 pages 3-169—3-171]. This selection of modeling notations impacts the particular common concepts but is invariant to the proposed approach based on reduced models.

The contribution of this paper is the definition of consistency between e^3-*value* models and activity diagrams. Further, we discuss applying the presented approach to the remaining two consistency relationships: e^3-*value* model and communication diagram, and activity diagram and communication diagram.

The paper is structured as follows: In Section 2, we present an example business case and its models from value and process perspectives. Additionally, we provide an intuitive consistency definition which we use later as a validation criterion. In Section 3, we define the reduced model between the e^3-*value* notation and UML Activity diagram and introduce our consistency definition. Further in Section 4, we discuss the impact of model granularity on our consistency definition. Section 5 validates the proposed consistency definition by a comparison with the intuitive consistency definition. The last three sections discuss implications for the application communication perspective, related work, and conclusion and future work.

2 Example

We consider a business case with the following businesses taking part: a buyer, a seller, and a shipping company. The seller has a shop and a warehouse at two different locations. It can directly sell products to customers only from the shop. If a product is purchased that is not present in the shop then a delivery from the warehouse must be made. A shipping company is paid to arrange the logistics to the

buyer's home. The seller processes two payment methods: (1) in a case of an off-the-shelf product, the seller requires immediate payment in cash; (2) in a case of a purchase from the warehouse, the seller allows late payment by, e.g., a bank transfer.

2.1 Economic Value Perspective

We use the *e³-value* modeling notation [7] to represent the value aspect of a business model. Below, we explain the semantics of the concepts we use. We refer to Fig. 1 for the graphical representation on the concepts. An *actor* is an economically independent entity modeled as a rectangle. A *value interface* indicates which value object is available, in return for another value object. It is shown by a rounded box, connected to an actor. A *value exchange* represents that two actors are willing to exchange value objects with each other. It is a prototype for actual trades between actors and is shown by a line. A *value object* represents a value for one or more actors: e.g., services, goods, money, or consumer experiences. Value objects are shown as text.

A *dependency path* represents the internal coordination within actors. It shows via which value interfaces an actor must exchange value objects, given the exchange of objects via another interface of that same actor. A dependency path is a set of dependency nodes and connections. A *dependency node* is a *stimulus* (represented by a bullet), an *OR-fork* (represented by a triangle), or an *end node* (represented by a bull's eye). A stimulus represents a consumer need that triggers the chain of exchanges; an OR-fork represents alternative paths; and an end node represents the model boundary. A *dependency connection* connects dependency nodes and value interfaces, represented by a broken line.

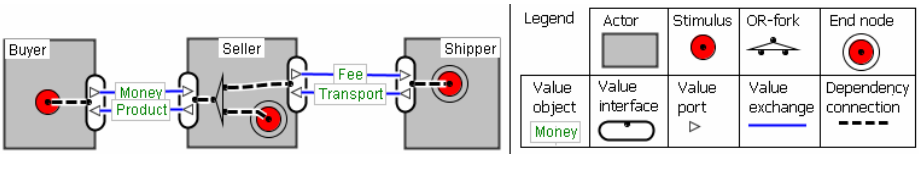

Fig. 1. Value model of the example business case, using *e³-value* modeling notation

Fig. 1 shows a value model of the example business case described above. It contains the three businesses connected by value exchanges. The buyer is willing to give money in return for a product. The two potential exchanges are with the seller, in the middle of the figure, who has an interest in the same value objects. The dependency path, which starts in the buyer, is split at the seller. This is an OR-fork which exemplifies that some products are handed to the buyer immediately while others must be transported by a shipping company. The seller and shipper exchange the value objects `Fee` and `Transport` which are paired together reciprocally.

2.2 Business Processes Perspective

The *e³-value* model focuses on the pairing of objects that have economic value for businesses. We now discuss the coordination of activities performed by each business

to achieve the exchange of value objects. We use an UML Activity diagram [10 pages 3-155—3-169] to represent the business processes perspective of our example.

Fig. 2 shows the sequence of actions performed during a purchase of a product. The process starts with the buyer requesting a product. Her order is processed and two outcomes are possible: either the desired product is present in the shop; or the product must be reserved and shipped from the warehouse. These options are represented in Fig. 2 as a choice in the seller's swimlane. In case the path to the left (marked with 1 in the figure) is followed then the product is handed directly and payment in cash is received in return. In the second case (marked with 2 in the figure), a reservation is made. This is followed by two parallel branches which represent the payment of an invoice and the transportation of the product. The latter requires coordination with the logistics provider, which is shown as message exchanges between the seller and shipper swimlanes. The actual delivery of the product is represented in the bottom of Fig. 2 as a message from the shipper swimlane to the buyer.

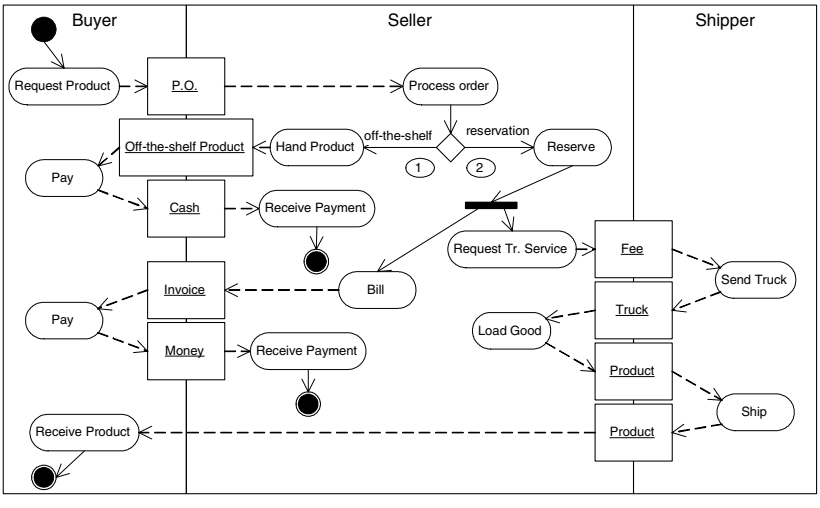

Fig. 2. Process model of the example business case, using UML Activity diagram

2.3 Intuitive Consistency

We consider the presented activity diagram and e^3-*value* model to be consistent because there is at least one system implementing both specifications. An activity diagram and an e^3-*value* model are consistent if (1) for every alternative dependency path in the value model, an execution sequences exists in the process model such that exactly the product value exchanges described by the path are executed and (2) for every execution sequence in the process model, there exist a dependency path in the value model such that it is possible to bind all exchanged products to all product value exchanges.

The following terms need further clarification:

- an alternative dependency path represents a distinctive scenario of value exchanges in an *e³-value* model. A dependency path can include several scenarios, respectively, several alternative dependency paths;
- an execution sequence is a sequence of activities (possibly executed in parallel) in an activity diagram that (1) begins with the start stimulus and ends with termination points and (2) does not contain choices. An activity diagram can include several execution sequences;
- a product value exchange refers to an exchange of good or a service in the *e³-value* model.

3 Consistency

The *e³-value* model and the activity diagram are not directly comparable. The *e³-value* model is based on value exchanges disregarding the order in which they are performed. The activity diagram is based on sequences of object flows disregarding relationships of reciprocal economic value among objects. Thus, we construct a reduced model containing the common concepts and relations of the *e³-value* model and the activity diagram to make the two models comparable.

3.1 Reduced Model

The reduced model is used to compare abstractions of the *e³-value* model and the activity diagram. To avoid confusion of terminology as both notations use the concept *object*, we refer to objects and object flows in the activity diagram as messages and message exchanges, respectively.

In particular, the reduced model used for consistency checking consists of business units, common value objects, and common value exchanges, where:

- a *business unit* (called *unit* for short) corresponds to an actor from the *e³-value* model and a swimlane from the activity diagram. It represents organizational units (grouping of responsibilities) within a business, which are profit and loss responsible but not necessarily legal entities;
- a *common value object* (called *common object* for short) corresponds to a value object from the *e³-value* model and a message from the activity diagram. It represents an object of economic value in the *e³-value*-model sense which is used for coordination of business activities;
- a *common value exchange* (called *common exchange* for short) corresponds to a value exchange in the *e³-value* model and a message exchange in the activity diagram. It represents a bilateral exchange of coordination value objects between profit and loss responsible entities disregarding order, reciprocity and bundling.

Fig. 3 shows the visual notation of the reduced model concepts, where (a) represents a business unit, (b) represents a common value object, and (c) represents a common value exchange.

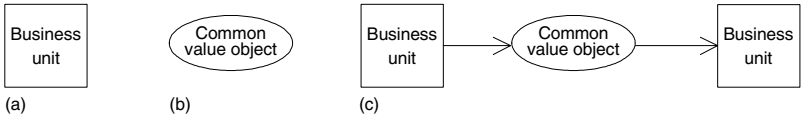

Fig. 3. Visual notation of the reduced model

A reduced model is an explicit representation of a single alternative dependency path in an e^3-*value* model and of a single execution sequence in an activity diagram. It contains the value exchanges executed in one possible scenario in a business case.

Common value exchanges are closely related to value exchanges because they are more generic than message exchanges which explicitly represent message ordering. Reciprocity, as contained in the value model, is not considered in the reduced model because there is no corresponding concept in activity diagrams. Alternatives are represented explicitly because OR-forks in e^3-*value* models and choices in activity diagrams are not always comparable. This approach of comparing alternatives independently of each other is well known, e.g., from deciding properties of workflow models, which is often based on occurrence graphs derived from Petri nets.

Value objects can be divided into three sub-types, namely goods, services and experiences. The term product refers to both goods and services. We require that products are represented in the reduced model with common objects; whereas, we omit experiences from the reduced model as it is unlikely that these are modeled as message exchanges in an activity diagram.

An exchange of a value object in the e^3-*value*-model sense corresponds to a sequence of messages exchanged between two swimlanes in an activity diagram. We will call such a sequence a transaction. Since a sequence of message exchanges does not provide a direction as a value exchange does, we select a single message exchange as a representative of the sequence. The direction of the selected message exchange reflects the direction of the value object exchange. Correspondingly, we map the message of the selected message exchange to a common object.

3.2 Semantic Relationships Between Instances

Besides the conceptual transformation from a value and process model to a reduced model, the instances of the concepts also have to be semantically correlated. The semantic relationship between instances can be one-to-one, one-to-many, and many-to-many. The first two relationship types can be observed in the example described in Section 2 and are covered in this section, while the many-to-many relationship is discussed in Section 4.

The transformation of an e^3-*value* model or an activity diagram results in reduced models. In particular, the reduced models must be based on the same set of semantic instances of units and common objects. The existing semantic relationship between instances of actors and swimlanes is represented by two relationships: between an actor and a unit, and between the same unit and a swimlane. Respectively, the semantic relationship between instances of value objects and messages is represented by two relationships: between a value object and a common object, and between the same common object and a message. To restrict the relationships between actors (and

value objects) and swimlanes (and messages) to one-to-one and one-to-many, we allow a unit (and a common object) to take part in at most a single one-to-many relationship.

The instances of a business unit and a common value object in the reduced model are determined by an expert who has knowledge about the instances of the corresponding concepts in the *e³-value* model and the activity diagram. The expert also determines the mapping between the instances, which is captured in transformation tables.

For the business case in our example (Section 2), the mappings of actors from the *e³-value* model and activity diagram to reduced models are listed in Table 1 (a) and Table 1 (b), respectively. Due to the construction of the example, the rows in Table 1 contain the same actor names representing one-to-one relationships. Table 2 (a) lists the mapping between value objects in the *e³-value* model and the common objects in the reduced model. Again, due to the construction of the example this mapping represents a one-to-one relationship. Table 2 (b) lists the mapping between selected messages in the activity diagram and common objects of the reduced model. The mapping in Table 2 (b) contains two one-to-many relationships; i.e., the `Money` and `Cash` messages of the activity diagram map to the `Money` common object of the reduced model, and the `Product` and `Off-the-self product` messages of the activity diagram map to the `Product` common object of the reduced model.

In the following, we describe the transformation of an arbitrary *e³-value* model and activity diagram to their underlying reduced models by means of the example above.

Table 1. Mapping of: (a) actors of the *e³-value* model of Fig. 1 to business units of the reduced model (b) business units of the reduced model to swimlanes of the activity diagram of Fig. 2

(a)

e³-value model	Reduced model
`Buyer`	`Buyer`
`Seller`	`Seller`
`Shipper`	`Shipper`

(b)

Reduced model	Activity diagram
`Buyer`	`Buyer`
`Seller`	`Seller`
`Shipper`	`Shipper`

Table 2. Mapping of: (a) value objects of the *e³-value* model of Fig. 1 to common value objects of the reduced model (b) common value objects of the reduced model to messages of the activity diagram of Fig. 2

(a)

e³-value model	Reduced model
`Money`	`Money`
`Product`	`Product`
`Fee`	`Fee`
`Transport`	`Transport`

(b)

Reduced model	Activity diagram
`Money`	`Money`
	`Cash`
`Product`	`Product`
	`Off-the-shelf product`
`Fee`	`Fee`

Further, a notion of consistency based on equivalence of reduced models will be introduced.

3.3 Transformation from an e^3-*value* Model to Reduced Models

This section describes a transformation from an e^3-*value* model to reduced models. The transformation has three steps. The first step separates possible alternatives in a scenario. The second step selects the actors and value objects to be represented in the reduced model as units and common objects, and builds the transformation tables. The third step transforms actors and value objects to units and common objects.

Step 1: Separate alternatives. This step deals with OR-forks in the dependency path in e^3-*value* models. When we encounter an OR-fork, we duplicate the e^3-*value* model in accordance to the number of alternatives in the OR-fork. In each copy of the original model, we substitute the OR-fork with a dependency connection to form a single alternative. The other alternatives remain disconnected. This transformation step generates from a single e^3-*value* model potentially many e^3-*value* models.

The OR-forks are treated, beginning from the start stimuli, consequently in the order of their occurrence along the dependency path. This guarantees that all possible scenarios of value exchanges are captured in individual reduced models. At the end of the step, the exchanges that are not connected in a dependency path are removed from the models. Then, the dependency paths are also removed. The final result of step 1 of the transformation is shown in Fig. 4.

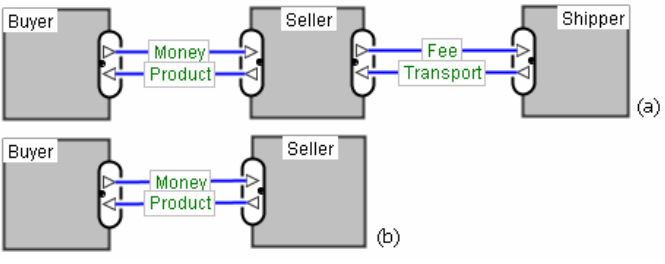

Fig. 4. Final result of transformation step 1

Step 2: Transformation tables. This step classifies the value objects into product and experience types. Product type value objects are entitled for transformation, where the experience type value objects are removed from the e^3-*value* model. As a result, actors that exchange only experience type of value objects are isolated and are, therefore, also removed from the model. In the e^3-*value* model of our example, all value objects are eligible for translation.

Remaining actors and product type value objects are mapped to business units and common value objects, respectively. With regard to our example, the mappings are represented in the transformation tables Table 1 (a) and Table 2 (a).

Step 3: Generate reduced models. This step transforms each e^3-*value* model representing an alternative into a reduced model. In particular, actors and value

objects are transformed into business units and common value objects as specified in the mapping tables (see Table 1 (a) and Table 2 (a)). As a result of this transformation the specific information on reciprocity of value exchanges and on bundling of value objects is omitted. The reduced models, derived form the *e³-value* model (see Fig. 1), are depicted in Fig. 5.

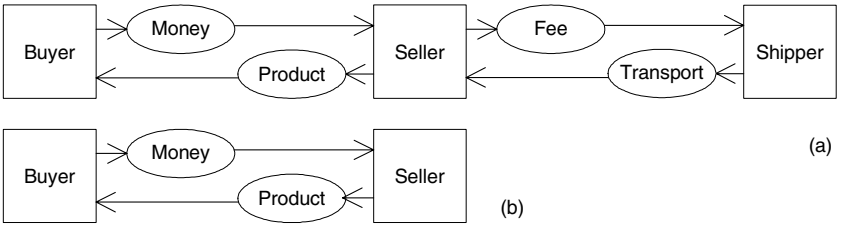

Fig. 5. Reduced models corresponding to the *e³-value* model

3.4 Transformation from an Activity Diagram to Reduced Models

The transformation of an activity diagram to reduced models is performed in three steps. The first step resolves choices in the control flow. The second step identifies sequences of messages, marks single messages as corresponding to value exchanges in the *e³-value*-model sense, and builds transformation tables. The third step transforms swimlanes to units and messages to common objects.

Step 1: Remove choices. This step transforms the activity diagram to a number of models which do not contain choices; i.e. the resulting models do not have conditional branches of execution flow. The transformation works in a similar way as the transformation of OR-forks in the *e³-value* model. We begin from the start stimuli and

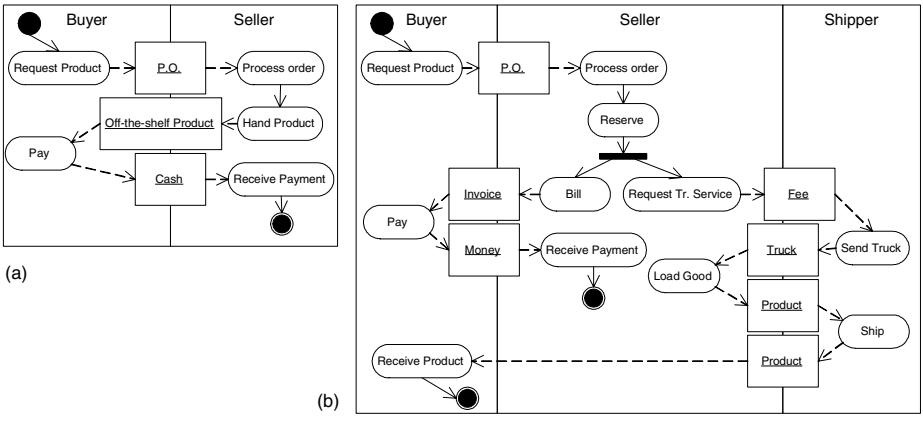

Fig. 6. Result of transformation step 1

we follow the execution flow. Each time we encounter a choice, we duplicate the model and substitute the choice with a direct transition to one of the alternatives. We cut the disconnected branches from the execution tree. This transformation step generates potentially many activity diagrams from a single activity diagram. Fig. 6 shows the result of transformation step 1 applied on the activity diagram from Fig. 2.

Step 2: Transformation tables. This step identifies the flow of messages between two swimlanes that result in an exchange of a value object in the e^3-*value*-model sense. Additionally in each sequence, a single message is selected to be further transformed to a common object.

The selected messages and their sending and receiving swimlanes are mapped to common value objects and business units, respectively. The mappings are represented in transformation tables which for our example are Table 1 (b) and Table 2 (b).

Step 3: Generate reduced models. This step transforms each activity diagram representing an alternative into a reduced model. In particular, swimlanes and messages are transformed into business units and common value objects as specified in the mapping tables Table 1 (b) and Table 2 (b). As a result of this transformation the specific information on sequence of message exchanges is omitted. The final reduced models, derived form the activity diagram (see Fig. 2), are depicted in Fig. 7.

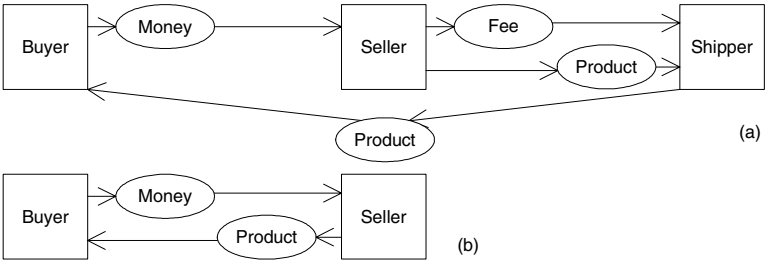

Fig. 7. Reduced models corresponding to the activity diagram

For clarity of presentation, we refer further to a reduced model derived from an activity diagram as a reduced process model. Respectively, a reduced model derived from a value model is referred to as a reduced value model. The origin of a reduced model is undistinguishable from the model itself; we name them differently for explanation purposes only.

3.5 Equivalent Reduced Models

Two reduced models are equivalent if each contains the same common value exchanges. This means that:

- each reduced model contains the same business units;
- each reduced model contains the same common value objects;
- in each reduced model, the sending and receiving business units of a particular common value object are the same.

In our example, we can determine that the reduced models of Fig. 5(b) and Fig. 7(b) are equivalent. In contrast, the models of Fig. 5(a) and Fig. 7(a) are not because (i) the `Transport` common object is not present in the reduced process model and (ii) the `Product` common object is exchanged between different units in the two models.

3.6 Transitivity

The reduced models in **Fig. 5**(a) and **Fig. 7**(a) can be made equivalent by applying transitivity on the `Product` common object in the reduced process model (see Fig. 7(a)). Transitivity removes intermediary units from a chain of common exchanges by directly representing the common exchange between the units in the beginning and the end of the chain. The reason for the unit in the beginning of the chain to involve additional units must be beneficial to the unit itself. Thus, the benefit must be provided by the intermediary unit because otherwise it would not have been involved.

As a consequence, the `Product` common exchanges between `Seller` and `Shipper`, and between `Shipper` and `Buyer` (see Fig. 7(a)) is represented as a direct `Product` common exchange between `Seller` and `Buyer`. The benefit introduced by the intermediary unit, i.e. `Shipper`, to `Seller` is represented by an unspecified common exchange. The unspecified common exchange can be instantiated by any reduced model common object. This introduces several additional options to be checked by the equivalence testing. With regard to the example, the instantiation with the `Transport` common object, as depicted in Fig. 8, results in equivalent reduced models.

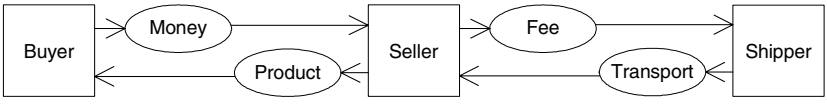

Fig. 8. Reduced model after transitivity transformation

Due to the equivalence of the reduced models, we consider the *e*^{3}*-value* model and the activity diagram to be consistent, as we intuitively do in Section 2.3.

3.7 Consistency

An *e*^{3}*-value* model and an activity diagram are consistent if there exists at least one non-trivial mapping under which the corresponding sets of reduced models are equivalent.

A non-trivial mapping is one for which holds that:

(1) every product value exchange in the *e*^{3}*-value* model is mapped to one common value exchange. This includes that (i) every product value object is represented in the reduced model and (ii) sending and receiving actors of a product value object are not mapped to a single business unit in the reduced model;

(2) every transaction in the activity diagram is mapped to one common value exchange. This includes that (i) every transaction is represented in the reduced model

and (ii) sending and receiving swimlane of the message representing the transaction are not mapped to a single business unit in the reduced model.

The restrictions listed above preserve the product value exchanges in the e^3-*value* model and the transactions in the activity diagram during the transformation; i.e., these are all represented in the reduced model. In case the granularity of the e^3-*value* model and the activity diagram is similar, the relationships between actors and swimlanes, and between value objects and messages are usually one-to-one or one-to-many. Nevertheless, the relationships are many-to-many in the general case. The consequences of which, we discuss in the next section.

4 Granularity of Models: Many-to-Many Relationships Between Instances

In the previous section, we show how our consistency check works between models with comparable granularity. In particular, the relationships between actors and swimlanes, and value objects and messages are only one-to-one and one-to-many; where, the latter one breaks down to one-to-one relationships when the alternatives are taken separately. However, there is no guarantee that the granularity of e^3-*value* models and activity diagrams is not the same. Therefore, the relationships between actors and swimlanes, and value objects and messages are in the general case many-to-many. In this section, we discuss when and, if so, how consistency of models with different granularity can be checked.

Our consistency definition is based on equivalence of reduced models, which requires equivalence of units and common objects. From (i) the equivalence of units, respectively common objects, and (ii) the way the mapping tables are constructed (see Section 3.2) follows that the semantic relationships between instances in the e^3-*value* model and the activity diagram are one-to-one. To guarantee a proper result of our consistency check, we have to ensure that the checked models have similar granularity.

There are two strategies for adapting granularity: either aggregation is performed on the more fine-grained model or division of the more coarse-grained model. We have explored both approaches and discovered the following drawbacks.

The *aggregation approach* may lead to a single actor and a single swimlane. This is because, a many-to-many relationship between actors and swimlanes can result in an aggregation of two swimlanes which may trigger an aggregation of two actors and so forth. Due to the aggregation, the exchanges of product value objects between aggregated actors in the e^3-*value* model are lost; the same holds for messages between swimlanes in the activity diagram. This loss of information makes the consistency decision less precise.

The *division approach*, on the other hand, may lead to the finest-grained granularity in both models, where a single actor exchanges a single value object or a single swimlane sends a single message. This is again a loss of information comparable to the aggregation case. In particular, the relation information between different value objects, respectively messages, is lost. Thus to limit the loss of information, we can constrain ourselves to division of only value objects and swimlanes or only actors and messages. From the two, we select the second because

actors are intuitively more coarse-grained than swimlanes and a single message may represent more than a single economic value. From a preliminary evaluation, this option is the most promising one, although, a more detailed evaluation has to be done.

Below, we illustrate how granularity of models is equalized by division of actors and messages. On behalf of an example, we show how we resolve one-to-many relationships.

4.1 Example

We consider a business case where a client is interested in the mortgage and insurance products of a bank. Fig. 9 represents the *e*^{3}*-value* model of the example. The client (to the left in the figure) is interested in exchanging monthly fees for a period of time in case it gets a loan in a form of a mortgage. Additionally to that for some economic reasons, the client is interested in insurance from the same bank. The dependency path includes an AND-fork, shown within the `Client` actor, which denotes that the client wants both products.

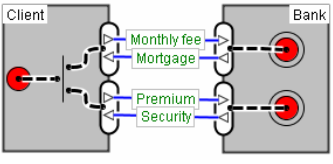

Fig. 9. Value model of the example

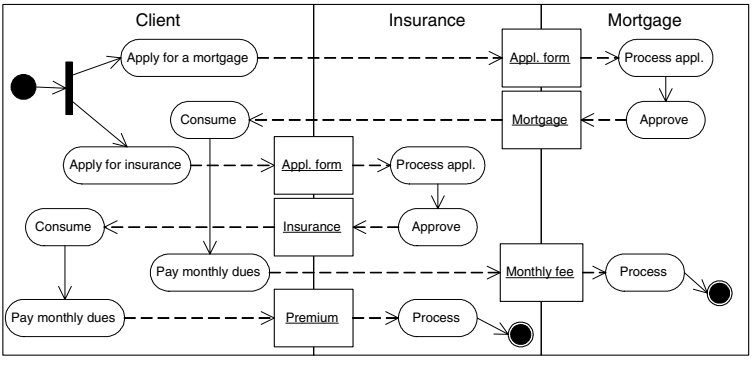

Fig. 10. Process model of the example with two swimlanes representing the bank

Fig. 10 represents an activity diagram for the above example case, where the bank is modeled as two swimlanes representing its Mortgage and Insurance departments separately. The client (the first swimlane from left to right) requests simultaneously a mortgage and insurance. The handling of the requests is performed in a similar way: e.g., the mortgage request (in the top of the figure) is processed by the `Mortgage`

swimlane which after processing the request grants mortgage to the client. Once given, the insurance and the mortgage are utilized by the client, which is shown in the client swimlane as a `Consume` activity. The bottom of Fig. 10 shows the monthly payments performed by the client: two separate payments to the `Mortgage` and `Insurance` departments are represented by the `Monthly fee` and `Premium` messages.

4.2 Splitting of Actors

The models in Fig. 9 and Fig. 10 differ in granularity: the bank from the e^3-*value* model is represented as two individual swimlanes in the activity diagram. To resolve the one-to-two relationship between actor and swimlanes, we split the bank actor in the e^3-*value* model. The newly appeared actors, named `BankM` and `BankI`, need to distribute the value exchanges of the original actor `Bank`. In our example case, there are four exchanges and we generate all possible combination with the new actors. We check consistency with all combinations.

The choice of splitting the `Bank` actor is derived from the mapping tables, where we observe the one-to-many relationship. We split an actor by splitting the corresponding unit in the reduced model.

Based on the splitting of the units in the value reduced model it turns out that the reduced models are equivalent, which fits to the intuitive consistency.

4.3 Splitting of Messages

Fig. 11 shows a second activity diagram for the example case. The diagram differs in two points from Fig. 10. First, the bank is represented as one swimlane and correspondingly the activities and messages belonging to the `Mortgage` and `Insurance` swimlane are in the `Bank` swimlane. The second difference is in the way payment of monthly dues is modeled. In the bottom of Fig. 11, payment by the client is represented as a single message.

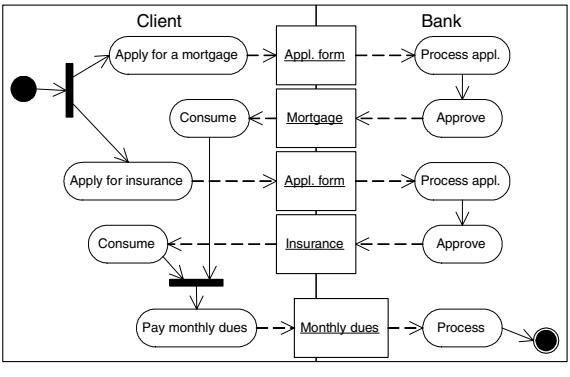

Fig. 11. Process model of the example with one message representing the monthly dues

The models in Fig. 9 and Fig. 11 differ in granularity: the monthly payments are represented as two distinctive value objects in the *e³-value* model while in the activity diagram they are modeled as one message. To resolve the two-to-one relationship between value objects and a message, we split the `Monthly dues` message into `IMonthly dues` and `MMonthly dues`. The new messages share the same sender and receiver as the original message.

The choice of splitting the `Monthly dues` message is derived from the mapping tables, where we observe the one-to-many relationship. We split a message by splitting the corresponding common object in the reduced model.

Based on the splitting of the common objects in the process reduced model it turns out that the reduced models are equivalent, which fits to the intuitive consistency.

4.4 Implications

We have analyzed a number of examples to justify our approach of splitting actors and messages. We classify them based on:

- Cardinality of the relationship, where we consider one-to-many and many-to-many relationships;
- Direction of the relationship, where we consider one instance in the *e³-value* model related to many instances in the activity diagram and vice versa;
- Type of the relationship, where we specialize relationships into *individual* and *aggregation*. An individual one-to-many relationship means that one entity is mapped to several independent entities each of which represents the entity as a whole. (This is the type of relationship we have in our first example in Section 3.) An aggregation one-to-many relationship means that one entity is mapped to several independent entities which together represent the entity as a whole. (This is type of relationship we have in our second example in Section 4);
- Arguments of the relationship, where we considered relationships between actors and swimlanes, and value objects and messages.

Our analysis of all combinations of the classifications criteria above shows that it is possible to adapt the granularity of models applying the approach of splitting actors and messages. It is possible to reduce the one-to-many and many-to-many relationships to one-to-one relationships in all cases except one where we have one-to-many aggregation type relationship between a swimlane and actors. Although such a relationship is possible, we think it is rarely used; intuitively, an *e³-value* model is at a higher level of granularity than an activity diagram.

5 Validation

The proposed consistency check is valid with respect to the intuitive consistency definition if all model pairs considered to be intuitively consistent are consistent with regard to our consistency definition and vice versa. To argue that this is the case, we will decompose the intuitive consistency definition from Section 2.3 and compare it with the building blocks of our consistency definition.

For the intuitive consistency, we make the following observations:

1. It is based on relations between separate alternative dependency paths and separate execution sequences;
2. The relation between an alternative dependency path and an execution sequence is based on a single set of product value exchanges happening in both models.

Our transformation procedures represent the original model as several reduced models, one per alternative, which is based on alternative dependency paths and execution sequences. That is, one alternative dependency path (execution sequence) results in a single reduced model. Thus, the granularity of the performed consistency check is the same as in the intuitive one.

The second observation says that an alternative dependency path and an execution sequence result in the same product value exchanges. Our definition of equivalent reduced models requires identical common value exchanges in the two models. This shows that both consistency definitions require a relationship between models based on *the same set of* product value exchanges and *on the same set of* common value exchanges.

As we describe in Section 3 the relationship between value exchanges and common exchanges is one where every product value exchange is represented in the reduced model. Similarly, transactions in the activity diagram are identified as such if they result in a product value exchange. Thus in case of a non-trivial mapping, every product exchange is transformed to a common exchange.

We conclude that the proposed consistency definition is valid with respect to the intuitive consistency definition.

6 Consistency with the Application Communication Perspective

Throughout this paper, we discuss the economic value and business processes perspectives and their consistency relationship (see Section 1). The two perspectives focus on the value and control flow aspects among businesses. Our third perspective, i.e. the application communication perspective, models the data flow without explicating alternatives in a similar sense to alternatives in the value and process models. The differentiation between data-flow alternatives is based on knowledge gained from the development process of the communication model. Thus, each alternative in the application communication perspective is modeled separately, represented as a partial communication model.

From a preliminary investigation, we can state that the consistency check approach based on alternatives can be applied on the remaining two consistency relationships between a value model and a communication model, and between a process model and a communication model. However, a more detailed analysis will be performed in future work.

We select the consistency relationship between the e^3-*value* model and the activity diagram as we consider it the most difficult one. An e^3-*value* model includes several alternatives, which represent several possible scenarios of value exchanges. Respectively, an activity diagram includes choices which result in several possible execution sequences. In comparison with the application communication perspective,

the additional steps to separate the alternatives in an *e³-value* model and an activity diagram make the consistency checking more complex.

7 Related Work

Consistency can be checked in various ways. The approach with *syntactic translation* (also called direct translation) [2] is based on directly relating terms of two notations. Then, one specification is translated to the modeling language of the other. The *common semantic model* (also called canonical representation) approach [3] selects a single modeling notation (not necessarily one already in use) and transform all models to that notation. The *meta-representation* approach [11, 12] does not require transformation between models. It specifies relations between meta-modeling and modeling concepts from each modeling notation. These relations must hold between the concepts and their instances in each model.

Our approach is based on the common semantic model approach. We define a common semantic model, which we call reduced model, in a pair-wise fashion. This gives us richer reduced models compared to a single reduced model for all perspectives. Additionally, our approach introduces a consistency check based on alternatives; i.e. models are decomposed into smaller models and checked individually for consistency. This provides a consistency check that matches with the intuitive consistency definition.

Our work is an extension of Gordijn's [8] requirement engineering approach to innovative e-commerce ideas. He specifies a method for exploration of business opportunities based on the distribution of value in business networks. Additionally to the value viewpoint (viewpoint is a synonym for perspective), the approach includes two more: a business process and an information systems viewpoints. The three viewpoints match closely with our perspectives. However, we explicitly check for consistency as we assume independent development of models; whereas Gordijn's approach is based on a common set of scenarios represented in each model.

Our approach requires a semantic mapping between concepts in the value and process models. The work of Gordijn, Akkermans and Vliet [6] elaborates on the differences between business and process modeling by showing semantic differences between concepts. We use this information to define our transformation tables. While Gordijn, Akkermans, and Vliet specify differences between concepts of the different models. The proposed approach specifies semantic relationships between instances.

Wieringa and Gordijn [13] define a correctness relationship between an *e³-value* model and a process model. We use this to define our intuitive consistency. In addition, we provide an operationalization of this intuitive consistency definition based on transformations to reduced models.

The work of Dijkman et al. [4] is also based on the common semantic model approach. It relates viewpoints (viewpoint is a synonym for perspective) by means of a basic viewpoint which contains pre-defined concepts and relations. Every viewpoint from a design framework need to be mapped to basic concepts and relations from the basic viewpoint. Our approach differs in the way how the reduced model (the basic viewpoint in Dijkman et al.'s terms) is defined. We do not require a pre-defined reduced model with abstract basic constructs, but we determine the reduced model

after the modeling notations are selected. As we pointed in Section 0, this allows defining richer reduced models in terms of common concepts and relations.

Consistency of a workflow model can usually be defined based on the set of potential execution sequences, a straight forward approach to check consistency is on a single workflow model. This approach has been applied to several workflow models, like for example by v.d.Aalst and Weske [1] to Workflow Nets (WF-Nets), by Fu et.al. to guarded Finite State Automata [5], by Yi and Kochut [17] to Coloured Place/Transition Nets, or by Wodtke and Weikum [15] to statecharts. In either case it is checked whether the execution of the workflow results in a deadlock, that is, no further action is possible although a final state has not been reached yet. However, there exists also approaches on checking consistency between several workflows represented in the same modeling approach, like e.g. [1, 16, 9]. In our paper, consistency between different modeling approaches is defined.

8 Conclusions and Future Work

The contribution of this paper is a definition of consistency between an e^3-*value* model and an activity diagram. We operationalize the consistency check by defining a reduced model that contains the common concepts from two models. Further with the help of mapping tables, we transform the e^3-*value* model and the activity diagram to reduced models. Finally, we check equivalence of reduced models. We argue that the consistency definition is valid with respect to the intuitive consistency definition.

The e^3-*value* model and the activity diagram represent two of the three perspectives of a development framework used to align information system capabilities and business objectives [18]. The third perspective represents the communication among components of the business information systems. Future work includes the definition and operationalization of the remaining two consistency relationships: between the e^3-*value* model and the communication perspective model, and between the activity diagram and the communication perspective model. Further, we aim to investigate the usefulness of the three binary consistency definitions as necessary conditions for a global consistency. Finally, an implementation has to be provided to automate the consistency check of the different perspectives as more realistic and complex examples are likely to contain high numbers of reduced models.

Acknowledgements

We thank Roel Wieringa and Maya Daneva for their valuable comments on the draft versions of this paper.

References

1. Aalst van der, W.M.P., Weske, M.: The P2P approach to Interorganizational Workflows. Proceedings of 13th International Conference on Advanced Information Systems Engeneering (CAISE). Interlaken, Switzerland (2001)
2. Bowman, H., Steen, M.W.A., Boiten, E.A., Derrick, J.: A formal framework for viewpoint consistency. Formal Methods in System Design, 21, September (2002) 111—166

3. Braatz, B., Klein, M., Schröter, G.: Semantical Integration of Object-Oriented Viewpoint Specification Techniques. In Integration of Software Specification Techniques for Applications in Engineering, Lecture Notes in Computer Science, Springer. (2004)
4. Dijkman, R.M., Quartel, D.A.C., Pires, L.F., Sinderen van M.J.: An Approach to Relate Viewpoints and Modeling Languages. In: Proceedings of the 7th IEEE Enterprise Distributed Object Computing (EDOC) Conference, Brisbane, Australia (2003) 14—27
5. Fu, X., Bultan, T., Su, J.: Realizability of Conversation Protocols with Message Contents. Proceedings IEEE International Conference on Web Services (ICWS) (2004) 96—103
6. Gordijn, J., Akkermans, J.M., Vliet van, J.C.: Business Modelling is not Process Modelling. In: Conceptual Modeling for E-Business and the Web, LNCS 1921. Salt Lake City, USA, October 9-12 (2000) 40—51
7. Gordijn, J., Akkermans, J.M.: Value based requirements engineering: exploring innovative e-commerce idea. Requirements Engineering Journal, 8, 2 (2003) 114—134.
8. Gordijn, J.: Value based requirements engineering: Exploring innovative e-commerce ideas. PhD. thesis, Vrije Universiteit Amsterdam (2002)
9. Kindler, E., Martens, A., Reisig, W.: Inter-operability of Workflow Applications: Local Criteria for Global Soundness. Business Process Management, Models, Techniques, and Empirical Studies (2000) 235—253
10. OMG: OMG UML Specification (2003) http://www.omg.org/cgi-bin/doc?formal/03-03-01
11. Sunetnanta, T., Finkelsteing, A.: Automated Consistency Checking for Multiperspective Software Specifications. Workshop on Advanced Separation of Concerns. Toronto (2001)
12. Sunetnanta, T.: Multiperspective Development Environment for Configurable Distributed Applications. Ph.D. Thesis. Department of Computing, Imperial College. February (1999)
13. Wieringa, R.J., Gordijn, J.: Value-Oriented Design of Service Coordination Processes: Correctness and Trust. ACM Symposium on Applied Computing (2005)
14. Wieringa, R.J.: Design Methods for Reactive Systems. Morgan Kaufmann (2002)
15. Wodtke, D., Weikum, G.: A Formal Foundation for Distributed Workflow Execution Based on State Charts. In: Afrati, F.N., Kolaitis, P. (eds.): Proceedings of the 6th International Conference on Database Theory (ICDT) (1997) 230—246
16. Wombacher, A., Fankhauser, P., Aberer, K.: Overview on Decentralized Establishment of Consistent Multi-Lateral Collaborations Based on Asynchronous Communication. Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE) (2005) 164—170
17. Yi, X., Kochut, K.J.: Process Composition of Web Services with Complex Conversation Protocols: a Colored Petri Nets Based Approach. Proceedings of the Design, Analysis, and Simulation of Distributed Systems (2004) 141—148
18. Zlatev, Z., Daneva, M., Wieringa, R.J.: Multi-Perspective Requirements Engineering for Networked Business Systems: A Framework for Pattern Composition. 8th Workshop on Requirements Engineering. Porto, 13-14 June (2005)