Value-Oriented Design of Service Coordination Processes: Correctness and Trust

Roel J. Wieringa[*]
Department of Computer Science
University of Twente
P.O. Box 217 tel. +31 53 489 4189, Fax +31 53 489 2927
7500 AE Enschede, The Netherlands
roelw@cs.utwente.nl

Jaap Gordijn
Faculty of Sciences
Vrije Universiteit
De Boelelaan 1081a
1081 HV Amsterdam, The Netherlands
gordijn@cs.vu.nl

ABSTRACT

The rapid growth of service coordination languages creates a need for methodological support for coordination design. Coordination design differs from workflow design because a coordination process connects different businesses that can each make design decisions independently from the others, and no business is interested in supporting the business processes of others. In multi-business cooperative design, design decisions are only supported by all businesses if they contribute to the profitability of each participating business. So in order to make coordination design decisions supported by all participating businesses, requirements for a coordination process should be derived from the business model that makes the coordination profitable for each participating business. We claim that this business model is essentially a model of intended value exchanges. We model the intended value exchanges of a business model as e^3*-value* value models and coordination processes as UML activity diagrams. The contribution of the paper is then to propose and discuss a criterion according to which a service coordination process must be correct with respect to a value exchange model. This correctness is necessary to gain business support for the process. Finally, we discuss methodological consequences of this approach for service coordination process design.

General Terms

Business process modeling and specification, Requirements for service-oriented processes, E-Business

Keywords

Value modeling, Service coordination, Correctness, Trust

[*]This work is partially supported by the BSIK-funded project Freeband/AMUSE.

SAC'05, March 13-17, 2005, Santa Fe, New Mexico, USA.

1. INTRODUCTION

Whenever one business requests more than one service from another business, a process is needed to coordinate the service provisioning across the different companies [2]. A service coordination process is a specification of a cross-organizational coordination process. Several languages for service coordination have been proposed, including WS-coordination [8], BPSS (ebXML's business process specification schema) [10], WSCI (Web Service Choreography Interface) [4] and BPEL4WS [3]. These languages can be used to specify a coordination infrastructure or to specify the coordination process to be used by partners. There is currently a lack of guidelines to actually *do* the specifying. In this paper, we investigate the problem of designing service coordination processes.

This is different from the problem of designing an intra-business workflow. Even though many issues in the design of coordination specification languages are the same as issues in the design of workflow specification languages [1], the *process* of specifying inter-organizational coordination processes is quite different from the *process* of specifying intra-organizational workflows. Where intra-organizational workflow design takes place in a single decision-making hierarchy, interorganizational coordination process design takes place in a network with multiple decision centers. Each party in a coordination is an independent business that may decide to leave the network, may decide to behave differently from what is agreed, and wants to keep confidential some of its own business rules and processes. This makes the design of interorganizational coordination both different and more difficult than workflow design.

We consider the problem of designing static coordination processes, i.e. processes that are agreed upon in advance among a set of business partners and then are executed as specified. An example is the trade procedure for international container trade [6], used as an example in this paper.

We approach this problem by observing that when two or more businesses coordinate their activities, presumably they do this because it is beneficial for each of them. We assume that this is the reason that each of the businesses decided to join the cooperation—in the absence of a hierarchical decision structure, each business will decide whether the cooperation is profitable, or at least beneficial for its own interests in some way. A coordination designer must therefore first understand this business model, so that he or she can define coordination actions that facilitate it. Note that correctness of a coordination process with respect to a business model is not a guarantee that the benefits for each partner will actually materialize by the cooperation. Rather, correctness should entail that the business transactions deemed to be beneficial according to the business model, can be executed according to the process. The precise notion of correctness involed will be detailed later in this paper. Along the way, it will turn out that the correctness of a coordination process with respect to a business model must be supplemented with the correctness of trust assumptions that are made by the business model and coordination process. The less the business partners trust each other, the more complex the coordination process and underlying business model, and vice versa. In sum, this paper argues for the following claims.

- A coordination process design must be based on a business model in which two or more business actors decide to cooperate.
- This business model is a model of profitable value exchanges among businesses.
- A coordination process must be proven to be correct with respect to the underlying business model.
- A coordination process comes with trust assumptions that determine its scope of application.

In section 2 we explain how we can specify *business models* in which two or more business partners can engage in a number of business transactions in which they exchange objects of economic value. In section 3 we then show how we can design a coordination process based on a business model. In section 4 we show how this can be shown to be correct with respect to the business model. Sections 5 and 6 refine the notion of correctness by introducing the notion of trust. Section 7 draws methodological lessons from this, and section 8 concludes the paper with a discussion of further work.

2. VALUE MODELS

Since we claim that a coordination process should be correct with respect to a business model, we must show what a business model looks like. We consider the core of a business model to be a model of value exchanges, showing which businesses perform valuable services for which other businesses. To represent this economic value aspect of a business model, we use the e^3*-value* method [14] to describe the value exchanges in the business model. We illustrate e^3*-value* with a simple business model: a shipper ships goods through a carrier to a consignee (see Fig. 1). The value model of Fig. 1 shows the economic value aspect of this business model. It shows that in return for delivering goods, the consignee pays the shipper, and in return for providing a transport service, the shipper pays the carrier. More in detail, an e^3*-value* model contains the following concepts.

Actor. An actor is perceived by its environment as an independent economic (and often also legal) entity. An actor intends to make a profit or to provide a non-profit service. In a sound, sustainable, business model each actor should be capable of creating a net value. Commercial actors should be able to make a profit, and non-profit actors should be able to create a value that in monetary terms exceeds the costs of producing it in order to sustain. In Fig. 1 three actors are show: a consignee, a shipper and a carrier.

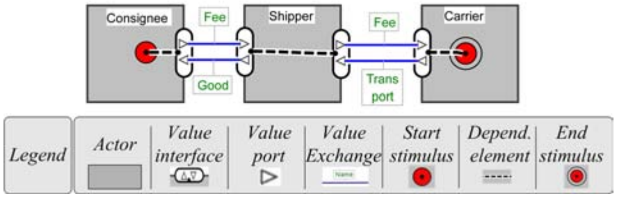

Figure 1: A simple value model of a business transaction. The legend is not part of the notation.

Value Object. Actors exchange value objects, which are services, products, money, or even consumer experiences. A value object is of value to at least one actor. In Fig. 1 Good and Fee are value objects, as well as Transport. Specifically, we view *services* as value objects that can be delivered to actors.

Value Port. An actor uses a value port to show to its environment that it wants to provide or request value objects. A value port has a direction, namely outbound (e.g. a service provision) or inbound (e.g. a service consumption). A value port is represented by a small arrowhead that represents its direction.

Value Interface. A value interface consists of ingoing and outgoing ports of an actor. Grouping of ingoing and outgoing ports model economic reciprocity: an object is delivered via a port, and another object is expected in return. An actor has one or more value interfaces, each modelling different objects offered and reciprocal objects requested in return. The exchange of value objects across one value interface is atomic: either all objects are exchanged, or none at all. A value interface is represented by an ellipsed rectangle.

Value Transfer. A value transfer connects two value ports of opposite directions of different actors with each other. It is one or more potential trades of value objects between these value ports. A value transfer is represented by a line connecting two value ports. In e^3*-value* , value transfers model *service deliveries.*

Value Transaction. Value transfers come in economic reciprocal pairs, which are called value transactions. This models 'one good turn deserves another': you offer something to someone else only if you get adequate compensation for it. So, Fig. 1 contains two value transactions, each consisting of the two exchanges, namely Good/Fee, and Transport/Fee.

Note that a value model describes an ideal world: In the actual world, a consignee may not pay for the goods, a shipper may sell forged goods, a carrier may loose the goods, etc. This is not modeled in a business model. A business model represents the ideal state of affairs in which actors make money by engaging in certain business transactions. The question whether this model holds up against all contingencies of the real world, and which safeguards must be built in to avoid fraud, is one to be answered by the coordinatioon process that implements the business model. This is related to the fact that a business model does not represent a process; it merely says that certain business transactions take place. We return to this point in a moment.

With the concepts introduced so far, we can describe who exchanges values with whom. However, one of the uses of a value model is to assess the net cash flow of each business actor as the result of a consumer need (see [14] for an elaborated example). The whole network of business actors exists to satisfy a consumer need (the need of the consignee in Fig. 1). To assess the net cash flow generated by the occurrence of such a need, we must *count* the number of value transfers triggered by one consumer need. To do this, we include in the business value model a representation of *dependency paths* between value interfaces. A dependency path connects value interfaces in an actor, meaning that if one of these interfaces is triggered, the other ones connected to it must be triggered as well. We need to know this for a profitability computation because if value crosses one interface, values cross the interfaces dependent on it as well.

A dependency path consists of dependency nodes and connections.

Dependency node. A dependency node is a stimulus (represented by a bullet), an AND-fork or AND-join (short line), an OR-fork or OR-join (triangle), or an end node (bull's eye). A stimulus represents a consumer need and can be seen as a trigger for the exchange of economic value objects. An end node represents a model boundary (for instance, in Fig. 1 we do not consider exchanges the carrier has to do to deliver transport).

Dependency connection. A dependency connection connects dependency nodes and value interfaces. It is represented by a dashed line.

Dependency path. A dependency path is a set of connected dependency nodes and connections, that leads from one value interface to other value interfaces or end nodes of the same actor. The meaning of the path is that if a values are exchanged at value interface I, then value interfaces pointed to by the path that starts at interface I are triggered according to the and/or logic of the dependency path. If a branch of the path points to an end node, then no more exchanges are triggered.

Dependency paths allow one to reason about a network as follows: When an end consumer generates a stimulus, this triggers a number of value interfaces of the consumer as indicated by the dependency path starting from the triggering bullet inside the consumer. These value interfaces are connected to value interfaces of other actors by value transfers, and so these other value interfaces are triggered too. This in turn triggers more value interfaces as indicated by dependency paths inside those actors, and so on. Following the value transfers and dependency paths, and estimating the value of each value transfer for the involved actors, we can compute the net incoming and outgoing cash flows for each actor triggered by a consumer need.

Note that an e^3*-value* model should not be seen as a process model [15]. A value model does not represent behavior but only shows objects that are of *economic value* for someone. We will see that a process model expressing coordination between the actors, shows many objects that are not of direct value, but are needed to control the process flow or synchronize activities in different actors. Fig. 1 does not show control flow and there is a variety of coordination processes that can implement the desired value transfers. Additionally, business coordination processes do not have the notion of economic reciprocity, which is in e^3*-value* captured by the 'value interface' concept. If we confuse decisions about the exchange of economic values with decisions about the coordination processes that put such exchanges into operation, we confuse exchanges essential for the businesses with activities that might be changed without changing the business model.

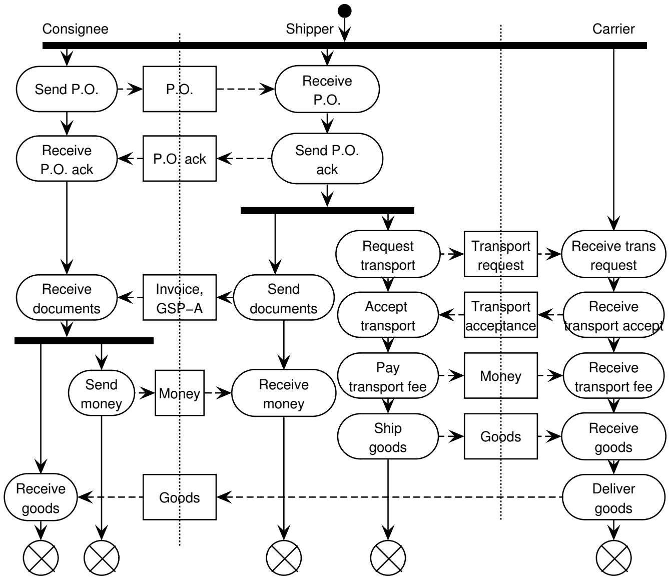

Figure 2: A simple coordination process that correctly implements the value model of 1.

3. COORDINATION PROCESSES

The business model in Fig. 1 assumes an ideal world: In response to a consumer need of the consignee, the consignee and shipper will exchange both a good *and* a fee, and the shipper and carrier will exchange a transport service and fee, or none of this will happen. It is for instance in Fig. 1 not possible that a consignee obtains a good without paying for it.

To implement the business value model of Fig. 1 in a coordination process, we need to make the basic choice who takes the risk of not getting anything in return for a delivered service. For example, if the consignee pays before the goods are shipped, the shipper is sure of getting his money but the consignee takes the risk of not getting the goods. If the shipper ships first, the risks are reversed. This is a business decision that depends upon how much trust each actor places in the other. If all actors trust each other completely, it does not matter who takes the risk because the other party can be trusted to honour his part of the deal. Let us first assume all three parties in the model can be trusted completely and ask what a correct coordination process is.

We use simple activity diagrams as notation for coordination processes, using the UML 2.0 notation (see `www.uml.org`). This choice is reasonable because UML activity diagrams are used in at least two coordination process standards, namely BPSS [10] and RosettaNet [18]. However, our approach does not depend on the choice of UML activity diagrams and is valid for other process notations too that can express object flows. For example, Bons et al. [6] use an extension of Petri Nets called Documentary Petri Nets.

Fig. 2 shows a simple coordination process for the simple business transaction of Fig. 1. Ovals represent activities, rectangles represent objects (data, material, or money), unbroken arrows represent control flow and dashed arrows represend object flow. In our diagrams, objects can contain data, goods, services, or money. Control flow can be structured using solid bars to represent parallel splits and parallel joins, diamonds to represent choices, a bullet to point at the start of the process, and a "lamp" (crossed circle) to represent the end of a flow. A parallel split indicates that parallel processes start. The ordering of actions in different parallel procsseses is not specified: If A is parallel to B, this means that A can occur before, during or after B.

We structure the layout of the diagram in such a way that the actions of one business actor are listed in one column. We put the name of the actor at the top of the column and separate the columns with dotted lines so that each actor has its own swimlane.

Fig. 2 shows three actors, a consignee, a shipper and a carrier. We use the diagram to represent a coordination process, which means that it is a *prescription*: the three actors are *required* to behave this way. They may *actually* behave in many other possible ways, but this is not represented in the diagram.

The three actors are required to perform their part of the process in parallel, which is why we start with a parallel split. The consignee must start with sending a purchase order (P.O.) to the shipper, who must respond by sending an acknowledgement. The shipper then must negotiate a transport agreement with a carrier (there may be an umbrella contract with a carrier, in which case the negotiation is very simple). Having agreed on a transport agreement, the shipper pays the fee for the transport and ships the goods. In a parallel process the shipper sends an invoice to the customer and a document called GPS-A, which is a certificate of origin. Upon receiving these, the consignee must pay and must receive the goods from the carrier.

Note that the two business transactions in figure 1 are implemented in activities distributed in time, which in different process executions may occur in different sequences. The transaction between shipper and carrier is implemented by the money transfer between them and the carrier activities "Receive goods" and "Deliver goods". The transaction between consignee and shipper are implemented by the money transfer between these two actors, and the series of activities starting with "Ship goods(Shipper)" and ending with "Receive goods(Consignee)". Due to the presence of parallel activity flows, there is no simple sequence between the two transactions in the value model. For example, in one execution, the consignee may pay the fee for the good after he receives the good, and in another execution, he may pay before the goods are shipped.

A second observation is that the transport service provided by the carrier does not occur as an activity in the coordination process. It is sufficient to know in the coordination process that after the carrier receives the goods from the shipper, it will delivers them to the consignee. This is modeled by the activities "Receive goods" and "Deliver goods" of the carrier.

It is possible to give activity diagrams a formal execution semantics [11]. Here, we make three remarks about our intended execution semantics: First, a parallel split starts two or more processes whose actions may be interleaved in any other. For example, our model specifies that the consignee must pay after receiving the documents, but it allows payment to occur before or after receiving the goods. Similarly, the two parallel processes in which the shipper interacts with the consignee and the carrier, respectively, may be executed in any interleaving.

A second remark to be made about execution semantics is that the control arrow (the unbroken arrow) contains a wait state: if an actor performs activity A followed by B, then the actor may wait indefinitely inbetween. So after the coordination starts, all three actors may wait indefinitely before performing their first activity, and after an actor performed an activity, it may wait indefinitely before executing the next.

A third and final remark about control flow is that

- each activity needs its inputs when it starts,
- each activity must terminate, and
- when it terminates, an activity produces its output.

This implies that the first activity of the process in Fig. 2 is "Send P.O.".

4. BASIC CORRECTNESS CRITERIA

In which sense is the coordination process of Fig. 2 correct with respect to the value model of Fig. 1? An informal statement of the correctness property is this:

- The coordination process is correct with respect to the value model if after every possible execution of the process, the value transactions triggered by the consumer need in the value model have all been executed.

To formalize this, we must indicate when a value transaction has been performed in the coordination process. In our simple example, there are two business transactions. We now introduce two definitions.

- We define the value transaction between the shipper and consignee to have occurred when in the coordination process, the activities "Receive goods" and "Receive Money" have been executed by the consignee and shipper, respectively.
- We define the value transaction between the shipper and carrier to have occurred when the activities "Pay transport fee" and "deliver goods" have been performed by the shipper and carrier, respectively.

Note that these definitions are part of process design. They have to be validated with the businesses.

The atomicity of the value transactions means that for every occurrence of a consumer need, both transactions are performed entirely or none at all. Using the above two definitions, this translates in the coordination process into the property that in every terminating execution of the coordination process, the actions defined to implement a value transaction are either all executed or none at all.

We can formalize by defining the proposition *ReceiveGoods*$(Consignee)$ to be true exactly in the state where the consignee just finished executing the activity "Receive goods". We define the propositions $ReceiveMoney(Shipper)$,

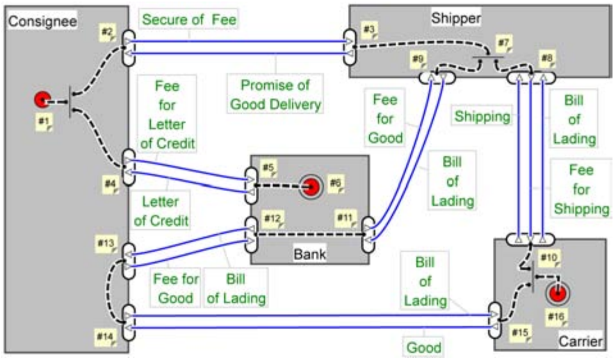

Figure 3: A value model of a business transaction using a bank as trusted third party.

$ReceiveTransportFee(Carrier)$ and $DeliverGoods(Carrier)$ analogously. We can then express the correctness property in linear temporal logic [17] as follows:

$$(\Diamond ReceiveGoods(Consignee) \wedge \Diamond ReceiveMoney(Shipper))$$

$$\wedge(\Diamond ReceiveMoney(Carrier) \wedge \Diamond ReceiveMoney(Carrier)).$$

$\Diamond$ means "some time in the future". So the formula is true if in all execution paths, $ReceiveGoods(Consignee)$, $ReceiveMoney(Shipper)$, $ReceiveTransportFee(Carrier)$ and $DeliverGoods(Carrier)$ become true before the execution terminates (but necessarily all at the same time). We can check this by mapping the activity diagram to the input of a model checker [9] and using the model checker to check this formula [11, 12]. However, we are not concerned with formalization of correctness criteria in this paper, but with the analysis of possible definitions of correctness, and with the methodological support for the service designer we can extract from that. To continue this exploration, we now drop one of our trust assumptions to see what happens with our value and coordination processes.

5. DISTRUSTING THE CONSIGNEE

As pointed out, the activity diagram of Fig. 2 is not a description of all possible things that might happen, but a prescription of what must happen in a particular coordination process. We have checked that this process implements the value model of Fig. 1. Now we drop one of our trust assumptions: What if the consignee does not pay? What if the consignee does not follow the prescribed process of figure 2? This consignee may or may not pay. The standard solution to avoid this in international trade procedures is to use a trusted third party, in most cases a bank, that handles payment by the consignee.

5.1 Value model

The value model of this solution is given by Gordijn and Tan [16] and is shown in Fig. 3. We number the nodes in the diagram for ease of reference; this is not part of the notation. The idea is that the bank has a relationship with both shipper and consignee. If the bank has no relationship with the shipper, then a second bank is introduced that does have a relationship with the shipper. To keep the picture simple we consider the situation with one bank only. We now make the following trust assumption:

- The bank, shipper and carrier are to be trusted.

The bank now issues a letter of credit (LoC) to the consignee, stating the credit-worthiness of the consignee. Essentially, the LoC can be seen as a service delivered by the bank to the consignee that guarantees the shipper that he will be paid for his delivered good. This is a service to the consignee, not to the shipper, because the consignee has an interest in getting the shipper to deliver the goods. This is made clear in the value model because in return for this service, the consignee pays the bank a fee (transaction between interfaces #4#5). Transaction #4#5 facilitates transaction #2#3, the exchange of a promise by the shipper to deliver a good, for a guarantee of the consignee pay for that good. Payment by the consignee is guaranteed by the LoC. Note that whenever a consumer need occurs (#1), both business transactions take place due to the AND-fork.

The carrier produces a bill of lading (BoL) and hands this over to the shipper in return for the goods to be delivered (#8#10). The BoL is a guarantee that the goods have been shipped [6]. Whoever has the BoL, can claim the goods as owner. It is a performative document in that only the original can have this function; copies of it do not have the legal force of a guarantee. The shipper gives the BoL to the bank and gets paid for the delivery of the goods (#9#11). The bank in turn collects the fee from the consignee, who receives the BoL in return (#12#13) and thereby can claim the goods. When the carrier delivers the goods (#14#15), he hands over the goods in return for the BoL.

This process does not make it impossible for the consignee to omit payment, but it does alter the distribution of risk: Under the trust assumptions listed above, the shipper is sure of being paid, and the consignee runs the risk of losing his relationship with the bank when he does not pay.

5.2 Coordination process

An activity diagram of a coordination process that implements this value model is shown in Fig. 4. It extends the coordination process of Fig. 2 at several points:

- The consignee acquires a LoC from the bank. The bank sends a copy of this to the shipper, who then knows that it is safe to ship the goods.
- In parallel to shipping the goods, the shipper receives a BoL from the carrier and uses this to obtain payment from the bank.
- After receiving the documents from the bank, the consignee pays for the BoL, and in parallel to that waits for the goods.
- When the goods arrive, the consignee obtains them by handing over the BoL.

It is interesting to see when business transaction #2#3 takes place in this process. The transaction takes place when the consignee terminates the activity "Receive LoC". Given the trust assumptions, at this point the shipper obntains the desired guarantee, and is therefore willing to promise delivery of goods.

6. MORE CORRECTNESS CRITERIA

To show that the updated process is correct with respect to the updated value model, we must again associate with

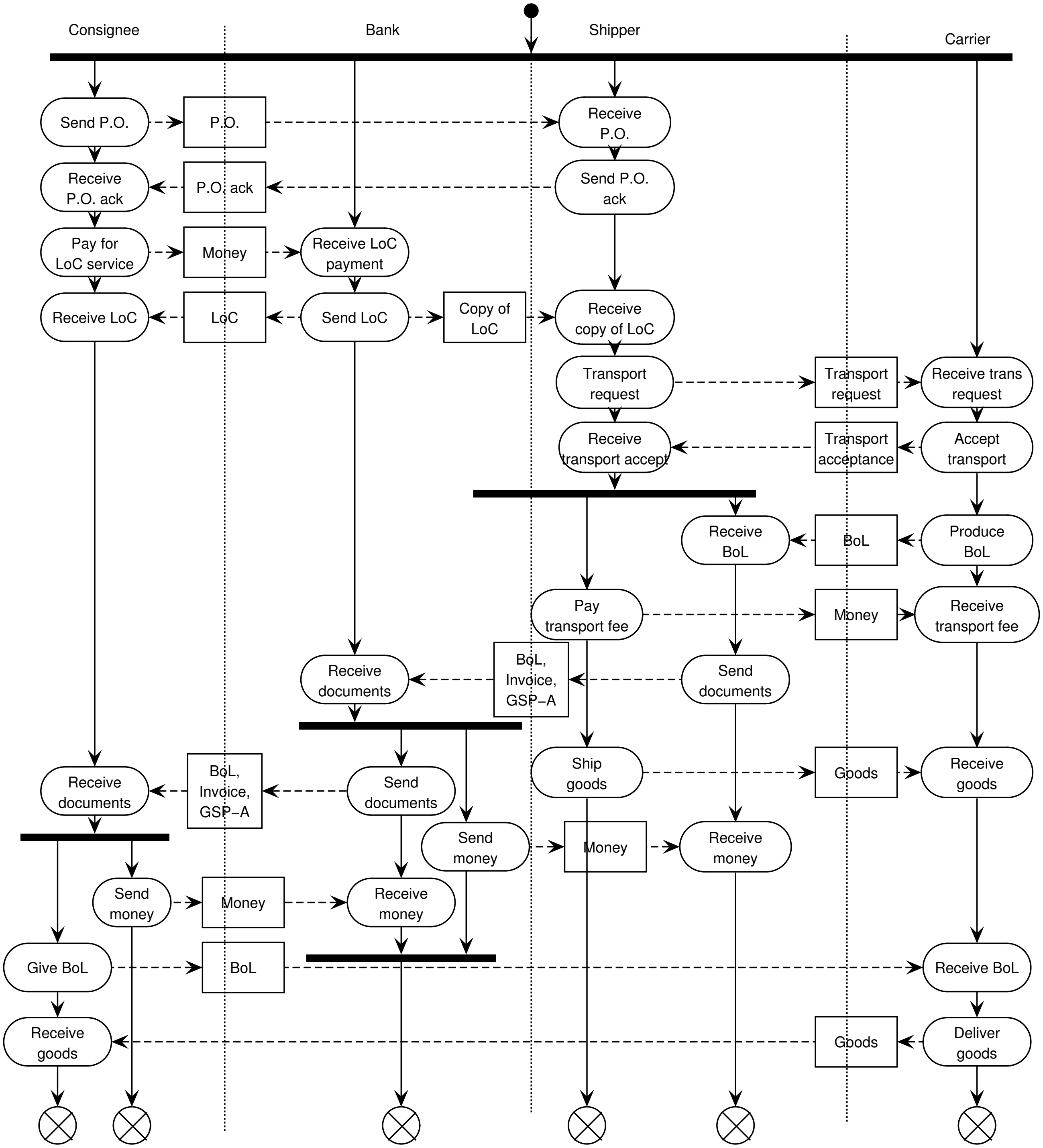

Figure 4: A coordination process that correctly implements the value model of figure 3.

every business transaction a correctness formula. For example, the correctness formula of transaction #4#5 is

$$\Diamond \mathit{ReceiveLoC}(Consignee),$$

where $\mathit{ReceiveLoC}(Consignee)$ is defined to be true exactly in the state where the consignee terminated the activity "Receive LoC". The other formulas are defined similarly. We abbreviate these formulas by the numbers that indicate the business transactions. The formula representing the correctness of the coordination process with respect to the value model then says that when a consumer need occurs (#1 in the value model), then all business transactions in the dependency path starting from #1 are triggered, according to the Boolean logic of the connectors in the path. In the value model of Fig. 3, there is one and-split (#7), so this leads to a simple conjunction as follows:

$$\begin{aligned}&\#4\#5 \wedge (\#2\#3 \wedge\\ &\qquad (\#9\#11 \wedge \#12\#13 \wedge \#14\#15) \wedge\\ &\qquad \#8\#10)\end{aligned}$$

This formula follows the structure of the dependency path.

At this point it is useful to step back and see what we have done. We started with the claim that a coordination process must be correct with respect to a business model. This claim was formalized by choosing the e^3*-value* notation for business models and activity diagrams for coordination processes. Furthermore, we found that the coordination process is really a *prescription.* It is a trade procedure to be followed by business actors. We then formulated the question When is an activity diagram of a coordination process correct with respect to an e^3*-value* value model? The answer we propose is that the coordination process should contain activities that realize the business transactions in the value model, according to the logic of the dependency paths. We used LTL to suggest how this can be formalized.

Now, we found a correct coordination process in Fig. 2, but we also found that this is correct under some strong trust assumptions. The correctness problem is therefore shifted from the relationship between the coordination process and the value model to the relationship between the coordination process and actual business behavior.

Both activity diagrams are correct with respect to their respective value models, according to the proposed correctness criterion stated above. However, there are many cases where the trust assumptions of the first coordination process are not satisfied and we need the second coordination process, which creates security for the shipper that he gets paid. Even this model contains a trust assumption often times violated, namely that the shipper always delivers his goods to the carrier. We can drop this assumption if we add an insurance company that insures the freight and that pays the consignee when the goods are paid for but not delivered—and we add the assumption that the insurance company can be trusted. The insurance company must be paid by the shipper, so the value model must be adapted to this as well, and this in turn creates additional activities and service provisions in the coordination process. The more trust assumptions we drop, the more complex the coordination processes and value model become, and the closer they get to the actual procedures for overseas trade [5].

7. METHODOLOGICAL DISCUSSION

Our correctness criterion has two properties.

- The coordination process involves the same business actors as the value model.
- A transaction between actors A_1 and A_2 in the business model need not be realized by an activity of A_1 or A_2 in the coordination process. We saw that transaction #2#3 is implemented by an activity involving the consignee but not the shipper. In general, a business transaction between actors A_1 and A_2 in the business model may be realized by an activity involving different A_3 or A_4 in the coordination process.

We extract the following methodological guidelines for coordination process design from our analysis.

- The business requirements for the process are derived from the business model. So start by representing the business model by a value model. Note that the coordination process designer does not *design* the business model. He *describes* it, using a particular notation, namely e^3*-value* . Note also that it is not feasible to design a business model in its full complexity. In practice, we first design a simple model that assume maximum trust among partners and we then stepwise remove the trust assumptions that are not realistic.
- Note the trust assumptions made by the value model. The process must make the same trust assumptions, so that the process is applicable in the same cases as the value model is. The trust assumptions determine which possible process violations you can afford to ignore. Business people are usually expert at identifying trust assumptions and risks taken (or they would be out of business), so these assumptions must be identified jointly with the business people.
- For every business transaction in the value model, decide which activities in the process implement the transaction.
- Start with assuming maximal trust and design the process accordingly. Prove correctness with respect to the value model by showing that every execution path contains the activities that implement the business transactions according to the Boolean logic of the dependency path triggered by the consumer need.
- Drop trust assumptions as necessary, adding actors and activities as needed. Again, standard trade patterns exist to deal with every dropped trust assumption.

The added value of using a value model to identify the business requirements for the coordination process is that identifies exactly the services exchanged between the business involved in the coordination. This allows us to identify essential activities needed in the coordination process, that implement these transactions, from accidental activities, used in the process for determining control flow. The essential activities must be present in every process implementing the business model, the accidental ones may differ in different implementations of the business model.

Furthermore, every business in the cooperation is an independent decision center that makes decisions in its own interests. This makes it critical for the success of the coordination design, to base this design on a business model that makes the interests of each business partner in the coordination clear. The e^3*-value* value model does this, and the correctness argument for the activity diagram then makes clear that the coordination process does indeed support these interests.

8. CONCLUSIONS AND FURTHER WORK

We presented an approach to designing coordination processes that starts by identifying essential services in the business model and designs the process so that it performs the required business transactions according to the logic of the business model. This places the coordination process in the required context of business cooperation.

A number of topics need to be elaborated. First, we need to develop automated support for generating correctness formulas from business models. Once the designer supplied correctness formulas for each business transaction, the complete formula can be generated automatically from the value model. Second, due to the presence of data, model checking activity diagrans with object flows is substantially more difficult than model checking diagrams without object flows. We yet have to extend our model checking implementation [12] with object flows. Third, there is a huge benefit to be reaped by using patterns known from organizational control theory to build coordination process patterns. Schaad and Moffett [19] have made a start towards this but they do not focus particularly on cross-organizational service coordination processes. A lot of work can be done in this area, with potentially large rewards in making web services a business reality. Finally, we want to study the automatic generation of coordination process specifications from activity diagrams. An initial proposal exists for BPEL4WS [13], but we yet have to investigate which execution semantics is implemented in this prototype. Additionally, we want to look at other notations, such BPMN (Business Process Modeling Notation) [7], that is intended to visuale BPEL4WS, and of course and see if we can formalize this to make it formally verifiable, as we have done with activity diagrams.

9. ACKNOWLEDGMENTS

The authors thank Rik Eshuis of the Technical University Eindhoven and Pascal van Eck of the University of Twente for constructive comments on an earlier version of this paper.

10. REFERENCES

[1] W. v. d. Aalst. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications.* Springer, 2004.

[3] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawanara. Business Process Execution Language for Web Services Version 1.1. Technical report, BEA Systems, IBM, Microsoft, SAP, Siebel, 5 May 2003.

[4] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riener, S. Struble, P. Takasci-Nagy, I. Trickovic, and S. Zimek. Web Service Choreography Interface (WSCI) 1.0. Technical report, BEA Systems, Intalio, SAP, SUN Microsystems, 8 August 2002.

[5] R. Bons, R. Lee, and R. Wagenaar. Implementing international electronic trade using open-edi. *European Journal of Information Systems and Inter-Organizational Networks*, 1994.

[6] R. Bons, R. Lee, R. Wagenaar, and C. Wrigley. Modelling inter-organizational trade procedures using documentary Petri nets. In *Proceedings of the Hawaii International Conference on System Sciences*, 1995.

[7] Business process modeling notation, working draft (1.0), August 25 2003. `www.bpmi.org`.

[8] L. Cabrera, G. Copeland, W. Cox, M. Feingold, T. Freund, J. Johnson, C. Kaler, J. Klein, D. Langworthy, A. Nadalin, D. Orchard, I. Robinson, J. Shewchuk, and T. Storey. Web Services Coordinatioon (WS-Coordination). Technical report, BEA Systems, IBM, Microsoft, September 2003. `ftp://www6.software.ibm.com/software/developer/library/ws-coordination.pdf`.

[9] E. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* The MIT Press, 1999.

[10] ebXML Business Process Specification Schema Version 1.01. `http://www.ebxml.org/specs/ebBPSS.pdf`, 11 May 2001.

[11] R. Eshuis and R. Wieringa. Verification support for workflow design with UML activity graphs. In *24th International Conference on Software Engineering (ICSE 2002)*, pages 166–176, 2002.

[12] R. Eshuis and R. Wieringa. Tool support for verifying UML activity diagrams. *IEEE Transactions on Software Engineering*, Accepted for publication.

[13] T. Gardner. UML modelling of automated business processes with a mapping to BPEL4WS. In *First European Workshop on Object Orientation and Web Service (EOOWS), Darmstadt, Germany*, 21 July 2003.

[14] J. Gordijn and J. Akkermans. Value-based requirements engineering: Exploring innovative e-commerce ideas. *Requirements Engineering Journal*, 8(2):114–134, 2003.

[15] J. Gordijn, J. M. Akkermans, and J. C. van Vliet. Business modelling is not process modelling. In S. W. Liddle and H. C. Mayr, editors, *Conceptual Modeling for E-Business and the Web*, volume 1921 of *LNCS*, pages 40–51, Berlin, D, 2000*c*. Springer Verlag. Also available from `http://www.cs.vu.nl/~gordijn/`.

[16] J. Gordijn and Y.-H. Tan. A design methodology for modeling trustworthy value webs. *International Journal of Electronic Commerce*, to appear.

[17] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent System Specification.* Springer, 1992.

[18] RosettaNet. Rosettanet overview. `www.rosettanet.org`, Web site.

[19] A. Schaad and J. D. Moffett. A framework for organisational control principles. In *18th Annual Computer Security Applications Conf.*, page paper 25, Las Vegas, Nevada, 2002. ACSAC, Columbia.