On the Alignment of Business Models and Process Models

Ananda Edirisuriya and Paul Johannesson

Department of Computer and System Sciences, Stockholm University
Forum 100, SE-164 40 Kista, Sweden
{si-ana,pajo}@dsv.su.se

Abstract. A value transfer is an interaction between two actors where the exchange of object of economic value takes place with their rights, custody and some evidence. Business models are used to describe organizational businesses by focusing on value transfers between actors. Process models are using to describe organizational business processes. They can be used for executing and coordinating value transfers. The paper proposes a way to align business models with process models grounded on value object and value transfer analysis.

Keywords: Business Model, Process Model, Activity Dependency Model, Value Object, Value Transfer, Right, Custody, Evidence document.

1 Introduction

Customer demands are unlimited, changing all the time and changing in different ways. This makes it necessary to introduce new products or change the requirements of existing products continually. As a result, new business opportunities are created which are met by creating new manufacturing or service processes. This increase in complexity often requires enterprises to operate as a value web. In a value web, a group of enterprises join together to fulfil customer needs, by excelling their own specific specialty, products, and services [1].

Business models and process models are two types of model in the chain of models used by enterprises to describe different aspects of a business in a value web. A business model identifies the actors involved in a value web, resources, value transfers among actors, and how the value is created and marketed. In a value transfer, an actor provides a right on a resource, custody of the resource and an evidence document to a recipient actor [2]. A business model provides a high-level view of the activities taking place within and between actors by focusing on the 'what' aspect of a business. On the other hand, a process model depicts the behaviour of actors, in particular the order of transferring resources. A process model focuses on the 'how' aspect of a business by explaining the operational and procedural details of business communication, like control flows, data flows, and message flows. The purpose of a process model is to facilitate the coordination of value transfers and communication among business partners.

The important characteristics of a process modelling technique are business orientation, traceability, and flexibility [3]. *Business orientation*: The concepts used to

D. Ardagna et al. (Eds.): BPM 2008 Workshops, LNBIP 17, pp. 68–79, 2009.

describe process models should be business oriented and understandable by business users. *Traceability*: The design decisions taken during the designing of a process model should be traceable and justifiable using business terms. *Flexibility*: The process modelling techniques should allow changes to the structure of the process model at design and run time. The design of a business model is motivated by the goals of the business. The business goals of an enterprise may evolve over time and these changes should somehow be reflected in the business models and process models. This shows the necessity of linking the operational process model and the upper-level business model.

Several approaches have been put forward to align business models and process models. The unified framework of Jayaweera [4] integrates business models and process models. A chaining methodology is proposed by Andersson et al. [5] to derive a process model from a business model. In the e^{3}transition approach [6], an intermediary model known as the e^{3}transition model is proposed by extending the economic *e^{3}value* model. The main weakness of these approaches is that they do not properly address the issues of traceability in designing process models. The notion of an activity dependency model is introduced in [3] to bridge the gap between a business model and a process model. This intermediary model identifies, classifies, and relates the activities needed for executing and coordinating value transfers. However, this approach does not address some important issues in value transfers. Hence, the process model design using this approach misses some important business processes.

The research question addressed in this paper is 'How can a process model be systematically derived from a business model?' The contribution of this paper is twofold. The paper: (i) extends the activity dependency model in [3] by complementing it with value transfer analysis and (ii) introduces a set of process patterns to increase business process flexibility.

The paper is structured in the following way. In section 2, we introduce a business model, a process modelling notation and process patterns in general. Section 3 describes the concepts used in the activity dependency model. In section 4, we introduce the transformation rules. Section 5 concludes with a discussion of the results, future research and limitations.

2 Background

In this section, we first explain the main concepts in the *e^{3}value* business model. We then describe business process models using BPMN notation and process patterns.

2.1 The *e^{3}value* Model

The *e^{3}value* business ontology was originally proposed to model the value networks of cooperating business partners [7]. The ontology aims at identifying the exchanges of objects of economic value (value objects) between the involved actors in a business collaboration. The *e^{3}value* ontology provides a rich set of software tools to design and analyse value webs, including a graphical notation. It also provides a minimal set of concepts and relations, thus making it easier to be understood by all the involved stakeholders. These factors motivate us to use *e^{3}value* ontology as our basis for a business model in this paper.

The main concepts used in *e³value* ontology are: actor, value object, value port, value offering, value interface, value exchange, value transaction, market segment, and value activity. An *actor* is an economically independent entity. An actor is perceived by its environment as often, but not necessarily, a legal entity. Customers and suppliers are examples of actors. A *value object* is something that actors exchange and has an economic value for at least one of the actors involved. Services, products, and money are examples of value objects. A *value port* is used by an actor to provide or receive value objects to or from other actors. A value port has a direction, *in* or *out*. An *in-port* is used to receive a value object (e.g., receives a payment) and an *out-port* is used to provide a value object (e.g., sends a product). A *value offering* is a set of equally directed value ports belonging to one actor. A *value interface* groups value ports and consists of in-port(s) and out-port(s) that belong to the same actor. A *value exchange* is a pair of value ports of opposite directions belonging to different actors. When an actor gives up something, another actor takes it up. A *value transaction* is a set of economic reciprocal value exchanges between one or more actors. A value transaction aggregates all value exchanges. A *market segment* is a group of value interfaces belonging to actors, who may value exchanging economic objects equally. A *value activity* is an operation that produces value objects and could be carried out in an economically profitable way for at least one actor.

We introduce the Massively Multiplayer Online Games (MMOG) as a running example. This business scenario involves four actors: a game provider (GP), an Internet service provider (ISP), a shipper, and a customer. A game provider is responsible for producing the game contents as well as selling and distributing its software CDs to the customers. An ISP provides a hosting service and Internet services for a game provider and receives a payment in return. The customers make a payment to the game provider to play games. The business scenario is modeled using the *e³value* model and is shown in Fig. 1.

2.2 Business Process Model

The purpose of a process model is to describe the internal business processes of an organization. A business process 'is a specific ordering of work activities across time and place, with a beginning, an end, and clearly defined inputs and outputs; a structure for action' [8].

The notation we use for process models in this paper is BPMN [9], a standard developed by the Business Process Management Initiative (BPMI). The goal of BPMN is to provide an easily comprehensible notation for a wide spectrum of stakeholders ranging from business domain experts to technical developers. The BPMN specifications can be mapped to executable XML language XPDL [10].

The BPMN elements we use in this paper are sub-process, event, gateway, sequence flow, message flow, lane and pool. A *sub-process* is used to represent activities performed by an actor and shown by a rounded rectangle with a '+' sign. A sub-process can be repeated: this is represented by a circular arrow inside the rounded rectangle. An *event* is something that happens externally during the course of a business process. A *start* event starts the flow of a business process, while an *end* event terminates the flow. The flows of activities are controlled using gateways for branching, forking, merging, and joining paths. They are shown using a diamond symbol.

Sub-processes, events, and gateways are connected by sequence flows. A *sequence* flow is represented by an arrow which indicates the order of execution of the activities. A *pool* represents a participant in a process and is shown by an oblong rectangle. The *lanes* are patricians of a pool. A pool contains sub-processes, events, gateways, and sequence flows between them. A *message* flow shows the flow of the messages between actors and is shown using a dotted arrow. An example of a BPMN business process diagram is given in Fig. 3.

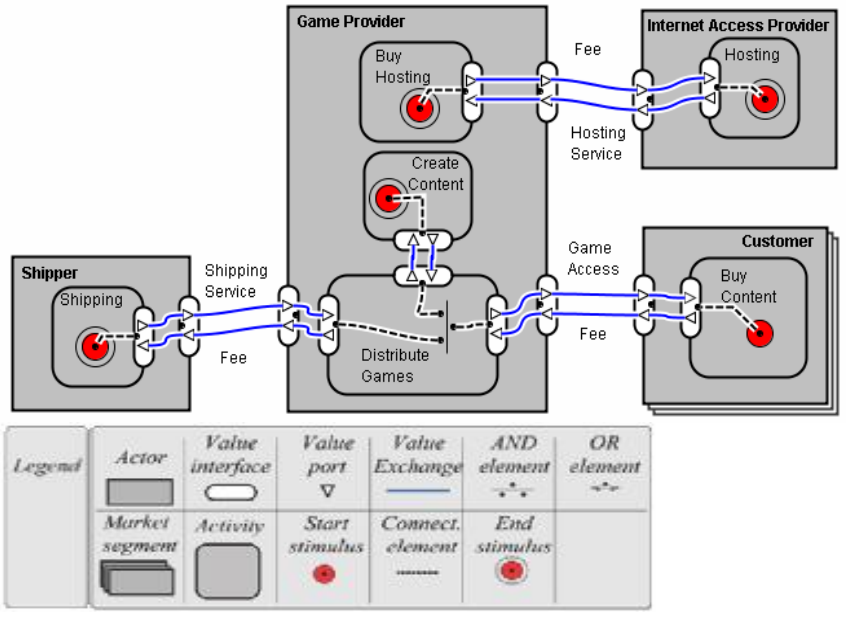

Fig. 1. *e*^{3}*value* model for the MMOG case

2.3 Process Patterns

A designing and creating process model is a complicated and time-consuming task. In designing a process model, a starting point is to use the business model of an enterprise [3]. The derivation process is considered to be a non-deterministic task and cannot be fully automated. The additional knowledge about the intended processes needs to be introduced. A good design practice to overcome these difficulties is to use already provided solutions: process patterns. A *pattern* is a description of a general solution to a specific analysis and design problem, when to apply the solution, and when and how to apply the solution in a new context [11]. It is up to a designer to compare the alternatives and select the most attractive pattern given his/her business goals. In the Open-EDI [12] initiative, business collaboration is divided into a number of phases: planning, identification, negotiation, actualization, and post-actualization. These phases can be used as a fundamental process pattern. In this work, we only consider the negotiation and actualization phases.

3 Activity Dependency Model (ADM)

In this section, we present a notion of an activity dependency model (ADM). Section 3.1 discusses the concepts used in constructing an ADM. Section 3.2 discusses the internal structure of a value object and a value transfer.

3.1 Notation for an Activity Dependency Model

A business model shows the exchange of value objects between actors and from where these value objects are being produced. The order of exchanges of value objects are not modelled in a business model. On the other hand, the behaviours of actors are modelled in the process models. A process model shows the order of exchange of value objects, the order of how they are created, message exchanges, and data flows. This means both models contain information which is not present in the other model. The purpose of an ADM is to bridge the gap between business models and process models. An ADM provides more details than a business model and fewer details than a process model. It identifies and classifies the activities that are necessary to exchange resources, produce resources, deliver services, and the relations that exist among these activities.

The main concepts used in an ADM are actor, resource, activity, and relationships between activities. An *actor* is someone who is able to participate in an activity. Actors can be categorized as an *actor type*. The actors in one category share some common properties. Actors exchange resources. A *resource* is an object that is regarded as valuable by some actors. A resource may have one or many properties. These properties are called *features* [13]. An example is the colour of a car. When an actor has a resource, he can use it to produce other resources or consume it to gain some experience, or trade it with other actors. A *right* on a resource means how an actor is entitled to use the resource. There are different types of rights: ownership, use, and possession are a few examples. When an actor has the ownership of a house, he/she can sell it or rent it.

An activity can change a feature or a right on a resource. In the work of Hruby [13], the activities are classified as transfer or conversion. A *transfer* activity transfers the right on a resource from one actor to another. When transferring the right on a resource, one actor gives up the right on the resource and another actor may receive the right on the resource. We distinguish between two types of transfer activities: take and give. A *take* activity receives the right on a resource, and a *give* activity gives up the right on a resource. A *conversion* activity changes some features of a resource by using some other resources. A *production* activity is a conversion activity that creates or modifies a resource by changing some features of a resource.

In the work of Andersson et al. [2], transfer and conversion activities have been categorized, by using the notation of process as, transfer, exchange, transaction, and transformation. They are a specialized set of processes. A *transfer* is a process consisting of a give transfer activity and a take transfer activity associated with two different actors (types). A transfer specifies that one actor (type) is prepared to give a right on a resource to another actor (type), who takes it. An *exchange* process consists of all the transfer activities associated with the same actor (type) in a one duality. Thus, an exchange process is a number of give transfers and take transfers which

belong to the same actor (type) in a one duality. A *transaction* is a process consisting of a number of transfers, hence a number of transfer processes. A *transformation* is a set of conversion activities all associated with the same actor (type).

Additions to the above several other constructs are needed to describe an ADM. An ADM is always constructed with respect to a single actor called a base actor. This means that an ADM focuses on one actor in a business model and the value transactions involving this actor. An ADM can be seen as a graph with nodes and edges. The nodes represent exchange processes, production processes, and coordination processes. A *coordination* process coordinates the value exchanging processes as well as the production processes associated with a transaction process.

The edges represent dependencies or relations between these processes. There are three kinds of dependencies: trigger, flow, and trust. A *trigger* dependency [3] from a coordination process to a production process or exchange process expresses that they are initiated and managed by the coordination process. A *flow* dependency [14] from one process to another expresses that the resource obtained by the first process is needed as input to the second process. For example, computer games can be created only after obtaining Internet and hosting services. A *trust* dependency [4] between a take activity and a give activity or a give activity and a take activity of an exchange process of the same coordination process expresses that the first activity has to be carried out before the second activity as a consequence of low trust between the

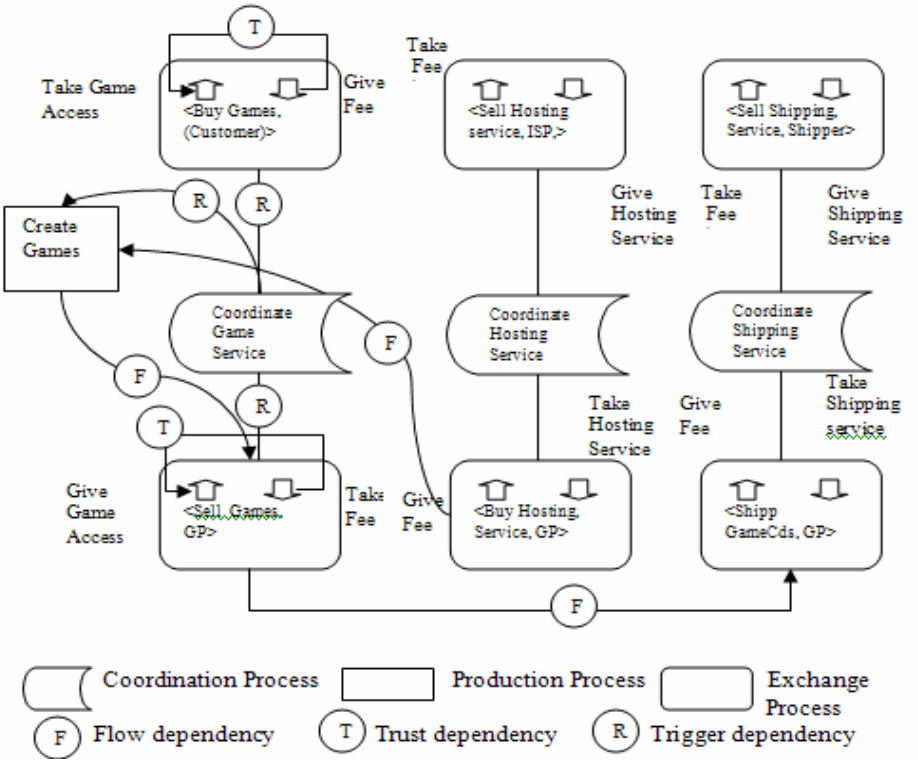

Fig. 2. An activity dependency model

involved actors. An example could be a game provider requiring a payment from a customer before delivering a game service. The designer of the model has to add these dependencies using the business domain knowledge.

An example of an ADM which is based on the *e*^{3}*value* model in section 2.1 is shown in Fig. 2. The legends describe the meaning of each symbol. The diagram shows three columns of coordination and exchange processes corresponding to the three value transactions of the base actor (game provider). The *Coordinate Game Service* coordinates the game provisioning value transaction. In the *Sell Games* exchange process, a game provider (GP) provides game access and receives a fee. An exchange process can be described by the template <*Process Name, Actor (Type)*>. An actor (type) represents a participant in a value exchange and the process name is the role played by the actor. *Take Fee* and *Give Game Access* are transfer activities in the *Sell Games* exchange process. There is one production activity, *Create Games*, to produce the resources required for the exchange. There are a number of flow, trust, and trigger dependencies. For example, there is a flow dependency from *Create Games* to *Sell Games*, meaning that games must be produced before selling them. The trust dependency from *Take Fee* to *Give Game Access* in *Sell Games* expresses that the customer must pay before receiving game access services. The trigger dependency from *Coordinate Game Service* to *Buy Games* exchange process meaning that the latter is coordinated by the former. We distinguish between actor and actor type within an exchange process by describing an actor type (Customer) within parenthesis.

3.2 Analysis of Value Object and Value Transfer

A value object is something that actors transfer and that has an economic value. When a resource is transferred from one actor to another, what is actually transferred is the ownership of the resource. When someone has the ownership of a resource, s/he can enjoy the product according to her/his wish. So, the ownership can be seen as a bundle of rights. Hence, a value object could be a certain right on some resource [2].

A value transfer involves a value object. Therefore, in a value transfer both the resource being transferred and the right on the resource must be specified. The party that holds the right should be able to exercise the right. A right for one party means an obligation for the other party. For example, when someone buys software through Amazon, the CDs should be delivered to the desired destination. This is known as transferring the *custody* of the resource [2]. When a buyer has custody of a resource, s/he can enjoy the right on that resource. A shipper may have the right to transfer the CDs to a desired destination, but not to use them.

A value transfer may also include the transfer of some documentary evidence [2] to certify that a buyer has a certain right on the resource. An example is an e-Voucher that certifies its owner has the right on the resource. Thus, a value transfer is an aggregation of right on a resource, custody of a resource and an evidence document. The first component is compulsory, while the last two are optional. For example, when buying shares on the stock market, a buyer is not necessarily given the custody of the resource. In an instantaneous exchange of goods for money, the transfer of right is performed tactically and no evidentiary document is provided. The purpose of this analysis is to identify the type of processes that should be included in a process model.

4 Transformation Rules

After investigating a large number of examples, we identify a set of transformation rules to map one model to another. Some of the rules are more natural and others are identified in an iterative process. In section 4.1, we introduce a set of rules to derive an ADM from a business model. Section 4.2 proposes a set of rules to derive a process model from an ADM.

4.1 Transforming the *e³value* Model to an Activity Dependency Model (ADM)

An ADM can be partially derived from the business model on which it is based. Let BM denote an *e³value* model.

R1: For every *value transaction* in BM, introduce a *coordination process* in ADM.
The purpose of a coordination process is to coordinate all the activities relevant to a value transaction.
R2: For every *value interface* in BM, introduce an *exchange* process in ADM.
R3: For every *in-port* in a *value interface* in BM, introduce a *take* activity within the exchange process introduced in *R2*.
R4: For every *out-port* in a *value interface* in BM, introduce a *give* activity within the exchange process introduced in *R2*.
The purpose of a take and a give activity, respectively, is to receive and provide a right on a resource. An exchange process bundles these together.
R5: For every *value object* in BM, also introduce if necessary an optional *production* activity in ADM to produce or modify a resource.
To keep ADM simpler and less cluttered, actors and market segments are described within the exchange processes.
R6: For each *actor* or *market segment* in BM, add an entry of type *actor* or *actor type* in the relevant exchange process.

(Note: We distinguish an actor and an actor type within an exchange process by describing an actor type within parenthesis.)

The ADM given in Fig. 2 can be partially derived by applying these mapping rules to the *e³value* model given in Fig.1. Each value transaction in the *e³value* model gives rise to one coordination process, and each value interface gives rise to one exchange process. The exchange processes within one transaction are related to the corresponding coordination process by trigger dependencies. A production activity may be added when the base actor has to produce some resource needed for an exchange process. There is a trigger dependency from a coordination process to a production process. The flow, trigger, and trust dependencies are not derivable from an *e³value* model. To construct a complete ADM, a model designer needs to add these dependencies using the business knowledge of a case involved.

4.2 Transforming an Activity Dependency Model into a Process Model

The construction of a process model from an ADM is about detailing the control flows and message exchanges between activities. Each coordination process in the

ADM becomes a process defined within a pool. A pool contains the activities needed to exchange resources between a base actor and another actor and the production activities needed to produce the resources. For the purpose of clarity, a pool is divided into three lanes. The activities carried out by each actor are defined in the inner and outer lanes. In the middle lane (*Global*), we model the activities common to both actors. Such a pool contains a series of activities, gateways, and events connected by sequence flows. The entire process model will consist of a number of such pools that communicate with each other. Let ADM denote an activity dependency model and PM denote a process model. Also let C be a coordination process and P_C a pool. A lane L in P_C can be identified by $P_{C,\,L}$. A process E in lane L of P_C can be identified by $P_{C,\,L.E}$. A process E attached by a trigger dependency to C can be identified by $C.E$.

For every *coordination* process C in ADM:

P1: Introduce a *pool* P_c in PM.
P2: Introduce an optional *negotiation* sub-process in $P_{c,\,Global}$.
P3: For every *exchange* process of C, introduce an exchange *sub-process* E_i in $P_{c,\,Actor}$. This sub-process is repeating if the other *exchange* process E_j of C involves an actor type.
P4: For every *trigger* dependency from C to a production activity *Prod*, introduce one production sub-process *Prod* in $P_{c,\,Base\ Actor}$.
P5: For every *flow* dependency from an exchange process $D.E_i$ (where D is a coordination process $\neq C$) to a production process *Prod*, add one sub-process $D.E_i$ in $P_{c,\,Base\ Actor}$. Add a sequence flow from $P_{c,\,Base\ Actor.Ei}$ to $P_{c,\,Base\ Actor.\ Prod}$.
P6: For every *flow* dependency from an exchange process $D.E_i$ (where D is a coordination process $\neq C$) to an exchange process $C.E_j$, add one sub-process $D.E_i$ in $P_{c,\,Base\ Actor}$. Add a sequence flow from $P_{c,\,Base\ Actor.Ei}$ to $P_{c,\,Base\ Actor.Ej}$.
P7: For every *flow* dependency to an exchange process $D.E_i$ (where D is a coordination process $\neq C$) from an exchange process C.E_j, add one sub-process $D.E_i$ in $P_{c,\,Base\ Actor}$. Add a sequence flow from $P_{c,\,Base\ Actor.Ei}$ to $P_{c,\,Base\ Actor.Ej}$.
P8: For every *flow* dependency from a process A_i to A_j, where the corresponding sub-processes are included in the $P_{c,\,Base\ Actor}$, add a sequence flow from $P_{c,\,Base\ Actor.Ai}$ to $P_{c,\,Base\ Actor.Aj}$.
P9: For every *trust* dependency expand the exchange process by using a suitable process pattern that will be discussed in section 4.3.

(Note: A lane can be labelled by the actor (type) name in the exchange process description.)

The sub-processes can be linked using these rules, leaving the left and right ends open. There will be a number of leftmost sub-processes $L1..Lm$ with no incoming sequence flows and a number of rightmost sub-processes $M1..Mn$ with no outgoing sequence flows. Add a gateway G to $P_{C,\,Global}$ with a sequence flow from the *negotiation* sub-process to G. There will be sequence flows from G to each of $L1..Lm$. Also, add a gateway G to $P_{C,\,Global}$ to merge the paths. There will be sequence flows from each of $M..Mn$ to G. Finally add *start* and *end* events to $P_{C,\,Global}$ and complete the process diagram.

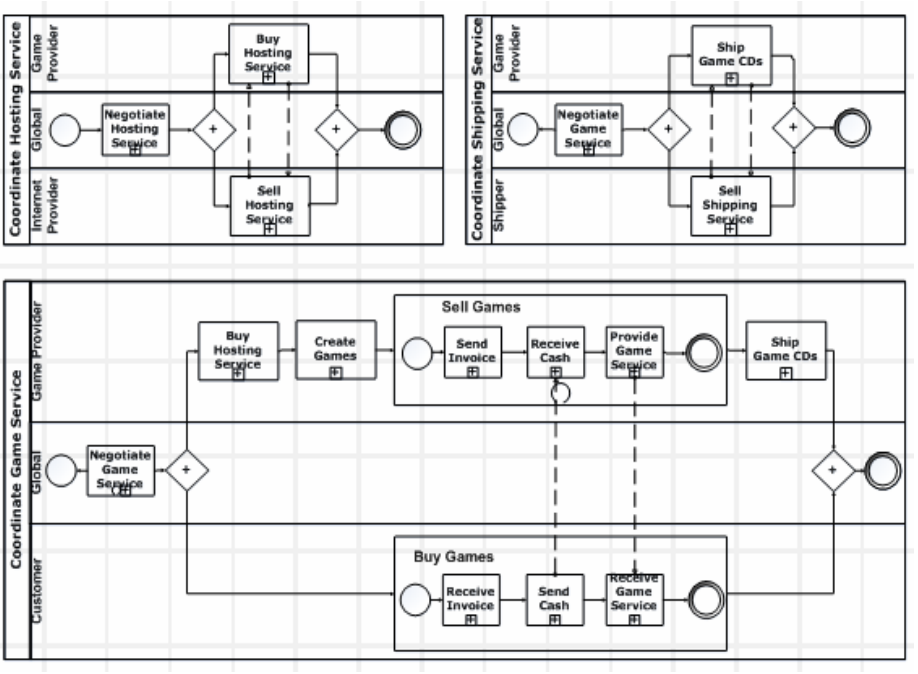

Fig. 3. Business process diagram to coordinate game service

To illustrate the method, mapping rules are applied to the running example. The business process diagram corresponding to the coordination process *Coordinate Game Service* is shown in Fig. 3. To be complete, we also show process diagrams based on the other two coordination processes *Coordinate Hosting Service* and *Coordinate Shipping Service*. The *Negotiate Game Service* corresponds to the OPEN-EDI negotiation phase (*P2*). The two sub-processes *Sell Games* and *Buy Games* are based on the trigger dependency from the coordination process *Coordinate Game Service* to the exchange processes *Sell Games* and *Buy Games*, respectively (*P3*). The sub-process *Create Games* is added as there is a trigger dependency from *Coordinate Game Service* to *Create Games* in the ADM (*P4*). In addition, sub-processes that acquire relevant resources from other transactions need to be added. The sub-process *Buy Hosting Service* is added as there is a flow dependency from *Buy Hosting Service* to *Create Games* in the ADM. *Ship GameCDs* is added because of the flow dependency from *Sell Games* to *Ship GameCDs*. The sequence flow from *Create Games* to *Sell Games* in the process model is due to the sequence flow from *Create Games* to *Sell Games* in the ADM. The trust dependency can be managed using the process patterns in section 4.3.

The *e*^{3}*value* model, the ADM and the process model described above can be represented using XML syntax. Thus, all mapping rules can be stored, maintained, and manipulated using an XML-based transformation language such as XSLT [15].

4.3 Process Patterns

A value transfer is an interaction between two actors and is an aggregation of the business objects right on a resource, custody of a resource and an evidence document. During a value transfer, a provider must provide these business objects to a recipient who receives them. To model the transfer of these business objects, we introduce a set of primitive process patterns. They are expressed using the template <*Action, Business Object (BO), Cardinality*>, where *Action*∈ {*Send/Provide, Receive*}, *Business Object*∈ {*Right, Custody, Evidence Document*} and *Cardinality*∈ {*1*,*}.

Table 1. Primitive value transfer process patterns

No	Pattern	Description
1.	<*Provide, BO, 1*>	An actor provides a BO to another actor
2.	<*Receive, BO, 1*>	An actor receives a BO from another actor
3.	<*Provide, BO, **>	An actor provides a BO of the same type to many other actors
4.	<*Receive, BO, **>	An actor receives a BO of the same type from many other actors

These patterns can be used in business transactions. There are different types of business transactions: prepaid, postpaid and point of sale etc. In a *prepaid* transaction, a provider first asks a receiver to make the payment. An invoice (evidence document) is sent to the customer for the payment. Once the payment is received, the provider provides his resource. In a *postpaid* transaction, a provider first provides his resource. An invoice is then sent to make the payment. In a *point of sale (POS)* transaction, a provider and a recipient are engaging in an instantaneous exchange of economic resources. Most of the time, the transfer of rights is performed tactically and no evidence documents are provided.

The trust dependencies shown in the coordination process *Coordinate Game Service* can be managed by expanding the sub-processes *Sell Game* and *Buy Game* as shown in Fig. 3, using the patterns described in Table 1. The purpose of *Send Cash* is to provide the fee and *Send Invoice* provides the evidence document. The transfer of a right on a resource is a communication action, thus mapped to a message exchange in the BPMN process diagram.

5 Conclusions, Future Works and Limitations

In this paper, we have discussed a method for constructing a process model from a business model. The process model designed can be used as a reference model to coordinate value exchanges among a network of business partners. The method uses value transfer analysis. A value transfer is an aggregation of right on a resource, custody, and an evidentiary document. An intermediary model known as an activity dependency model is discussed to bridge the gap between a business model and a process model. A set of mapping rules was proposed to go from one model to another.

As future work, we plan to test the completeness and correctness of the mapping rules by applying them to more complex cases. Other possible topics are to investigate whether additional kinds of activity dependencies are required and the inclusion of more OPEN-EDI phases.

References

1. Tapscott, D., Ticoll, D., Lowy, A.: Digital Capital - Harnessing the Power of Business Webs. Harvard Business School Press, Boston (2000)
2. Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Gordijn, J., Grégoire, B., Schmitt, M., Dubois, E., Abels, S., Hahn, A., Wangler, B., Weigand, H.: Towards a Reference Ontology for Business Models. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 482–496. Springer, Heidelberg (2006)
3. Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P.: A Declarative Foundation of Process Models. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 233–247. Springer, Heidelberg (2005)
4. Jayaweera, P.: A Unified Framework for e-Commerce Systems Development: Business Process Pattern Perspective (BP3). PhD Thesis, University of Stockholm, Sweden (2004)
5. Andersson, B., Bergholtz, M., Gregoire, B., Johannesson, P., Schmitt, M., Zdravkovic, J.: From Business to Process models–A Chaining Methodology. In: Latour, T., Petit, M. (eds.) Proceedings of Conference on Advanced Information Systems Engineering (2006)
6. Pijpers, V., Gordijn, J.: Bridging Business Value Models and Process Models in Aviation Value Webs via Possession Rights. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS) (2007)
7. Gordijn, J., Akkermans, M., Vliet, C.: Business Modeling is not Process Modeling. In: Conceptual Modeling for E-Business and the Web. Springer, Heidelberg (2000)
8. Davenport, T.: Process Innovation: reengineering work through information technology. Harvard Business School (1992)
9. Business Process Modeling Notion (BPMN) Specification: Business Process Management Initiative (BPMI), Version 1.0 (May 2004)
10. Workflow Management Coalition Workflow Standard: Technical report (2005)
11. Larman, C.: Applying UML and patterns: an introduction to object oriented analysis and design, 3rd edn. Prentice Hall, Englewood Cliffs (2004)
12. UN/CEFACT Modeling Methodology: User Guide, `http://www.unece.org/cefact/umm`
13. Hruby, P.: Model-Driven Design Using Business Patterns. Springer, Heidelberg (2006)
14. Malone, T., et al.: Tools for inventing organizations: Toward a handbook of organizational processes. Massachusetts Institute of Technology (March 1999)
15. XSL Transformations, Version 1.0. W3C (1999), `http://www.w3.org/TR/xslt`