A pattern and dependency based approach to the design of process models

Maria Bergholtz, Prasad Jayaweera, Paul Johannesson, Petia Wohed

Department of Computer and System Sciences
Stockholm University/Royal Institute of Technology
Forum 100, SE-164 40 Kista, Sweden
{maria, prasad, pajo, petia}@dsv.su.se

Abstract. In this paper an approach for building process models for e-commerce is proposed. It is based on the assumption that the process modeling task can be methodologically supported by a designers assistant. Such a foundation provides justifications, expressible in business terms, for design decisions made in process modeling, thereby facilitating communication between systems designers and business users. Two techniques are utilized in the designers assistant, namely process patterns and action dependencies. A process pattern is a generic template for a set of interrelated activities between two agents, while an action dependency expresses a sequential relationship between two activities.

1 Introduction

Conceptual models have become important tools for designing and managing complex, distributed and heterogeneous systems, e.g. in e-business and e-commerce, [2, 17]. In e-commerce it is possible to identify two basic types of conceptual models: business models and process models. A business model focuses on the *what* in an e-commerce system, identifying agents, resources, and exchanges of resources between agents. Thus, a business model provides a high-level view of the activities taking place in e-commerce. A process model, on the other hand, focuses on the *how* in an e-commerce system, specifying operational and procedural aspects of business communication. The process model moves into a more detailed view on the choreography of the activities carried out by agents.

A business model has a clearly declarative form and is expressed in terms that can be easily understood by business users. Therefore, business models function well for supporting communication between systems designers and business users. In contrast, a process model has a more procedural form and is at least partially expressed in terms, like sequence flows and gateways, that are not immediately familiar to business users. Furthermore, it is often difficult to understand why a process model has been designed in a certain way and what consequences alternative designs would have. In order to overcome these limitations, we believe that process models should be complemented by and be based on a more declarative foundation. Such a foundation would provide justifications, expressible in business terms, for design decisions made in process modeling, thereby facilitating communication between systems designers and business users. In this paper, we propose a designers assistant that provides a

declarative foundation for process modeling suggests a method for gathering domain knowledge. The work reported in this paper extends the work of [1] and [10] in that we propose two instruments for a declarative foundation of process models: process patterns and action dependencies. A process pattern is a generic template for a set of interrelated activities between two agents, while an action dependency expresses a sequential relationship between two actions.

The rest of the paper is organized as follows. Section 2 presents the notions of business models and process models. Section 3 introduces process patterns and makes a distinction between transaction patterns and collaboration patterns. Section 4 discusses action dependencies. Section 5 proposes a designers assistant that supports a designer in the construction of a process model. Section 6 concludes the paper and gives suggestions for further research.

2 Business Models and Process Models

For illustrating business and process models, a small running case is introduced. It is a simplified version of the Drop-Dead Order business case described in [8]. In this business scenario, a Customer requests an amount of fried chicken from a Distributor. The Distributor then requests formal offers from a Chicken Supplier and a Carrier. Furthermore, the Distributor requests a down payment from the Customer before accepting the offers from the Chicken Supplier and Carrier. As the Customer completes the down payment to the Distributor, the Distributor accepts the offer from the Chicken Supplier by also paying a down payment and the offer from Carrier. When the Chicken Supplier has provided the fried chicken and the Carrier has delivered them to the Customer, the Distributor has thereby fulfilled the Customer's order. After that, the Customer settles the final payment to the Distributor. Finally, the Distributor settles the Chicken Supplier's final payment and the payment for the Carrier.

2.1 Business Models

As a foundation for business models, we will use the REA ontology [13], which has been widely used for business modeling in e-Commerce, [17]. The REA framework is based on three main components: **R**esources, Economic **E**vents, and **A**gents, see Fig. 1[1]. An Agent is a person or organization that is capable of controlling Resources and interacting with other Agents. A Resource is a commodity, e.g. goods or services that is viewed as being valuable by Agents. An Economic Event is the transfer of control of a Resource from one Agent to another one. Each Economic Event has a counterpart, i.e. another Economic Event that is performed in return and realizing an exchange. For instance, the counterpart of a delivery of goods may be the payment of the same goods. This connection between Economic Events is modeled through the relationship *Duality*.

[1] Due to space restrictions and for the purpose of readability we use abbreviated forms of the terms in the original REA ontology. This is done by dropping the term 'Economic' for Economic Contract, Economic Commitment, Economic Resource, and Economic Agent.

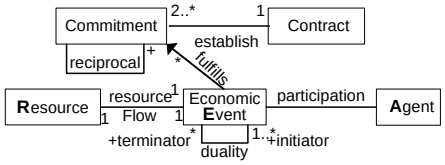

Fig. 1 REA basis for business models

Furthermore, a Commitment is a promise to execute a future Economic Event, for example fulfilling an order by making a delivery. The Duality between Economic Events is inherited by the Commitments, where it is represented by the association *Reciprocal*. In order to represent collections of related Commitments, the concept of Contract is used. A Contract is an aggregation of two or more reciprocal Commitments. An example of a Contract is a purchase order composed of one or several order lines, each one representing two Commitments (the goods to be delivered and the money to be paid for the goods, respectively).

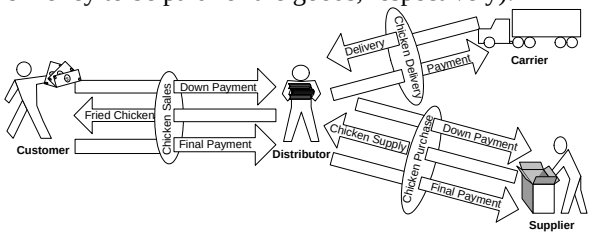

Fig. 2 Business Model for the Fried Chicken Business Case

A business model based on REA will consist of instances of the classes Resource, Economic Event and Agent as well as the associations between these. The business model for the running case described above can be visualized as in Fig. 2. Here, arrows represent Economic Events labeled with relevant Resources. The transfer of resource control from one Agent to another is represented by the direction of arrows. Ellipses represent relationships between Economic Events belonging to the same Duality.

2.3 Process Models

The notation we will use for process models is BPMN [4], a standard developed by the Business Process Management Initiative (BPMI) [3]. The goal of BPMN is to be a easily comprehensible notation for a wide spectrum of stakeholders ranging from business domain experts to technical developers. A feature of BPMN is that BPMN specifications can be readily mapped to executable XML languages for process specification such as BPEL4WS, [2].

In this paper, a selected set of core elements from BPMN have been used. These elements are Activities, Events, Gateways, Sequence flows, Message flows, Pools and Lanes. Activity is a generic term for work that an Agent can perform. In a BPMN Business Process Diagram (abbreviated BPMN diagram), an Activity is represented by a rounded rectangle. Events, represented as circles, are something that "happens" during the course of a business process. There exist three types of Events: Start, End and Intermediate Events. Activities and Events are connected via Sequence Flows that show the order in which Activities will be performed in a process. Gateways are used

to control the sequence flows by determining branching, forking, merging, and joining of paths. In this paper we will restrict our attention to XOR and AND branching, graphically depicted as a diamond with an 'X' or a '+', respectively. Lanes and Pools are graphical constructs for separating different sets of Activities from each other. A Lane is a sub-partition within a Pool used to organize and categorize Activities. Message flows depicted as dotted lines are used for communication between Activities in different Pools. (An example of them appear later in Fig. 12)

An example of a BPMN diagram is shown in Fig. 3. The diagram shows a single Business Transaction in one pool with three lanes. A Business Transaction is a unit of work through which information and signals are exchanged (in agreed format, sequence and time interval) between two Agents [17]. A Business Transaction consists of two Activities, one Requesting Activity where one Agent initiates the Business Transaction and one Responding Activity where another Agent responds to the Requesting Activity. (See Fig. 4)

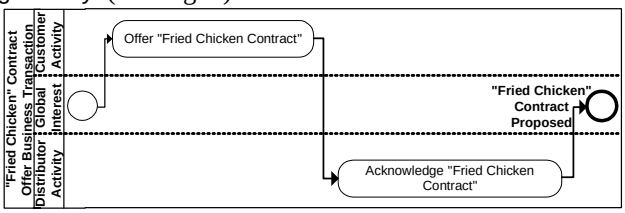

Fig. 3 Example of a BPMN diagram

Several Business Transactions between two Agents can be combined into one binary Business Collaboration. It turns out that it is often fruitful to base binary Business Collaborations on Dualities, i.e. one Business Collaboration will contain all the Business Transactions related to one Duality. This gives a starting point for constructing a process model from a business model. Each Duality in the business model gives rise to one binary Business Collaboration, graphically depicted as a BPMN diagram in a Pool. In this way, a process model will be constructed as a set of interrelated Business Collaborations.

Furthermore, a binary Business Collaboration can naturally be divided into a number of phases. Dietz, [6], distinguishes between three phases. The Ordering phase, in which an Agent requests some Resource from another Agent who, in turn, promises to fulfill the request. The Execution phase, in which the Agents perform Activities in order to fulfill their promises. The Result phase, in which an Agent declares a transfer of Resource control to be finished, followed by the acceptance or rejection by the other Agent. The ISO OPEN-EDI initiative [15] identifies five phases: Planning, Identification, Negotiation, Actualization and Post-Actualization. In this paper, we use only two phases: a *Contract Negotiation phase* in which contracts are proposed and accepted, and an *Execution phase* in which transfers of Resources between Agents occur and are acknowledged. In the next section, we will discuss how a binary Business Collaboration can be constructed utilizing patterns for these phases.

3 Generic Process Patterns

Designing and creating business and process models is a complicated and time consuming task, especially if one is to start from scratch for every new model. A good designer practice to overcome these difficulties is, therefore, to use already proven solutions. A *pattern* is a description of a problem, its solution, when to apply the solution, and when and how to apply the solution in new contexts [11]. The significance of a pattern in e-commerce is to serve as a predefined template that encodes business rules and business structure according to well-established best practices. In this paper such patterns are expressed as BPMN diagrams. They differ from the workflow patterns of [18], [16], [19] by focusing primarily on communicative aspects, while control flow mechanisms are covered on a basic level only.

In the following sub sections, a framework for analyzing and creating transaction- and collaboration patterns is proposed. We hypothesize that most process models for e-commerce applications can be expressed as a combination of a small number of these patterns.

3.1 Modeling business transactions

When a transaction occurs, it typically gives rise to effects, i.e. Business Entities like Economic Events/Contracts/Commitments are effected (created, deleted, cancelled, fulfilled). Furthermore, the execution of a transaction may cause the desired effect to come into existence immediately, or only indirectly, depending on the intentions of the interacting Agents. For example, the intention of an Agent in a transaction may be to *propose* a Contract, to *request* a Contract or to *accept* a Contract. In all three cases the business entity is the same (a Contract) but the intention of the Agent differs.

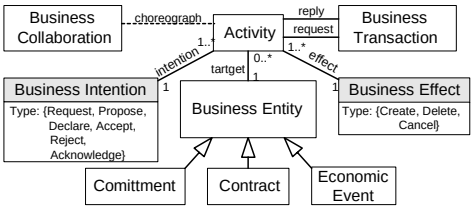

Fig. 4 Business Transaction analysis

Fig. 4 builds on REA and suggests a set of Business- Intentions, Effects and Entities. These notions are utilized in defining transaction patterns and transaction pattern instances as follows.

Definition: A *transaction pattern* (TP) is a BPMN diagram with two Activities, one Requesting Activity and one Responding Activity. Every Activity has a label of the form <Intention, Effect, Business Entity>, where Intention $\in$ {Request, Propose, Declare, Accept, Reject, Acknowledge}, Effect $\in$ {create, delete, cancel}, and Business Entity $\in$ {aContract, anEconomicEvent, aCommitment}. All End Events are labeled according to the Intention and Business Entity of the Activity prior to the sequence flow leading to the End Event.

Intuitively, the components of an activity label mean the following:

- Business Entity tells what kind of object the Activity may effect.
- Effect tells what kind of action is to be applied to the Business Entity – create, delete or cancel.
- Intention specifies what intention the business partner has towards the Effect on the Business Entity.

The meanings of the intentions listed above are as follows:

- Propose – someone offers to create, delete or cancel a Business Entity.
- Request – someone requests other Agents to propose to create, delete or cancel a Business Entity.
- Declare – someone unilaterally declare a Business Entity created, deleted or cancelled.
- Accept/Reject – someone answers a previously given proposal.
- Acknowledge – someone acknowledges the reception of a message.

Definition: A *pattern instance* of a transaction pattern is a BPMN diagram derived from the pattern by renaming its Activities, replacing each occurrence of aContract in an activity label with the name of a specific Contract, replacing each occurrence of anEconomicEvent in an activity label with the name of a specific EconomicEvent, and replacing each occurrence of aCommitment in an activity label with the name of a specific Commitment.

3.2 Transaction Patterns (TPs)

In the following sections three basic Contract Negotiation and two Execution TPs are suggested based on the framework described above.

3.2.1. Contract Negotiation TPs

The Contract-Offer TP models one Agent proposing an offer (<propose, Create, aContract>) to another Agent who acknowledges receiving the offer. The acceptance or rejection of an offer is modeled in the Contract-Accept/Reject TP, see Fig. 5.

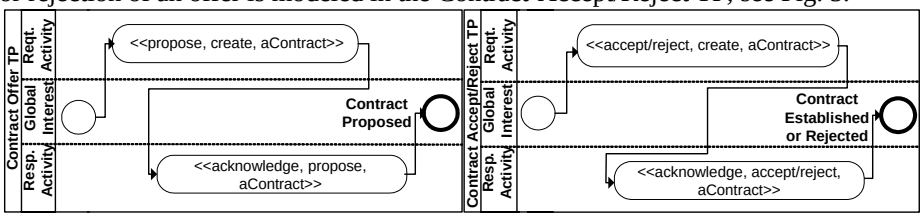

Fig. 5 TPs for Contract Negotiation: Contract-Offer and Contract-Accept/Reject

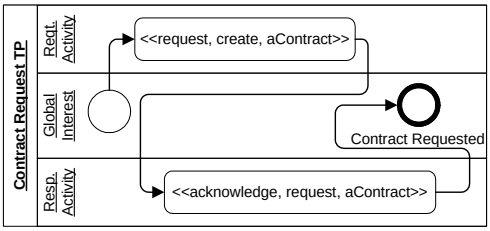

Fig. 6 TP for Contract Negotiation: Contract-Request

Fig. 6 models the Contract Request case where an Agent requests of other Agents to make an offer for aContract on certain Resources.

3.2.2 Execution TPs

We introduce two Execution TPs (see Fig. 7) that specify the execution of an Economic Event, i.e. the transfer of Resource control from one Agent to another. An example is a Chicken Distributor selling Chickens (a Resource) for $3 (another Resource).

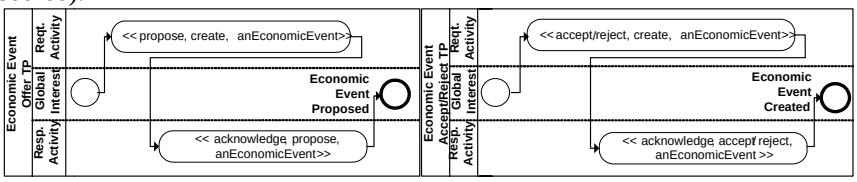

Fig. 7 TPs for Execution: Economic Event Offer and Economic Event Accept.

3.3 Assembling Transactions patterns into Collaboration patterns

An issue is how to combine the transaction patterns described in the previous section, i.e. how to create larger sequences of patterns. For this purpose, collaboration patterns define the orchestration of Activities by assembling a set of transaction patterns and/or more basic collaboration patterns based on rules for transitioning from one transaction/collaboration to another.

To hide the complexity when TPs are combined into arbitrarily large collaboration patterns, we use a layered approach where the TPs constitute activities in the BPMN diagram of the collaboration patterns.

Definition: A *collaboration pattern* (CP) is a BPMN diagram where the activities consist of transaction and collaboration pattern(s). A CP has exactly two end events representing success or failure of the collaboration, respectively. All end events are labeled according to the Intention and Business Entity of the Activity prior to the sequence flow that led to the end event.

3.3.1 Contract Negotiation CPs

The Contract Establishment CP, see Fig. 8, is assembled from the Contract-Offer and Contract-Accept/Reject TPs. An example scenario is a Chicken Distributor proposing an offer to a customer on certain terms. The contract is formed (or rejected) by the customers acceptance or rejection of the proposed offer.

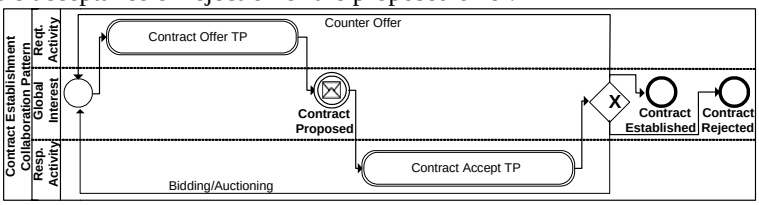

Fig. 8 Contract Establishment CP

The two recursive paths when a contract offer/request has been rejected have a natural correspondence in the business negotiation concepts ‘Counter Offer’ and ‘Bidding’

(or ‘Auctioning’) respectively. ‘Counter Offer’ refers to the switch of roles between Agents, i.e. when the responding Agent has rejected the requesting Agents offer, the former makes an offer of her own. ‘Bidding’ is modeled via the other sequence Flow from the gateway, i.e. when the responding Agent has turned down a contract offer, the requesting Agent immediately initiates a new Business Transaction with a new (changed) offer for Contract.

The Contract-Proposal collaboration pattern, Fig. 9[2], is assembled from the Contract-Request TP and the Contract-Establishment CP defined above.

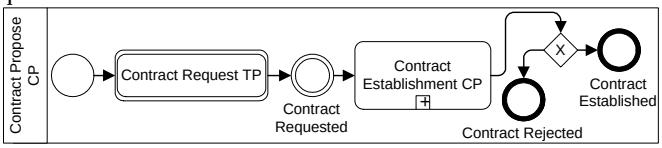

Fig. 9 Contract Propose CP

3.3.2 Execution CP

The execution collaboration pattern specifies relevant TPs and rules for sequencing among these within the completion of an Economic Event. The pattern is assembled from the Offer-Event and Accept/Reject Event TPs.

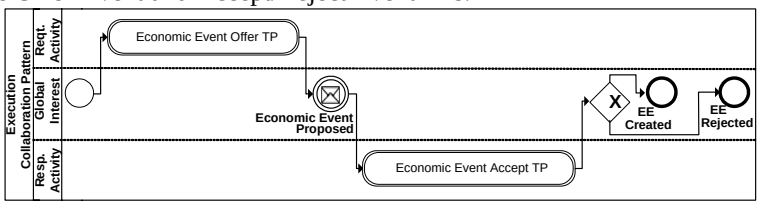

Fig. 10 Execution CP

4 Action Dependencies

The process patterns introduced in the previous section provide a basis for a partial ordering of the activities taking place in a business process, in particular the ordering based on contract negotiation and execution. We will refer to the activities involved in the different phases of a process as *contract negotiation* or *execution* activities respectively. However, the ordering derived from the process patterns only provide a starting point for designing complete process models, i.e. it needs to be complemented by additional interrelationships among the activities. These interrelationships should have a clear business motivation, i.e. every interrelationship between two activities should be explainable and motivated in business terms. We suggest to formalize this idea of business motivation by introducing the notion of action dependencies. An *action dependency* is a pair of actions (either economic events or activities), where the second action for some reason is dependent on the first one. We identify the following four kinds of action dependencies.

[2] When a CP is composed of other CPs, no lanes can be shown as the Requesting and Responding Activities are already encapsulated.

Flow dependencies. A flow dependency, [12], is a relationship between two Economic Events, which expresses that the Resources obtained by the first Economic Event are required as input to the second Economic Event. An example is a retailer who has to obtain a product from an importer before delivering it to a customer. Formally, a flow dependency is a pair <A, B>, where A and B are Economic Events from different Dualities.

Trust dependencies. A trust dependency is a relationship between two Economic Events within the same Duality, which expresses that the first Economic Event has to be carried out before the other one as a consequence of low trust between the Agents. Informally, a trust dependency states that one Agent wants to see the other Agent do her work before doing his own work. An example is a car dealer who requires a down payment from a customer before delivering a car. Formally, a trust dependency is a pair <A, B>, where A and B are Economic Events from the same Duality.

Control dependencies. A control dependency is a relationship between an execution Activity and a contract negotiation Activity. A control dependency occurs when one Agent wants information about another Agent before establishing a Contract with that Agent. A typical example is a company making a credit check on a potential customer (i.e. an exchange of the Resources information and money in two directions). Formally, a control dependency is a pair <A, B>, where A is an execution Activity and B is a contract negotiation Activity and where A and B belong to different Dualities.

Negotiation dependencies. A negotiation dependency is a relationship between Activities in the contract negotiation phase from different Dualities. A negotiation dependency expresses that an Agent is not prepared to establish a contract with another Agent before she has established another contract with a third Agent. One reason for this could be that an Agent wants to ensure that certain Resources can be procured before entering into a Contract where these Resources are required. Another reason could be that an Agent does not want to procure certain Resources before there is a Contract for an Economic Event where these Resources are required. Formally, a negotiation dependency is a pair <A, B>, where A and B are contract negotiation Activities in different Dualities.

5 A Designers Assistant

In this section, we will show how a process model can be designed based on process patterns and action dependencies. Designing a process model is not a trivial task but requires a large number of design decisions. In order to support a designer in this task, we propose an automated designers assistant that guides the designer through the task by means of a sequence of questions, divided into four steps, followed by a fifth step where the process model is generated based on the answers to questions in step 1-4, see Fig. 11.

Step 1. during which information is gathered about the Agents involved in the business process, the Resources exchanged between them, and the Economic Events through which these Resources are exchanged. The result from this step is a business model.

Step 2. during which information about the (partial) order between the Economic Events is gathered. The result from this step is an ordering of the Activities in the Execution phase of a process model.

Step 3. during which information about existing negotiation dependencies is gathered. The result from this step is an ordering of the Activities in the Negotiation phase.

Step 4. during which inter phase and inter pool dependencies are established. The result from this step is an ordering of Activities that crosses the Negotiation and Execution phases.

Step 5. during which a set of production rules are applied on the results of the previous steps in order to generate a process model.

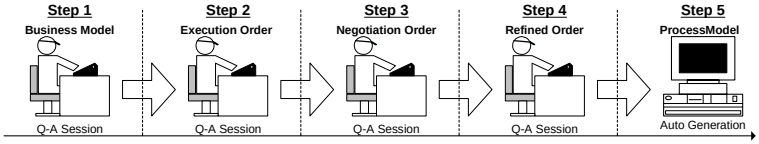

Fig. 11 Steps of the Designers Assistant

5.1 Step 1 - Business Model

In order to produce a business model the following four questions need to be answered. Answers according to the running case are given after every question.

1. *Who are the Agents?* Answers: Customer (Cust), Distributor (Dist), Chicken Supplier (Supp), Carrier (Carr)

2. *What are the Resources?* Answers: Money, Chicken, Delivery

3. *What are the Economic Events? Specify them by filling in the following table.*

Name of Economic event	Transferred Resource	From Agent	To Agent
DownPayToDist	DownPayment	Cust	Dist
FinalPayToDist	FinalPayment	Cust	Dist
ChickenToCust	Chicken	Dist	Cust
DownPayToSupp	DownPayment	Dist	Supp
FinalPayToSupp	FinalPayment	Dist	Supp
ChickenToDist	Chicken	Supp	Dist
DeliveryToDist	Delivery	Carr	Dist
PayToCarr	Payment	Dist	Carr

Table 1 Answers to question 3.

4. *Group the Economic Events into Dualities by filling in the following table.*

Economic event	Duality
DownPayToDist	Chicken Sales
FinalPayToDist	
ChickenToCust	
DownPayToSupp	Chicken Purchase
FinalPayToSupp	
ChickenToDist	
DeliveryToDist	Chicken Delivery
PayToCarr	

Table 2 The answers to question 4.

The answers to these four questions provide sufficient information to produce the business model shown in Fig. 2.

5.2 Step 2 – Execution Phase Order

Having identified the Economic Events, the designer is prompted to determine the dependency orders. In this step only flow and trust dependencies are considered.

5. Specify Flow *and* Trust Dependencies *by filling in the table below* (where the row and column headings are Economic Events identified in question 4). *If an EconomicEvent$_i$ (in row i) precedes an EconomicEvent$_j$ (in column j): put a '<' symbol in the corresponding cell (cell <i,j>). The '<' symbol is to be subscripted with 'f' or 't' depending on the type of dependency.*

	DPTD	FPTD	CTC	DPTS	FPTS	CTD	DTD	PTC
DownPayToDist (DPTD)			$<_{\mathbf{t}}$	$<_{\mathbf{f}(1)}$				
FinalPayToDist (FPTD)					$<_{\mathbf{f}(5)}$			$<_{\mathbf{f}(4)}$
ChickenToCust (CTC)		$<_{\mathbf{t}}$						
DownPayToSupp (DPTS)						$<_{\mathbf{t}}$		
FinalPayToSupp (FPTCS)								
ChickenToDist (CTD)			$<_{\mathbf{f}(2)}$		$<_{\mathbf{t}}$			
DeliveryToDist (DTD)			$<_{\mathbf{f}(3)}$					$<_{\mathbf{t}}$
PayToCarr (PTC)								

Table 3 Answers to the question 5 in the assistant.

The input from this step will be sufficient to roughly sketch the Execution phase in the BPMN diagram See the shaded area in Fig. 12 where for every Duality a pool is created. The numerical notes in the table are used to refer to the resulting sequence and message flows in the model. However, some dependencies, e.g. "$<_{f(1)}$" are later on overridden by refined orders and are then reduced from the final model.

5.3 Step 3 – Contract Negotiation Phase Order

After having gathered sufficient information to produce the BPMN diagram for the Execution phase, the analysis continues for the Contract Negotiation phase. As there are two ways for initiating a binary Business Collaboration according to the suggested collaboration patterns in Section 3, it is first necessary to identify which of these patterns to use for each binary collaboration.

6. For each binary Business Collaboration, ask whether
(a) a quotation already exists when the binary collaboration starts, or
(b) the binary collaboration is started by a partner requesting a quotation.

If the answer is (a) then the contract establishment collaboration pattern of Fig. 8 will be chosen. If the answer is (b) then the contract proposal collaboration pattern of Fig. 9 will be chosen.

Below, the answers to question 6 for the running case are given in bold.

6.1 ***(a)*** *Does a quotation already exist when the Cust-Dist collaboration starts, or*
(b) is the Cust-Dist collaboration started by a partner requesting a quotation?

6.2 (a) Does a quotation already exist when the Dist–Supp collaboration starts, or
***(b)** Is the Dist–Supp collaboration started by a partner requesting a quotation?*
6.3 (a) Does a quotation already exist when the Dist-Carr collaboration starts, or
***(b)** is the Dist-Carr collaboration started by a partner requesting a quotation?*

Note that abbreviated Agent names are used here in naming the collaborations above. The answers from this question are used to derive the beginning of each binary collaboration (see the white area in Fig. 12). We continue by identifying the negotiation and control dependencies.

7. Specify the Control *and* Negotiation Dependencies *by filling in the following table,* (where the row and column headings are pattern instantiations identified in questions 4 and 6). *If an Activity*[3] *(in row i) precedes an Activity (from column j) put a '$<_n$' symbol (for negotiation dependency) or a '$<_c$' symbol (for control dependency) in the corresponding cell (i.e. cell <i,j> in the table).*

Due to space restrictions and since the running case does not contain any control dependencies, we depict contract negotiation Activities only in Table 4.

	COCD	CECD	CRDS	CODS	CEDS	CRDC	CODC	CEDC
Contract-Offer:Cust-Distr (COCD)			**$<_{n(a)}$**			**$<_{n(b)}$**		
Contract-Accept/Reject:Cust-Distr (CECD)								
Contract-Request:Distr-Supp (CRDS)								
Contract-Offer:Distr-Supp (CODS)		**$<_{n(c)}$**						
Contract-Accept/Reject:Distr-Supp (CEDS)								
Contract-Request:Distr-Carr (CRDC)								
Contract-Offer: Distr-Carr (CODC)		**$<_{n(d)}$**						
Contract-Accept/Reject: Distr-Carr (CEDC)								

Table 4. Answers to the running case for question 7

Note, that the relationships within a binary collaboration are given by the process patterns and we have therefore crossed out the corresponding cells. The results from this question will give input for ordering of the activities from the Contract Negotiation phase across the binary collaborations. The alphabetical notes in the table refer to the resulting flows in the process model in Fig. 12.

5.4 Step 4 - Refined Order

In the first three steps, the Agents and Economic Events were identified. Furthermore, the activities in the Execution phase were ordered within and between binary Business Collaborations as well as the activities in the Contract Negotiation phase. In this step, we identify relationships that cross binary collaborations as well as Execution and Contract Negotiation phases.

8. *For each pair of Economic Events <EE_i, EE_j> (see Table 3), such that $EE_i <_f EE_j$: Is it required to perform EE_i before making a contract acceptance for EE_j, (i.e. a Contract Establishment between the Agents in EE_j)?*

[3] Formally the contents of the rows are *TP instances* (see Section 3.1), but for simplicity, we have referred to each TP instance by its first Activity.

The intuition behind this question is that an Agent may want to ensure that she has definite access to certain Resources before she is prepared to enter into a Contract for some product where these Resources are needed as input. It is possible to think about this question as a strengthening of a flow dependency – we say not only that EE_j cannot be performed before we have got the Resources from EE_i, but even that we are not prepared to enter into a Contract for EE_j before we have got the Resources from EE_i.

Below, the implementation of question 8 for the running case is given with answers.

8.1	*Must DownPayToDist be done before establishment of Dist-Supp Contract?*	**Yes**
8.2	*Must FinalPayToDist be done before establishment of Dist-Supp Contract?*	**No**
8.3	*Must FinalPaytToDist be done before establishment of Dist-Carr Contract?*	**No**
8.4	*Must ChickenToDist be done before establishment of Dist-Cust Contract?*	**No**
8.5	*Must DeliveryToDist be done before establishment of Dist-Cust Contract?*	**No**

9. *For each triple of Economic Events in table 3, EE_i, EE_j, EE_k, such that $EE_i <_t EE_j$ and $EE_k <_f EE_J$: Is it required to perform EE_i before making a contract acceptance for EE_k (i.e a Contract Establishment between the Agents in EE_k)?*

This question can be seen as a strengthening of a trust dependency. It says not only that we want to see another Agent perform EE_i before we perform EE_j, but that we want to see our partner to perform EE_i before we even start acquiring resources needed to perform EE_j.

Below, the implementation of question 9 for the running case with answers.

9.1	*Must DeliveryToDist be done before establishment of Cust-Dist Contract?*	**No**
9.2	*Must ChickenToDist be done before establishment of Cust-Dist Contract?*	**No**
9.3	*Must DownPayToDist be done before establishment of Dist-Supp Contract[4]?*	**No**
9.4	*Must DownPayToDist be done before establishment of Dist-Carr Contract?*	**Yes**

5.5 Business Process Generation

The final step of the proposed designers assistant is the generation of a BPMN diagram based on the answers from steps 1 – 4. This is achieved using the binary collaboration patterns introduced in Section 3 and a set of production rules to interconnect those instantiated binary collaborations into a multi-party collaboration. The set of production rules that are proposed can be categorized into four groups: *Rules for Binary Collaborations* (within a pool), *Rules for Inter-Collaborations* (between pools), *Reduction Rules*, and *Deadlock Prevention Rules*. However, due to space limitations, only the first two categories are summarized informally here.

Rules for Binary Collaborations

1. For each duality, introduce a binary collaboration contained in one pool. Such a collaboration will start with an instantiation of the BPMN diagram Contract Propose CP, (Fig. 9), if the answer to question 6 is a), otherwise the binary collaboration starts with an instantiation of the Contract Establish CP, (Fig. 8).
2. The BPMN diagram in a pool continues with instantiations of the Fig. 10 pattern, one for each Economic Event identified through question 5 in the designers assistant. These collaborations will initially be in parallel.

[4] Case 9.3 is already covered by case 8.1 and is only shown here for reasons of completeness.

3. For each trust dependency between two Economic Events, introduce a sequence flow between the corresponding execution activities.

Rule for Inter-Collaborations

1. For each flow dependency between two Economic Events, introduce a message flow between the corresponding execution activities.
2. For each control and negotiation dependency between two activities, introduce a message flow between these activities.
3. For each positive answer to questions 8 and 9, introduce a message flow between the relevant activities.

The BPMN diagram generated by these rules for the running case is shown in Fig. 12. (A formal definition of the generation via the production rules is found in Chapter 7 of [9])

6 Conclusions and Further Work

In this paper, we have proposed an approach for building process models on a declarative foundation. A starting point of the approach is that the process modeling task can be supported by gradually gathering domain knowledge, initially for the construction of business models and subsequently for their refinement and transformation into process models. The proposed designers assistant is structured on a division of e-commerce interactions into two phases: a Contract Negotiation Phase where a contract for exchanging economic resources is established; and an Execution Phase where the actual exchanges of the economic resources take place. We believe that this phase division provides an adequate starting point, but a topic for further work is to investigate more refined phase divisions, [6], [15].

The proposed approach is based on the concept of process patterns. A framework for representing process patterns is introduced, together with a number of basic process patterns. Two kinds of process patterns are identified: transaction patterns, basically capturing small communication chunks between two agents; and collaboration patterns, which are compositions of transaction patterns facilitating the representation of complex interactions. The value of this framework is not only that it provides an instrument for precise and unambiguous pattern definitions, but also that it gives a basis for motivating design choices in process modeling.

Finally, we also introduce the notion of action dependencies for capturing relationships between the activities within a process. Four kinds of dependencies are identified: flow, trust, control and negotiation dependencies. They can be stated declaratively, have a clear business motivation, and are used for the final derivation of a process model. A topic for further work is to investigate whether additional kinds of action dependencies are required.

A further line of future work is to examine the quality of the produced models, during which their completeness as well as their logical soundness should be investigated. While the work on completeness can primarily be done through empirical studies, the work on logical soundness can be supported by theoretical work like the one given in [5].

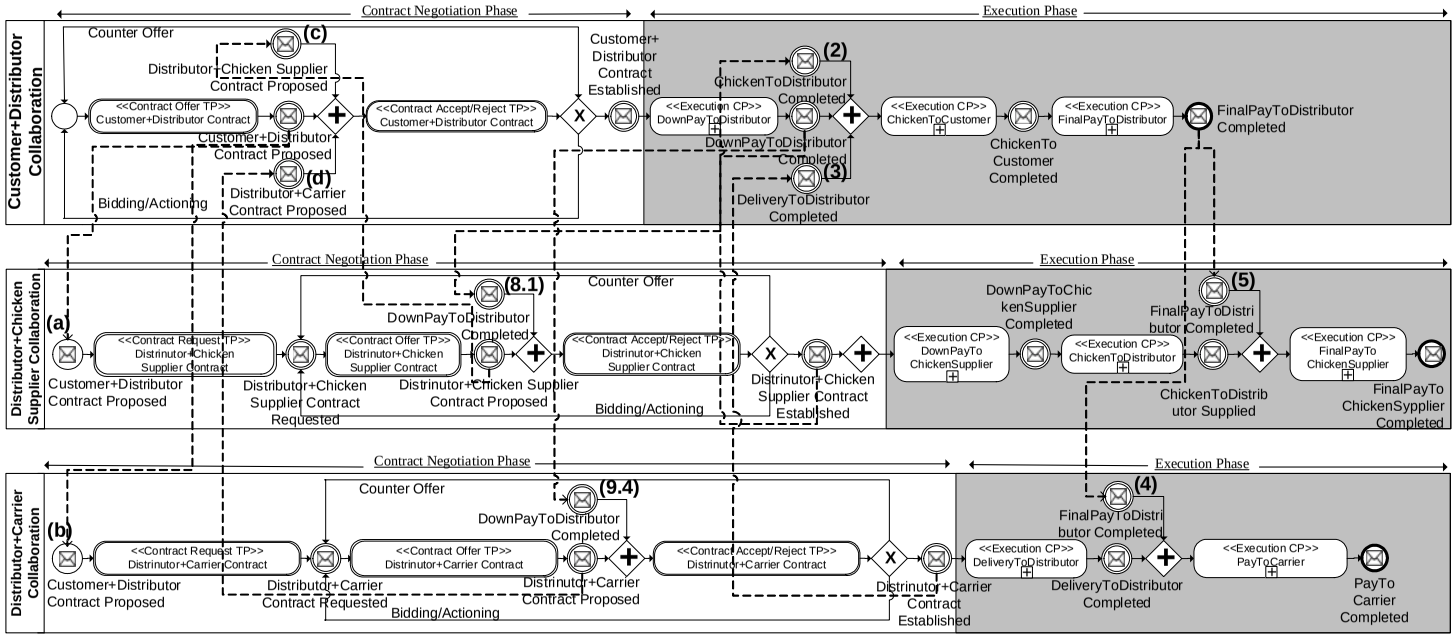

Fig. 12 Final BPNM diagram for Fried Chicken Case

References

1. Bergholtz M., Jayaweera P., Johannesson P., Wohed P., "Reconciling Physical, Communicative and Social/institutional Domains in Agent Oriented Information Systems – a Unified Framework", in Proc. of AOIS'03, held in conjunction with the *22nd, Int. Conference on Conceptual Modeling (ER'2003)*,Chicago, Illinois, USA
2. *Business Process Execution Language for Web Services*, OASIS WS-BPEL Technical Committee, Valid on 20040419, http://www.ebpml.org/bpel4ws.htm
3. *Business Process Management Initiative (BPMI)*, Valid on 20040419, http://www.bpmi.org/
4. *Business Process Modeling Notation (BPMN)*, http://www.bpmn.org/, Valid on 20040419
5. Chrzastowski-Wachtel P., Benatallah B., Hamadi R., O'Dell M., and Susanto A., "A Top-Down Petri Net-Based Approach for Dynamic Workflow Modeling", in Proc. of *Business Process Management Int. Conf.*, van der Aalst W., ter Hofstede A. and Weske M. (Eds.), LNCS 2678, pp.336-353, Springer, 2003
6. Dietz J., "Deriving Use Cases from Business Process Models", in Proc. of *22nd Int. Conference on Conceptual Modeling (ER'2003)*,Chicago, Illinois, USA. LNCS 2813, pp. 131-143, 2003
7. Gordijn J., Akkermans J. M. and Vliet J. C., "Business Modeling, is not Process Modeling", Proc. of the *1st Int. Workshop on Conceptual Modeling Approaches for e-Business (eCOMO'2000)*, at 19th Int. Conference on Conceptual Modeling (ER'2000), Salt Lake City, Utah, USA
8. Haugen B., Fletcher T., "*Multi-Party Electronic Business Transactions*", Valid on 20040419, http://www.ebpml.org/archive1.htm
9. Jayaweera P., "*A Unified Framework for e-Commerce Systems Development: Business Process Patterns Perspective*", PhD thesis, ISBN 91-7265-938-6, Valid on 20040924, http://www.dsv.su.se/~prasad/Publications/Thesis.pdf
10. Jayaweera P., Johannesson P., Wohed P., "Collaborative Process Patterns for e-Business", *ACM SIGGROUP Bulletin 2001/Vol 22, No. 2*
11. Larman C., "*Applying UML and patterns: an introduction to object oriented analysis and design*", ISBN 0-13-74880-7
12. Malone et al.: "*Towards a handbook of organizational processes*", MIT eBusiness Process Handbook, Valid on 20040419, http://ccs.mit.edu/21c/mgtsci/index.htm
13. McCarthy W. E., "*REA Enterprise Ontology*", Valid on 20040419, http://www.msu.edu/user/mccarth4/rea-ontology/
14. Moore S. A., "A foundation for flexible automated electronic communication", *Information Systems Research*, 12:1 (March 2001)
15. *Open-EDI phases with REA*, UN-Centre for Trade Facilitation and Electronic Business, Valid on 20040419, http://www.unece.org/cefact/docum/download/02bp_rea.doc
16. *The workflow patterns*, Valid on 20040419, http://tmitwww.tm.tue.nl/research/patterns/
17. *UN/CEFACT Modeling Methodology (UMM-N090 Revision 10)*, Valid on 20040419, http://webster.disa.org/cefact-groups/tmg/doc_bpwg.html
18. van der Aalst W.M.P., ter Hofstede A.H.M., Kiepuszewski B., and Barros A.P., "Workflow Patterns", *Distributed and Parallel Databases*, 14(3), pages 5-51, July 2003
19. van Dijk A., "Contracting Workflow and Protocol Patterns", in Proc. of Business Process Management Int. Conf., van der Aalst W., ter Hofstede A. and Weske M. (Eds.), LNCS 2678, pp.152-167, Springer, 2003