A Declarative Foundation of Process Models

Birger Andersson, Maria Bergholtz, Ananda Edirisuriya,
Tharaka Ilayperuma, and Paul Johannesson

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

Abstract. In this paper, a declarative foundation for process models is proposed. Three issues in process management and modeling are identified: business orientation, traceability, and flexibility. It is shown how these issues can be addressed by basing process models on business models, where a business model focuses on the transfer of value between agents. As a bridge between business models and process models, the notion of activity dependency model is introduced, which identifies, classifies, and relates activities needed for executing and coordinating value transfers.

1 Introduction

For information processing systems, it is possible to identify a number of functional aspects, [16]. The basic aspect is the services the system provides to its environment. By providing services, the system delivers value to its environment, which is the raison d'etre of the system. The behaviour, or process, aspect has to do with the ordering and control flow of services and activities. The communication aspect concerns interactions with other entities, like people, hardware, and software. Finally, the meaning aspect addresses the interpretation of symbols used in services. The process aspect is attracting ever more attention, as witnessed by the advent of new workflow and process management systems, new process modeling languages, and dedicated journals and conferences. Despite their success, process technologies and in particular process modeling techniques face a number of shortcomings and challenges that need to be addressed. In this paper, we will address three of these:

- Business Orientation. Process models are typically expressed through low level c\setlength{oncepts like control flow structures and message passing. Such concepts are not easily understood by business experts and users, who instead prefer to understand processes through business oriented notions like value exchanges and resource flows.
- Traceability. Constructing a process model includes taking a number of design decisions that affect the structure of the model. It should be possible to trace these design decisions back to explanations and motivations expressed in business terms.

- Flexibility. In most processes, the main structure is stable over time, while details may vary from case to case. Process models and systems should, therefore, allow for flexibility at design time as well as runtime.

One way to address these issues is to base process models on business models. By business model we mean a model which focuses on providing a high level view of the activities taking place by identifying agents, resources and the exchange of resources between the agents [12]. A process model, on the other hand, deals with operational and procedural aspects of business communication by focusing on the activities carried out by agents [1]. As the process model concentrates on technical details like *how* these activities will be carried out on the operational level, it becomes difficult for business users to understand it. Because of this it is hard for business experts and users to understand the connection between the high level view of the business model and the technical view of the process model.

The purpose of this paper is to propose a declarative foundation of process models based on business models. In order to bridge the gap between business models and process models, we introduce the notion of activity dependency model, which identifies, classifies, and relates activities needed for executing and coordinating value transfers. Having a declarative foundation will make it easier for designers to justify their design decisions at the technical level and trace them back to a business model. The paper builds upon and extends previous work, [1], by taking into account not only value transfer activities but also complimentary activities needed for their execution.

The paper is structured as follows. Section 2 gives an overview of related research. Section 3 gives an overview of business models. Section 4 introduces activity dependency models and shows how they are related to business models. Section 5 gives an overview of process models in a specific notation (BPMN [4]) and shows how they can be derived from activity dependency models. Finally, Section 6 concludes the paper and gives suggestions for further research.

2 Related Research

There exist several approaches to provide interfaces between technical concepts of software and systems design and business oriented requirements engineering. One of the most widely accepted approaches is the Unified Modeling Language, UML, with associated methodologies like RUP [13], where various types of design models represent static information in class diagrams as well as behavioral aspects modeled in interaction and activity diagrams. The UN/CEFACT Modeling Methodology (UMM) [15] uses UML as a base for specifying business processes involving information exchange in a technology-neutral and implementation-independent manner. A weak point in utilizing methodologies such as these, is still how to go from a business model to a process model in a systematic way. The DEMO methodology [5] was developed for the purpose of modeling essential business processes, abstracting completely from their realization. DEMO particularly stresses the distinction between information processes (generally modeled through low level message protocols) and the actual business processes, e.g. production acts where the agents fulfill the mission of the organization and coordination acts where agents enter into and comply with commitments. Dietz divides the collaboration between agents into three distinct 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 [8] identifies five phases: Planning, Identification, Negotiation, Actualization and Post-Actualization. In this paper, we use only two phases: a *Negotiation phase* in which commitments are proposed and accepted, and an *Execution phase* in which transfers of Resources between Agents occur and are acknowledged.

The approach proposed in this paper resembles DEMO since we also utilize the distinction between production processes and coordination processes. In our work this division among processes (or activities) is systematically derived from a business model through a set of activity dependencies, all expressed in business terms such as trust, commitments and transfer of value between agents [7] [11]. A similar approach may be found in the ***i**** framework [17], which has been explored for modeling trust relationships among strategic actors .The activity dependency models of this paper may be used as a modeling means to reason about design decisions, just as the models are in *i**. However, the specific aim of the activity dependency models here is to automate the process of going from business to process model.

In building process models, both an internal and external perspective on interactions between agents must be observed. In the Agent-Object-Relationship (AOR) approach [14], organizations and organizational behavior are modeled in two basic types of models: external and internal ones. An external AOR model adopts the perspective of an external observer who is observing agents and their interactions in the problem domain under consideration. In an internal AOR model, the internal (first-person) view of a particular agent to be modeled is taken. This notion is mirrored in our work, i.e. the activity dependency models introduced always take the view of one particular agent. Moreover, any interaction from one agent with respect to another agent has its counterpart in that other agent's activity model, i.e. one agent's internal activity is modeled as the other agent's external ditto. *PayInvoice* as an activity in one model has a reciprocal relationship with a *ReceivePayment* activity in the model of the corresponding agent. While AOR uses the internal business models as a point of departure to define so called interaction frames to describe the collaboration between agents, the work reported on in this paper suggests a way to automatically derive process models from business models. The proposed approach yields a process model that contains the main procedural logic. To get additional procedural details for sub-processes the notion of generic process patterns is used. The hypothesis is that most process models for organizational domains may be expressed as a combination of well documented design patterns [6] [9].

3 Business Models

The purpose of a business model is to describe and visualise the transfer of value between agents. A business model consists of three components: agents, value transfer offerings, and dualities. An agent is a person or (part of) an organisation that is capable of controlling, acquiring, and providing resources. Examples of agents are consumers, companies, and government authorities. An agent may transfer resources to another agent in a value transfer, e.g. the delivery of a product or a payment. A value transfer offering represents the willingness of an agent to perform value transfers with other agents. In a business setting, it never happens that one agent simply transfers a resource to another agent - she always expects to get another resource back as compensation. As the saying goes, "one good turn deserves another". To represent this reciprocity between value transfer offerings, we introduce the notion of duality. A duality associates two or more reciprocal value transfer offerings, e.g. the willingness to deliver a product and to pay for it.

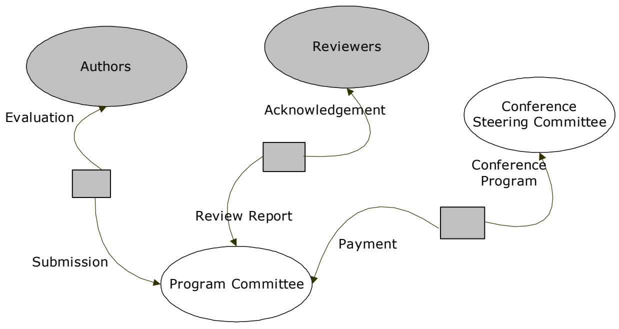

Fig. 1. A Business Model

An example of a simple business model for the well-known scientific conference case is shown in Fig. 1. For space restrictions, we only consider the submission and acceptance part of the case. Authors (agent) can submit papers (value transfer offering) to the program committee (agent), who in return (duality) provides the authors with evaluations and decisions for acceptance (value transfer offering). In order to make the decisions, the program committee obtains review reports (value transfer offering) from reviewers (agents), who in return (duality) will get acknowledgements in the conference documentation (value transfer offering). The program committee will based on submitted papers and reviews deliver a conference program (value transfer offering) to the conference steering committee (agent), who in return (duality) provides financial reimbursement (value transfer offering). In Fig. 3.1, individual agents are represented by plain ovals, classes of agents by shadowed ovals, dualities by shadowed small rectangles, and value transfer offerings by arrows between dualities and agents (either individual agents or classes of agents). More precisely, a business model is defined as follows.

Definition 3.1: A business model is a directed graph with three types of nodes *IndAgt, ClassAgt, Dual* and directed edges called *VTO* from *Dual* to (*IndAgt* $\cup$ *ClassAgt). IndAgt, ClassAgt* and *VTO* are sets representing Individual Agents, Class Agents and Value Transfer Offering between these Agents, respectively. *Dual* is a set of dualities where a duality d is a relation over $(IndAgt \cup ClassAgt) \times VTO \times (IndAgt \cup ClassAgt)$ with the following property: if $(x, y, z) \in d$ then $\exists w \in VTO$ such that $(z, w, x) \in d$ ■

4 Activity Dependency Models

4.1 Concepts and Notation for Activity Dependency Models

The purpose of an activity dependency model is to describe, on a high level, the activities needed for carrying out the value transfers specified in a business model. An activity dependency model provides more detail than a business model by identifying, classifying, and relating activities needed for executing and coordinating value transfers. On the other hand, an activity dependency model is less detailed than a process model, as it abstracts from ordering and control flow aspects. In this way, activity dependency models occupy a middle ground between business models and process models where they provide business-oriented information on activity coordination without going into procedural details. An activity dependency model is always constructed from a particular agent's perspective, called the base actor, i.e. the model takes the internal view as discussed in Section 2. This means that an activity dependency model focuses on one agent in a business model and the dualities involving this agent.

Structurally, an activity dependency model can be seen as a graph with four kinds of nodes, representing activities, and four kinds of directed edges, representing relationships between activities. The four kinds of activities are:

- *Value transfer activities.* A value transfer activity transfers resources from one agent to another and corresponds directly to the value transfer offerings in a business model.
- *Assignment activities.* An assignment activity relates a specific agent from a class of agents to the value transfer activities of one duality. An example is the assignment of a reviewer for reviewing a particular paper.
- *Production activities.* In a production activity, the base actor produces a resource required for a value transfer activity.
- *Coordination activities.* A coordination activity coordinates the value transfer activities within one duality as well as additional assignment and production activities.

The four kinds of relationships between activities are:

- *Duality dependencies.* A duality dependency from a coordination activity to a value transfer activity expresses that the latter is included in the duality of the former; recall that each coordination activity corresponds to one duality.
- *Flow dependencies.* A flow dependency, [10], from one activity to another expresses that the resource obtained by the first activity is needed as input to the second activity. An example is a retailer who has to obtain a product from an importer before delivering it to a customer.

- *Trust dependencies.* A trust dependency [1], between two value transfer activities within the same duality expresses that the first activity has to be carried out before the second one as a consequence of low trust between the involved 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 could be a car dealer requiring a down payment from a customer before delivering a car.
- *Trigger dependencies.* A trigger dependency from a coordination activity to an assignment or production activity expresses that the latter is to be initiated and managed by the coordination activity.

The components of an activity dependency model have a clear business motivation, i.e. they can be explained and motivated in business terms. This makes the activity dependency model a useful instrument for eliciting and communicating business knowledge. At the same time, an activity dependency model provides an adequate basis for constructing more detailed process models, as will be discussed in the next section.

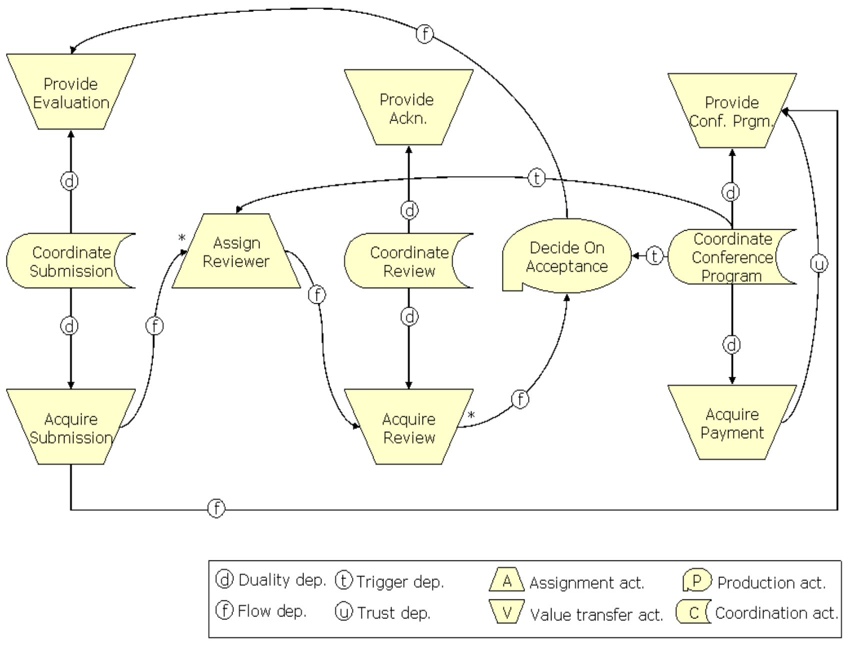

Fig. 2. An Activity Dependency Model

An example of an activity dependency model is shown in Fig. 2, which is based on the business model from the previous section. The base actor is the program committee, and the diagram shows three columns of coordination and value transfer activities corresponding to the dualities of this actor. There is one assignment activity *AssignReviewer* (for assigning a reviewer to a paper), one production activity *DecideOnAcceptance* (for deciding whether to select a paper), and a number of flow, trust, and trigger dependencies. For example, there is a flow dependency from *AssignReviewer* to *AcquireReview* meaning that an assignment of a reviewer to a paper must exist before a review of that paper can be obtained. Furthermore, cardinalities (as in UML) have been attached to some of the dependencies. For example, the star on the flow dependency from *AcquireSubmission* to *AssignReviewer* means that for each submitted paper, several assignments of reviewers may exist.

Definition 4.1: An activity dependency model is a directed graph with four types of nodes *VTA, Coord, Ass, Prod* and four types of directed edges *Dual, Flow, Trust,* and *Trigger*; where *VTA, Coord, Ass,* and *Prod* are sets representing value transfer activities, coordination activities, assignment activities, and production activities, respectively and *Dual, Flow, Trust* and *Trigger* are relations such that:

- *Dual* is an injective relation over $Coord \times VTA$.
- *Flow* is a relation over $(VTA \cup Ass \cup Prod) \times (VTA \cup Ass \cup Prod) \times \{1, *\} \times \{1, *\}$ such that, if $(x, y, z, w) \in Flow$ then $(x, y) \in \{(VTA \cup Ass \cup Prod) \times (VTA \cup Ass \cup Prod) \setminus (Ass \times (Ass \cup Prod))\}$ and $(z, w) \in (\{1,*\} \times \{1,*\})$
- *Trust* is a relation over $VTA \times VTA$ within the same duality
- *Trigger* is a relation over $Coord \times (Ass \cup Prod$ ■

4.2 From Business Model to Activity Dependency Model

An activity dependency model can partially be derived from the business model it is based on. Each duality in the business model gives rise to one coordination activity, and each value transfer in the business model gives rise to one value transfer activity. The value transfer activities within one duality are related to the corresponding coordination activity by duality dependencies. Furthermore, for dualities where the base actor may choose between several agents in a class of agents (for example choosing a reviewer for a paper), an assignment activity is added. For each duality, one production activity may also be added when the base actor has to produce some resource needed for a value transfer activity. These components of the activity dependency model are directly derived from the business model, but the activity dependency model will also contain a number of additional components not derivable from the business model, namely the flow, trust, and trigger dependencies.

Definition 4.2: Let BM = <*IndAgt, ClassAgt, Dual, VTO*> be a business model. Let AM = <*VTA, Coord, Ass, Prod, Dual, Flow, Trust, Trigger*> be an activity dependency model. AM *conforms to* BM if

- for each duality in *Dual* there is one coordination activity in *Coord*
- for each value transfer in *VTO* there is one value transfer activity in *VTA*
- for each element *d* = <*x, y, z*> in a duality in BM-*Dual* there is a pair <*c, v*> in AM-*Dual*, where *c* is the coordination activity corresponding to *d* and *v* is the value transfer activity corresponding to *y*, and a pair <*c, w*> where c is the same coordination activity and *w* is a value transfer activity corresponding to *z*.

- for each *d* in *Dual*, there is optionally one production activity in *Prod*
- for each *d* in *Dual* related to an element in *ClassAgt*, there is one assignment activity in *As* ■

5 Process Models

5.1 Concepts and Notation for 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 an 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 specifications such as BPEL4WS [2].

In this paper, we will use only a selected set of core elements from BPMN. 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 diagram, an Activity is represented by a rounded rectangle. An Activity can be atomic or compound. A Compound Activity is composed of other Activities and will be marked by a '+' sign inside the rounded rectangle. An Activity may also be repeated, which is graphically shown by a circular arrow inside the 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 by Sequence Flows, shown as arrows, indicating the order in which Activities will be performed in a business 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 shown as a diamond with an 'X' or '+', respectively. Pools are graphical constructs, in the form of oblong rectangles enclosing other BPMN elements, for separating different sets of activities from each other. Message flows, shown as dotted arrows, are used for communication between Activities in different Pools. An example of a BPMN process diagram is given in Fig. 3.

5.2 From Activity Dependency Model to Process Model

Moving from an activity dependency model to a process model is essentially about specifying the detailed control flow between activities. The starting point is to let each coordination activity in the activity dependency model become a process defined within one pool. This means that each pool models the exchange of resources between the base actor and one other actor as well as the assignment and production activities needed for this exchange. The entire process model will consist of a number of such processes within pools that communicate with each other.

A single pool essentially describes a binary collaboration between two partners. Such a collaboration may consist of several phases as discussed in Section 2, including planning, identification, negotiation, actualisation, and post-actualisation. In this paper, we consider only two phases: one Negotiation phase in which commitments for resource exchanges are proposed and accepted, and one Execution phase in which resource transfers occur and are acknowledged. Typically, a process will contain both a Negotiation phase and an Execution phase, but in some cases only the Execution phase is included. If the Negotiation phase is included, one sub-process for this phase is added to the pool. Thereafter, for each value transfer activity associated to the coordination activity, one sub-process is added. Furthermore, if there are any trigger dependencies from the coordination activity to assignment or production activities, one sub-process is added for each related assignment or production activity. There are also additional sub-processes for acquiring relevant resources, as specified through flow dependencies. Some of the sub-processes within a pool will be repeating. This occurs when there is a multi-valued flow dependency to the activity corresponding to the sub-process. The sub-processes are related by adding sequence flows between pairs of sub-processes if there is a flow or trust dependency between the corresponding activities. Finally, the negotiation sub-process is related to the other sub-processes by an AND-gateway.

As each coordination activity gives rise to its own pool, we will end up with a number of processes in pools that have to be connected to each other via message flows. Two pools need to be connected if one of the pools requires a resource provided by the other. There will be one message flow from the resource providing pool to the resource requesting pool, which informs about the delivery of the resource. Furthermore, there will be one message flow from the resource requesting pool to the resource providing if the latter contains a negotiation phase; this message flow is the request for the resource.

An example of a process based on the coordination activity *Coordinate ConferenceProgram* in Fig. 2 is shown in Fig. 3. It is assumed that this process does not contain a Negotiation phase but only an Execution phase (for reasons of completeness we also include the processes based on the other two coordination activities from Fig. 2 *Coordinate Review* and *Coordinate Submission* in Fig. 3, where *Coordinate Review* is assumed to contain a Negotiation phase) . The two value transfer activities associated to *CoordinateConferenceProgram* give rise to two sub-processes *ProvideConferenceProgramme* and *AcquirePayment.* These and other sub-processes may have a more or less complex internal structure, and we will return to this issue in Section 5.3. Two more sub-processes are added, *AssignReviewer* and *DecideOnAcceptance*, based on the trigger dependencies in the activity dependency model. Furthermore, sub-processes that acquire relevant resources need to be added, in this case *ReceiveSubmission* and *GetReview*. *ReceiveSubmission* is added as there is a flow dependency from *AcquireSubmission* to *ProvideConferenceProgram* in the activity dependency model. *GetReview* is added as there is a flow dependency from *AcquireReview* to *DecideOnAcceptance*. Finally, a sub-process *DeliverEvaluation* is added due to the flow dependency from *DecideOnAcceptance* to *ProvideEvaluation*. The sequence flows in the diagram are all derived from flow dependencies, except the one from *AcquirePayment* to *ProvideConferenceProgram*, which is based on a trust dependency. Fig. 3 shows the entire process model for the conference case with message flows included.

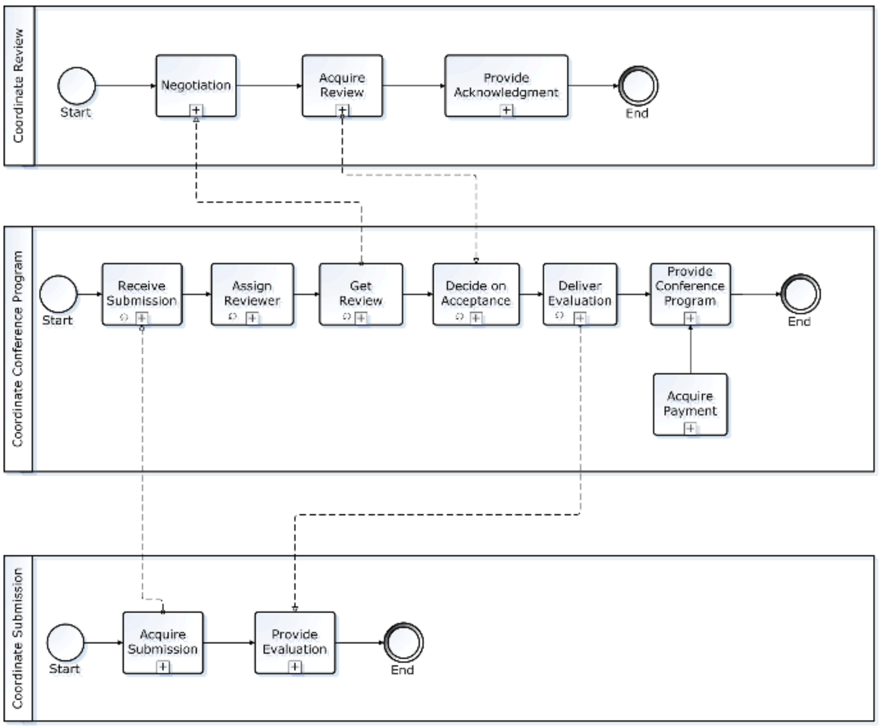

Fig. 3. A Process Model in BPMN

Definition 5.1: Let *P* be a pool containing an activity *A*. We will denote that activity by *P.A*. ■

Definition 5.2: Let AM = <*VTA, Coord, Ass, Prod, Dual, Flow, Trust, Trigger*> be an activity dependency model. A coordination activity *C* in *Coord* will be related through duality dependencies to one or several value transfer activities in which the base actor provides resources to another actor. These value transfer activities will be denoted $Out_1, \ldots, Out_n$. Analogously, the value transfer activities where the base actor acquires resources from another actor will be denoted $In_1, \ldots, In_n$. ■

Definition 5.3: Let AM = <*VTA, Coord, Ass, Prod, Dual, Flow, Trust, Trigger*> be an activity dependency model. Let PM be a BPMN diagram. PM *conforms to* AM if for each coordination activity *C* in *Coord* there is one pool P_C in PM fulfilling the following conditions:

- P_C contains optionally one sub-process entitled *Negotiation*
- For each value transfer activity In_i, there is one sub-process In_i
- For each value transfer activity Out_i, there is one sub-process Out_i
- For each flow dependency from an activity $D.In_i$ (where *D* is a coordination activity $\neq C$) to a value transfer activity Out_j, there is one subprocess $get(D.\ In_i)$.

The sub-process is repeating if the flow dependency is not injective. There is one message flow from P_D. In_i to *get(D.* In_i*)*. If P_D contains a sub-process *Negotiation*, then there is also one message flow from *get(D.*In_i*)* to P_D.*Negotiation*.

- For each trigger dependency from *C* to an assignment or production activity *Act*, there is one sub-process *do(Act)*. The sub-process is repeating if there is a flow dependency from a value transfer activity *D.*In_i to *Act* and *get(D.* In_i*)* is included in P_C and repeating, or if there is a flow dependency from *Act* to *C.*Out_ithat is multi-valued.
- For each flow dependency from an activity *Act* to a production activity *Prod* such that the corresponding sub-process *do(Prod)* is included in P_C, there is one sub-process *do(Act)*. The sub-process is repeating if *do(Prod)* is repeating, or if the flow dependency is not injective.
- If there is a production activity *Prod* with a flow dependency to a value transfer activity *D.* Out_i and the sub-process *do(Prod)* corresponding to *Prod* is included in P_C, then there is a message flow from *do(Prod)* to *D.* Out_i.

There is a sequence flow between two sub-processes if there is a flow dependency or trust dependency between the corresponding activities. When all sub-processes have been linked in this way, there will be a number of leftmost sub-processes, $L_1, \ldots ,L_m$, i.e. sub-processes with no incoming sequence flow. There will be a gateway *G* in P_C with a sequence flow from *Negotiation* to *G*. There will be sequence flows from *G* to each of $L_1, \ldots ,L_m$.

Finally, if a process model PM conforms to an activity dependency model AM and PM´ is derived from PM as specified below, then PM´ conforms to AM. PM should contain two repeating sub-processes *S* and *T* joined by a sequence flow. PM´ is derived from PM by replacing the repeating sub-processes and their joining sequence flow by one repeating sub-process containing *S* and *T* joined by a sequence flow. ■

The reason for the last paragraph of the definition is to allow for more flexibility in the process design. Without it, there would be an unwanted restriction, namely how repeating activities are treated. It would be assumed that if there are two subsequent repeating activities then there must be a repetition over the first activity followed by a repetition over the second. For example, in Fig. 3 all submissions have to be received before any assignment is made. However, this assumption is too restrictive in many

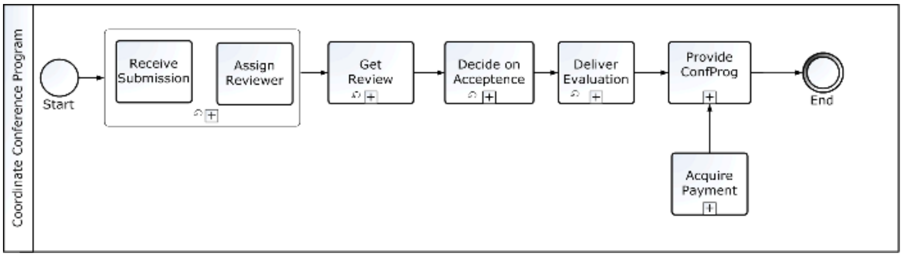

Fig. 4. An alternative Process Model in BPMN

cases, as an alternative is to have a single repetition over a sequence of the two activities. For example, in Fig. 4 a received submission may be directly followed by an assignment.

Due to space limitations we have only considered the success path, or "happy path", of a process, i.e. we have not included exceptions, failed negotiations, etc. Extensions for these possibilities are obviously needed, but they are standard and do not affect the basic relationships between activity dependency models and process models.

5.3 Process Patterns

A process model derived from an activity dependency model, as described above, contains only the main procedural logic. Additional procedural details for the sub-processes of the process model are needed. One approach is to treat sub-models as the main model, i.e. detailed views of the main business model is provided and sub-activities and sub-processes derived according to the transformation described in the previous section. We believe, however, that a more flexible way to provide the procedural details is to use the notion of process patterns. The idea is that each sub-process is based upon a generic process pattern.

Generic Process Patterns

UN/CEFACT has defined a number of business transaction patterns as part of UMM with the intention of providing an established semantics of frequently occurring business interactions. Individual sub processes such as, for example, negotiations and fulfillments of earlier commitments to perform value transfers are easily expressed or assembled on an arbitrary level of granularity through the usage of generic transaction patterns. Below we list a number of patterns from [15] and [1].

Negotiation Phase Patterns

The *Contract proposal* [11] transaction pattern models the non-legally binding negotiation phase in a contract formation, whereas the *Commercial (Offer-Accept)* [1515] tra nsaction pattern expresses the formal creation phase of a contract. These patterns may be assembled into more complex patterns of collaborations between partners. An example of the latter is the UMM "simple negotiation pattern", which combines variants of proposals and offers for contracts of transfer of value resources. The assembling of patterns is based on a layered approach where each base pattern constitute a sub-process in the assembled pattern, and where the "happy path" sequence flow of each pattern serves as connector between the pattern.

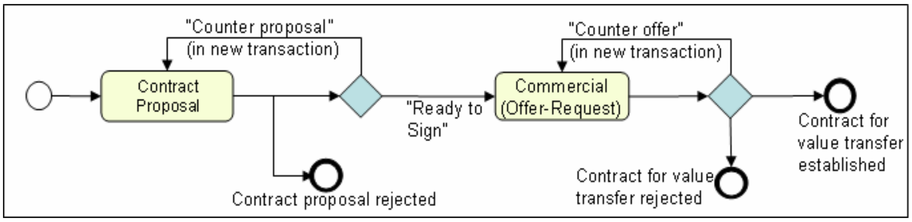
"Counter proposal"
(in new transaction)
Contract Proposal
Contract proposal rejected
"Ready to Sign"
"Counter offer"
(in new transaction)
Commercial (Offer-Request)
Contract for value transfer rejected
Contract for value transfer established

Fig. 5. Simple Negotiation pattern [15]

Execution Phase Patterns

The fulfillment pattern specifies the completion of a Value transfer between two partners. The Bi- and Unilateral Cancellation transaction patterns refer to the unilateral or bilateral cancellation of a contract or to commitment(s) within a contract for value transfers.

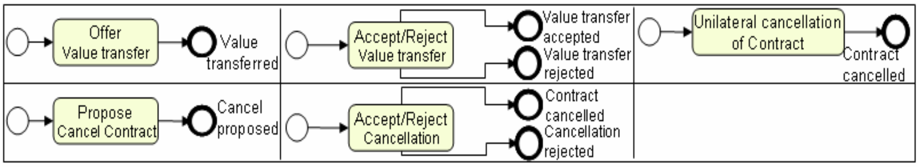

Fig. 6. Fulfillment, Fulfillment Accept/Reject, bilateral and unilateral cancellation transaction patterns [1]

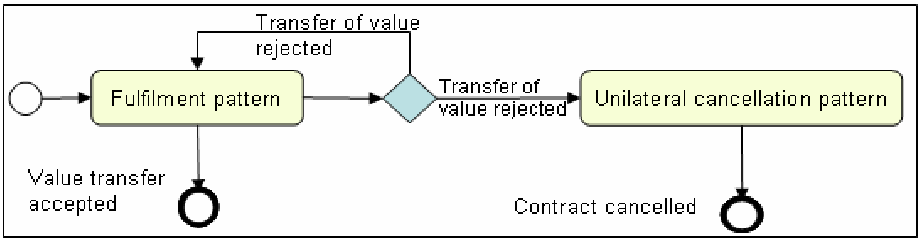
Transfer of value rejected
Fulfilment pattern
Transfer of value rejected
Unilateral cancellation pattern
Value transfer accepted
Contract cancelled

Fig. 7. Fulfillment collaboration pattern [1]

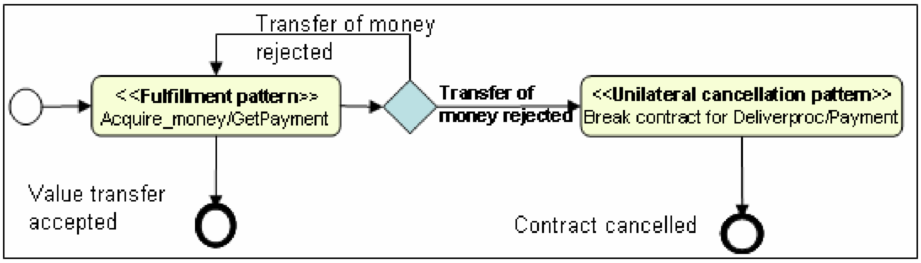
Transfer of money rejected
<<Fulfillment pattern>> Acquire_money/GetPayment
Transfer of money rejected
<<Unilateral cancellation pattern>> Break contract for Deliverproc/Payment
Value transfer accepted
Contract cancelled

Fig. 8. Instantiation of Fulfillment collaboration pattern

The fulfillment collaboration pattern specifies relevant transaction patterns (Fig. 7) and the rules for transitioning among these within the completion of a value transfer. The pattern is assembled from the Fulfillment and Unilateral Cancellation transaction patterns defined in the previous section.

Finally, as an example of an instantiation of a generic collaboration pattern, consider the case of sub-process "Get-payment" from the Conference case, where the procedural details may be defined in a flexible way through combinations of generic transactions patterns. In fig 8 the "Get-payment" sub-process is modeled through the

Fulfillment collaboration pattern, which specifies relevant transaction patterns and the rules (variants of the rules for when an how to break a contract of value transfer amounts to choosing the right combination of patterns) for transitioning among these within the completion of a value transfer.

6 Conclusions and Further Work

In this paper, we have proposed a declarative foundation of process models based on business models. A key element of the approach is the activity dependency model, which works as a bridge between a business and a process model. We believe that this approach effectively addresses the issues introduced in Section 1:

- Business Orientation. Instead of going directly into procedural details, an activity dependency model allows business experts and users to describe the underlying business reasons that govern the flow of processes. In particular, relations between activities can be specified in terms of notions like resource flow, trust, coordination, and reciprocity.
- Traceability. A process model is based on an activity dependency model, which in turn is based on a business model. This means that components in a process model can be explained by and tracked back to business oriented notions and motivations.
- Flexibility. The transformations from business model to activity dependency model to process model give the main structure of a process. However, the approach allows for flexibility by letting sub-processes be based on patterns. This means that the lower-level details of a process model can be tailored to the situation at hand by selecting appropriate patterns from a repository.

We have introduced the notion of activity dependencies for capturing relationships between the activities within a process. Four kinds of dependencies have been identified, flow, trust, trigger, and duality dependencies. These can be stated declaratively, have a clear business motivation, and are used for the construction of the process model. A topic for further work is to investigate whether additional kinds of activity dependencies are required. Another topic for further research is to study how to include more phases in the process models, in addition to the negotiation and execution phases.

References

1. Bergholtz M., Jayaweera P., Johannesson P., Wohed P., "A pattern and dependency based approach to the design of process models", in Proc. of the 23rd International Conference on Conceptual Modeling(ER2004), Shanghai, China
2. Business Process Execution Language for Web Services, OASIS WS-BPEL Technical Committee, Valid on 20041115, *http://www.ebpml.org/bpel4ws.html*
3. Business Process Management Initiative (BPMI), Valid on 20041115, *http:// www.bpmi.org/*

4. Business Process Modelling Notation (BPMN), Valid on 20041115, *http:// www.bpmn.org/*
5. Dietz J.L.G, Deriving "Use Cases from Business Process Models", ER2003, LNCS2813, pp.131-143, L-Y Song et al. (Eds.), Springer-Verlag Berlin Heidelberg 2003
6. Gamma, E., Helm, R., Johnson, R.,Vlissides, J.: *Design Patterns,* Addison-Wesley, 1995
7. Gordijn J., Akkermans J. M. and Vliet J. C., "Business Modelling, is not Process Modelling", Proc. of the 1th International Workshop on Conceptual Modeling Approaches for e-Business (eCOMO'2000), held in conjunction with the 19th International Conference on Conceptual Modeling (ER'2000), Salt Lake City, Utah, USA
8. *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
9. Larman, C., "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development", 3rd Edition, 2004, ISBN: 0131489062
10. Malone et al.: "Towards a handbook of organizational processes", MIT eBusiness Process Handbook, Valid on 20040419, http://ccs.mit.edu/21c/mgtsci/index.htm
11. Osterwalder, A, Parent, C., and Pigneur, Y., "Setting up an ontology of business model", CAISE/EMOI'2004 (INTEROP workshop), 2004
12. McCarthy W. E., "REA Enterprise Ontology", Valid on 20040419, http://www.msu.edu/ user/mccarth4/rea-ontology/
13. Rational Unified Process (RUP) *Valid on 20041115,* http://www-306.ibm.com/ software/awdtools/rup/
14. Wagner, G.,. "The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior". *Information Systems* 28:5 (2003)
15. UN/CEFACT Modeling Methodology (UMM-N090 Revision 10), Valid on 20040419, http://webster.disa.org/cefact-groups/tmg/doc_bpwg.html
16. Weiringa, R., Blanken, H., Fokkinga, M. , Grefen, P., Aligning Application Architecture to Business Context, Caise 2003, LNCS 2681 pp. 209- 225
17. Yu, E., Liu, L., "Modelling Trust in the *i** Strategic Actors Framework" Proceedings of the 3rd Workshop on Deception, Fraud and Trust in Agent Societies. Barcelona, Catalonia, Spain (at Agents2000), June 3-4, 2000