Representing and Validating Digital Business Processes

Lianne Bodenstaff[1], Paolo Ceravolo[2], Ernesto Damiani[2], Cristiano Fugazza[2], Karl Reed[3], and Andreas Wombacher[4]

[1] Information Systems Group, Dept. of Computer Science,
University of Twente (Nl)
l.bodenstaff@utwente.nl
[2] Dept. of Information Technologies,
Università degli Studi, Milan (It)
{ceravolo,damiani,fugazza}@dti.unimi.it
[3] Computer Science Dept.,
LaTrobe University, Melbourne, (Aus)
k.reed@latrobe.edu.au
[4] School of Computer and Communication Sciences,
École Polytechnique Fédérale de Lausanne (Ch)
andreas.wombacher@epfl.ch

1 Introduction

Today, the term *extended enterprise* (EE) is typically meant to designate any collection of organizations sharing a common set of goals. In this broad sense, an enterprise can be a whole corporation, a government organization, or a network of geographically distributed entities. EE applications support digitalization of traditional business processes, adding new processes enabled by e-business technologies (e.g. large scale Customer Relationship Management). Often, they span company boundaries, describing a network of relationships between not only a company and its employees, but also partners, customers, suppliers, and markets. In this scenario, *Business Process Modeling* (BPM) techniques are becoming increasingly important. Less than a decade ago, BPM was known as *workflow design* and was aimed at describing human-based processes within a corporate department. Today, BPM is used to design the orchestration mechanisms driving the interaction of complex systems, including communication with processes defined and executed by third parties according to well-defined protocols. Also, it can be used to check compatibility and consistency of the individual business processes that are defined by collaborating business entities. A large number of methodologies, languages, and software tools have been proposed to support *digital* BPM; nonetheless, much work remains to be done for assessing a business process model validity with respect to an existing organizational structure or w.r.t. external constraints, like the ones imposed by security compliance regulations. In particular, Web-based business coalitions and other inter-organizational transactions pose a number of research problems. OMG's *Model Driven Architecture* (MDA) [Aßmann et al., 2005] provides a framework

T.S. Dillon et al. (Eds.): Advances in Web Semantics I, LNCS 4891, pp. 219–246, 2008.

for representing processes at different levels of abstraction. In this paper we rely on a MDA-driven notion of business process model, constituted by three distinct components:

- A *static domain model*, including the domain entities (actors, resources, etc.);
- a *workflow model*, providing a specification of process activities;
- a *value model*, describing the value exchange between parties.

In the modeling of static models, we shall focus on expressive formalisms constituted by controlled fragments of natural languages, introducing their translation into logics-based static domain models, and describing their relations with Semantic Web (SW) metadata formats [W3C]. In fact, the latter allow to assign a specific semantics to entities in the domain model; particularly, we are interested in the entities that, for a number of reasons, may result in underspecified descriptions. This distinction will prove of foremost importance with regard to the derivation of implicit knowledge. The static model can also be used to provide a comprehensive description of the entities that interact with each other in the workflow model and the resources that are exchanged during workflow execution. Visual languages are typically used to produce business process descriptions that regulate the interaction between the different actors in the EE scenario. Logic-based formalismsmodels can be easily derived from business process descriptions and can therefore be integrated with the static model for checking consistency and computing a wide range of business metrics that take into account dynamic aspects of the business environment. Finally, although the process model represents the main source of information driving the actual implementation of business activities, it may not be the focus for business analysts that are required to evaluate the net outcome of transactions in terms of the value exchange between the interacting parties. Consequently, the overall model will be completed by providing the *value model* underlying business processes. As for process models, these are also typically expressed by means of visual languages and can be translated into logic-based to obtain data structures that are amenable to automated processing.

The Chapter is structured as follows: in Section 2, we introduce rule-based business modeling and its translation into a logic-based formalism. Section 2.2 addresses two distinct semantics that can be applied to the knowledge base that is derived from business rules, highlighting the need for integration of both paradigms into a *hybrid* deduction system. Furthermore, Section 2.4 introduces the issues related with the different *modal* interpretations of business rules that are required. Section 3 provides an overview of formalisms for modeling process workflows and then focuses on a practical example of BPMN diagram describing the orchestration of independent processes. Section 4 completes the picture with a value model to be associated with the entities that have been introduced in the static model and have been instantiated in the workflow model in order to define processes. Section 5 is addressing the relations between the three models that have been introduced by indicating some of the possible cross-checking mechanisms that can bind the distinct layers in the actual implementation. Finally, Section 6 draws the conclusions and highlights the main open issues.

2 Rule-Based Structural Description

This Section introduces the *business rules* (BR) approach to business domain modeling [Ross, 2003]. BR allows for a thorough specification of the entities that populate a specific state of affairs and the mutual relations between them. The high expressivity that is required by rules has led business analysts toward the adoption of natural language as the encoding formalism for BR. This strategy clearly fulfils the needs of knowledge sharing between humans, but inevitably complicates any sort of automated processing on rules. A tradeoff between expressivity and formal specification of statements is constituted by *controlled natural languages* and, among them, *controlled English* (CE): these formalizations are derived from natural languages by constraining the admissible sentential forms to a subset that is both unambiguous and expressive. A widely acknowledged example of CE is the *Attempto Controlled English* (ACE) [Fuchs et al., 1999], a general-purpose controlled natural language supporting specification of complex data structures, such as ontologies. As an example, a simple rule in ACE that may contribute to the definition of a business domain is the following:

A customer provides a credit card to a retailer. (1)

General-purpose controlled languages can be provided with a formal (e.g., logics-based) semantics; however, they fall short of being capable to model all the aspects of a business domain. With regard to the expressive power required by BR, rule-based languages may need to cover higher order logics and also, as explained in Section 2.4, may specify the modal interpretation to be associated with a statement. Also, as we discussed in Section 3, the static description provided by BR needs to integrate with process descriptions and, possibly, originate object-oriented data structures that software developers may use to flesh out applications. The recognition of these requirements was a major driver of OMG's *Semantics of Business Vocabulary and Business Rules* (SBVR) proposal [OMG, 2006], aimed at specifying a business semantics definition layer on top of its software-oriented layers. In the OMG vision, BR can then be integrated with the development process represented by OMG's own *Model Driven Architecture* (MDA) and, consequently, extend the applicability of object-oriented modeling not only to software product development but also to business process modeling and maintenance. SBVR provides business analysts with a very general controlled language, whose syntax visually separates the different tokens in a sentence (nouns, verbs, reserved keywords) with different styles and colors[1]. As an example, the rule in (1) corresponds to the following SBVR *fact type*.

Obligaton: a customer *provides* a credit card to a retailer (2)

[1] Here we shall not deal with *color markup* of rules, which is primarily intended to ease the reading of a large rule corpus.

For the sake of clarity, in the remainder of this Section we are not going to stick to any specific Controlled English (CE) formalism for expressing rules. However, we stress the importance of carefully evaluating the expressivity of candidate CE languages, prior to encoding business intelligence into one of these formalisms, because it may not necessarily meet the requirements of more comprehensive frameworks for corporate data reuse. ACE and SBVR represent only two of the many available CE formalisms, which may vary according to *i)* the syntactic restrictions that are applied to the corresponding natural language, *ii)* the fragment of *first-order logic* (FOL) that can be conveyed by statements, and *iii)* the applicability of automated reasoning. The reader can refer to [CLT] for a more complete survey of controlled natural language formalisms. Here, we investigate the feasibility of automated deductions over business rules, particularly in the EE scenario where independent business entities are required to integrate.

2.1 Formal Grounding of Business Rules

By using CE formalisms for expressing BR, it is possible to apply translation mechanisms that lead to a univocal logic formulation of statements. As an example, the grounding in formal logic provided by ACE is constituted by Discourse Representation Structures (DRS) [Fuchs and Schwertel, 2003] that represent a subset of FOL and provide a pathway to executable logic formulations [2]. More importantly, an Attempto Parsing Engine (APE) is available either as a standard Web interface and as a webservice, so that the engine can be remotely queried by programming logic developed by third parties. As an example, the DRS corresponding to the simple rule in (1) is the following:

$$[A, B, C, D]$$
$$object(A, atomic, customer, person, cardinality, count\_unit, eq, 1) - 1$$
$$object(B, atomic, credit\_card, object, cardinality, count\_unit, eq, 1) - 1$$
$$object(C, atomic, retailer, person, cardinality, count\_unit, eq, 1) - 1$$
$$predicate(D, unspecified, provide\_to, A, B, C) - 1$$

APE also provides a mapping between a subset of ACE and the OWL DL ontology language [W3C, 2004]; it can therefore take advantage of DL *reasoners* [Haarslev and Moller, 2001, Parsia et al., 2003] to infer implied knowledge. As will be shown in the following of this Section, DL represents only a small fragment of FOL; particularly, it is also limited to expressing binary relations between entities. As a consequence of this, even the simple ternary relation binding *customers*, *resellers*, and *credit cards* in (1) cannot be expressed without "objectifying" the relation by means of a newly introduced concept definition. This amounts to expressing predicate D in the corresponding DRS as a concept

[2] Recall that full FOL is proven to be undecidable; therefore, deriving the DRS corresponding to an ACE statement does not imply that such logic formulation can also be executed by programs.

definition that is the domain of three binary relations whose ranges (i.e., the categories of entities the relations map to) are concepts *customer*, *reseller*, and *credit card*, respectively. For a more traditional processing of ACE rules, DRS can also be translated into RuleML [Boley et al., 2001] to be processed by *rule engines*, such as the Jena framework [Jena]. Note that, in this case, the term "rule" is not indicating a BR, but instead the Horn fragment of FOL which guarantees a sound and complete reasoning on rules by applying either forward- or backward-chaining derivations. One of the main challenges of drawing inferences based on a BR model is the capability of applying the so-called *hybrid reasoning* on the knowledge base. The distinct inference paradigms which are associated, respectively, with OWL DL reasoning and Horn rules execution need to be integrated. It is not possible to adopt a single inference technique, because the entities in the business domain may have a different semantics associated with them. Information under full control of the stakeholder (e.g., a company) writing the model (e.g., the notion of *employee*) can be modeled as in traditional database design. In this case, BR simply provide a lingua franca by means of which business analysts and software developers can more easily translate company data requirements into real-world implementations. Other knowledge, however, needs to be introduced in order to compete and cooperate in the EE scenario (e.g., the notion of *competitor*); this knowledge is not under the modeler's full control, and may therefore be incomplete[3]. We indicate by *closed-world assumption* (CWA) the approach implemented by applications that only process complete data under the full control of their owner. In this case, failing to retrieve answers to a query (say, 'retrieve the credit card data associated with customer John Smith') automatically implies that such data do not exist. Consequently, customer John Smith constitutes a valid answer to the query 'retrieve customers that do **not** have a credit card associated with them' because incomplete knowledge amounts to false (i.e., negative) knowledge. This notion of negation (generally referred to as *negation as failure*) leads to the *non-monotonic reasoning* that provides the correct interpretation of closed systems. In the context of logic inference, the term 'non-monotonic' essentially means that conclusions (e.g., that *John Smith* is a valid answer to the previously defined query) may be contradicted by adding information to the knowledge base (e.g., the assertion John Smith provides VISA-041). On the contrary, we indicate by *open-world assumption* (OWA) the monotonic approach to inference that should be applied to heterogeneous data sources, such as those collected by individual systems in the EE scenario, and also (according to business analysts) to proprietary descriptions expressed by business rules, wherever not explicitly stated otherwise. OWA represents a fundamental requirement for Semantic Web (SW) languages [W3C] and, consequently, SW applications may also process data structures expressed by BR that cannot be considered as complete knowledge.

[3] This kind of incomplete descriptions may also express proprietary entities from within the business model. In fact, the complexity of business descriptions that need to be expressed by BR may not make it possible to exhaustively express the business domain.

2.2 Interpreting Entities in the Business Domain

The semantics of BR is typically described by providing a mapping from the rules syntax to some well-known logic formalism. Three (potentially conflicting) basic requirements of such formal models have been identified:

1. *High expressive power.* A basic requirement for the underlying logic is to match the high expressive power of the specific CE without further constraining the sentential forms that can be interpreted. Business analysts are accustomed to using plain English and would not accept any too severe limitation to the expressive power of the modeling language.
2. *Tractability.* In order to automate rule checking and execution, the underlying logics's expressive power has to be carefully balanced against tractability.
3. *Non-falsifiability.* BR semantics should rely on *monotonic reasoning*, i.e. on inference paradigms whose conclusions cannot be contradicted by simply adding new knowledge to the system.

These three requirements have been emphasized by business analysts and researchers as guidelines toward finding the correct logical interpretation of business models, but are NOT satisfied by current BR modeling. Furthermore, as anticipated above, managing this category of descriptions in the EE may pose novel requirements. Specifically, aggregating heterogeneous data sources that are not under full control of each system participating in the EE demands more attention when data is evaluated. The first set of entities that will be described belong to this category of *open* descriptions and we will show that only some of them can lead to automated deductions in such a way that their full semantics is preserved. The business rules that follow are meant to describe a generic *product* that is made available in a market as a consequence of cooperation among *manufacturers*, *distributors*, and *resellers*. These entities are to be considered external to the system that will process the information; you may suppose that a company is doing this in order to monitor markets that are interested by their business. Consequently, we are going to consider each of these entities as open; that is, incomplete with regard to their formal definition and also with regard to existing instance data associated with them. As an example, consider the following business rules:

$$\text{a product is produced by exactly one manufacturer} \tag{3}$$

$$\text{a product is distributed by at least one distributor} \tag{4}$$

$$\text{a product is reselled by at least one reseller} \tag{5}$$

Clearly, the rules above define constraints that instances of concept *product* must satisfy. Furthermore, they refer to attributes of a product instance (*produced by*, *distributed by*, and *reselled by*) that relate product instances with (possibly complex) data structures expressing *manufacturers*, *distributors*, and *resellers*. They may also indicate datatype attributes, such as *price*, *weight*, etc., that are required by metrics based on numeric calculations. Considering state of the art

paradigms for data storage, constraints (3)-(5) cannot take the form of mandatory attributes in a relational schema (or more expressive trigger-based constraints) because we assume that instance data may be incomplete, e.g. both the following tuples do indicate, in the knowledge base, valid product instances:

product	**manufacturer**	**distributor**	**reseller**
ITEM-01	COMP-01	-	-
ITEM-02	-	-	SHOP-01

On the contrary, rules (3)-(5) can be easily translated into the following FOL statements[4]:

$$\mathsf{Product}(x) \rightarrow \exists!y.\mathsf{Manufacturer}(y) \wedge \mathsf{producedBy}(x,y) \tag{6}$$

$$\mathsf{Product}(x) \rightarrow \exists y.\mathsf{Distributor}(y) \wedge \mathsf{distributedBy}(x,y) \tag{7}$$

$$\mathsf{Product}(x) \rightarrow \exists y.\mathsf{Reseller}(y) \wedge \mathsf{reselledBy}(x,y) \tag{8}$$

Unfortunately, existing FOL reasoners cannot process statements (6)-(8) because they do not comply with the Horn fragment: Specifically, all variables in a Horn rule consequent (*head*) must match variables in the rule antecedent (*body*), while variable y in statements (6)-(8) is not bound to any variable in the rule body. When applying reasoning, enforcing these rules clearly amount to asserting the *existence* of hypothetical class instances related with a product instance by the three properties. Because of the CWA approach of rule reasoners, the semantics of (6)-(8) cannot be expressed.

Instead, languages that are specifically designed for modeling incomplete data sources (like the ones used in the SW) can express these constraints without requiring instances of *concept* Product to actually refer to instances of concepts Manufacturer, Distributor, and Reseller:

$$\mathsf{Product} \sqsubseteq = 1\ \mathsf{producedBy.Manufacturer}$$

$$\mathsf{Product} \sqsubseteq \geq 1\ \mathsf{distributedBy.Distributor}$$

$$\mathsf{Product} \sqsubseteq \geq 1\ \mathsf{reselledBy.Reseller}$$

For the sake of clarity, here we express OWL structures by means of the corresponding Description Logics (DL) syntax. However, there is a one-to-one correspondence between OWL constructs and DL assertions[5]. Inference procedures that are associated with SW languages allow to evaluate data structures according to the OWA, while querying a (structurally equivalent) relational data model may not derive all possible conclusions. In fact, as for the Horn fragment of FOL, databases are bound to consider only existing data instances when executing queries. Consider for example the following query:

retrieve all instances that have a reseller associated with them

[4] In *knowledge representation* (KR), concept definitions are typically indicated by a leading uppercase letter; instead, property definitions start with a lowercase letter.

[5] The OWL Lite and OWL DL sub-languages are isomorphic to the $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ DLs [Baader et al., 2003], respectively, where $(\mathbf{D})$ indicates support for XML Schema datatypes.

Clearly, a database query would return ITEM-02 as the only individual satisfying the query because no reseller is associated with the other tuple. Instead, DL reasoning paradigms may derive that, because of rule (5), ITEM-01 must have a reseller, even if its identity is not known to the system at the moment. Consequently, the relational data model grounding mainstream dataware housing cannot be as expressive as SW formalisms when modeling data structures that are, by definition, incomplete. This not a minor difference, as incompleteness is the most common feature of information exchanged in an inter-organizational business process; it may also be an explicit decision that is taken to avoid defining aspects that are not relevant to the model and, nevertheless, cannot be considered as false knowledge. Unfortunately, the restricted set of constructs that are provided by SW languages, such as OWL DL, can express only a limited subset of the FOL structures that may stem from BR formalization. Firstly, although OWL is very good at expressing constraints on concept and property definitions, business rules often need to take into consideration data instances for their enforcement. Secondly, the model-theoretic approach to OWL reasoning services has dramatic consequences on computational complexity and this inevitably narrows the set of logic structures that can be expressed. As an example, the following definition cannot be modeled with OWL DL:

$$\begin{array}{c}\text{a direct distributor is a manufacturer of a product}\\ \text{that is also a distributor of the product}\end{array} \tag{9}$$

In fact, translating this rule amounts to comparing the fillers of properties producedBy and distributedBy (i.e., the instances at the other end of these relations) for any given product, in order to determine if the product's manufacturer is also a distributor for the same product. Instead, (9) can be easily translated into a Horn rule of the following form:

$$\mathsf{Product}(x) \wedge \mathsf{producedBy}(x, y) \wedge \mathsf{distributedBy}(x, y) \rightarrow \mathsf{DirectDistributor}(y) \tag{10}$$

In order to straightforwardly integrate Horn rules with the structural component of the knowledge base, rules are expressed in the SWRL formalism [Boley et al., 2004]. Since the entities on the left-hand side of the formula are to be considered open with regard to inference, it is possible that evaluating them under the CWA (the only possible interpretation for Horn rules) may not reflect the actual semantics of (9). Moreover, constraints expressed by BR on n-ary relations may not be expressed with OWL by objectifying the relation; consider the following rules introducing the notion of *market*.

$$\text{a product is distributed in at least one market} \tag{11}$$

$$\begin{array}{c}\text{a product that is distributed in a market is distributed}\\ \text{by exactly one distributor in the market}\end{array} \tag{12}$$

While it is possible to express (11) as a DL concept definition, the constraint expressed by (12) cannot. In fact, the triples market-product-distributor should

first be grouped according to the market, then according to the product, and only then it may be checked whether the constraint holds. This degree of complexity cannot be expressed as DL concept and property definitions. The second category of entities that can populate the business domain is constituted by *closed* entities, i.e. data structures that are under full control of the system. These data can be expressed with the wide range of SWRL constructs and can be evaluated according to CWA, with no loss in the semantics being expressed. Let us introduce in the business vocabulary the notion of *article* to indicate, among products in a market, those that are produced by the company under consideration:

$$\text{an article is a product that is produced by the company} \tag{13}$$

Whereas *article* represents a closed entity (i.e., the company is supposed to exhaustively enumerate its products in the knowledge base), rule (13) defines it as a specialization of *product*, which is an open entity. This is a major motivation for the conjunct evaluation of both categories of constructs. Moreover, the open/closed status of an entity may also be implicitly derived by those of the entities defining the former. Consider the following definition of *target market*:

$$\text{a target market is a market and an article is distributed in the market}$$

Since knowledge on articles is, by definition, complete, the rule identifies a closed specialization of the open concept *market* whose instances are of direct interest to the company because some of its products are distributed in that market. Even if, singularly taken, articles and products can be expressed in their full semantics by SWRL rules and OWL constructs, mixing them up may not preserve this property. This is due to information interchange between the distinct reasoning engines that are processing, respectively, closed and open constructs.

2.3 Interpretation Issues

So far, we have been using the term 'interpretation' in the broader sense of 'the act of interpreting'. Now, in order to explain the differences between CWA and OWA reasoning, we must shift to the precise notion of 'interpretation' used in *model theory*, that is a 'mapping from the language to a world'. In the bare-bones knowledge base introduced in this Section, interpretations can be roughly assimilated to assignments of individuals to variables in the logic structures derived from BR. In order to exemplify this, we further specialize concept *reseller* with concepts *shop retailer* and *web-enabled retailer*. These concepts distinguish resellers that are capable of selling goods online from those that don't.

$$\text{a reseller is a shop retailer or a web-enabled retailer} \tag{14}$$

Note that rule (14) also implies that, in our simple example, a reseller has to be *either* a shop retailer *or* a web-enabled retailer. These new open entities may be straightforwardly expressed with OWL DL through the union operator:

$$\mathsf{Retailer} \equiv \mathsf{ShopRetailer} \sqcup \mathsf{WebEnabledRetailer}$$

Now suppose that two distinct rules are created (it may be by different people and for different purposes) associating the newly introduced entities with discount rates that can be applied to them.

$$\text{a shop retailer has a discount of 10\%} \tag{15}$$

$$\text{a web-enabled retailer has a discount of 15\%} \tag{16}$$

These kind of associations generally require SWRL definitions because rules (15) and (16) amount to declaring new property instances relating individuals to literals '10%' or '15%'.

$$\mathsf{ShopRetailer}(x) \rightarrow \mathsf{hasDiscount}(x, \text{'10\%'})$$
$$\mathsf{WebEnabledRetailer}(x) \rightarrow \mathsf{hasDiscount}(x, \text{'15\%'})$$

Finally, suppose that individual SHOP-01 in the knowledge base is known to be a reseller, but it is not known whether it is a shop or a web-enabled retailer (this can be formalized with the assertion Reseller(SHOP-01)). Now, in order to show that it may not be straightforward to derive all possible answers to a query, it is sufficient to query the knowledge base for individuals that have a discount rate associated with them. Intuitively SHOP-01 should be returned because, by rule (14), either of the rules should be applicable to the individual (albeit it is not known, at the moment, which one). On the contrary, this is the typical situation where the model-theoretic approach of OWL reasoning cannot be integrated with the single-model approach of SWRL reasoning without losing information. Specifically, the former will consider SHOP-01 as either instance of ShopRetailer and WebEnabledRetailer in each of the interpretations that are computed[6]; instead, SWRL reasoning would not consider either assignment to hold in the interpretation computed on the basis of facts that are explicitly known to the system.

2.4 Modal Evaluation of Business Rules

Business rules determine which states and state transitions are possible or permitted for a given business domain. Modal BR can be of *alethic* or *deontic* modality. Alethic rules are used to model necessities (e.g., implied by physical laws) which cannot be violated, even in principle. For example, an alethic rule may state that an employee must be born on at most one date. Deontic rules are used to model obligations (e.g., resulting from company policy) which ought to be obeyed, but may be violated in real world scenarios. For example, a deontic rule may state that it is forbidden that any person smokes inside any company building. It is important to remark that widespread domain modeling languages such as the Unified Modeling Language (UML) typically express alethic statements only. When drawing a UML class diagram, for instance, the modeler is

[6] Actually, also as instances of both concepts at the same time, because this is not explicitly prohibited by definition (14).

stating that domain objects belonging to each UML class MUST have the attribute list reported in the class definition, implicitly taking an alethic approach to domain modeling. In business practice, however, many statements are deontic, and it is often important (e.g., for computing metrics) to know if and how often they are violated. Much research work has been done to provide a logics-based model for BR including modalities. Indeed, supporting modalities does not mean that it is mandatory to map the BR to a modal logic. For instance, work by the BR OMG team and specifically by Terry Halpin (including his package NORMA [Curland and Halpin, 2007], an open-source tool which supports deontic and alethic rules) addresses logical formalization for SBVR by mapping BR's deontic modalities into modal operators *obligatory (O)*, *permitted (P)* (used when no modality is specified in the rule), and *forbidden* (F). Deontic modal operators have the following rules w.r.t. negation:

$$\sim Op \equiv P \sim p$$
$$\sim Pp \equiv O \sim p \equiv F\ p$$

Other modal operators used for mapping BR alethic rules are *necessary* ($\Box$), i.e. true in all possible states of the business domain, *possible* ($\Diamond$), i.e. true in some state of the business domain, and *impossible* ($\sim \Diamond$). Alethic operators' negation rules are as follows:

$$\sim \Diamond p \equiv \Box \sim p$$
$$\sim \Box p \equiv \Diamond \sim p$$

Terry Halpin's NORMA approach represents BR as rules where the only modal operator is the main rule operator, thus avoiding the need for a modal logics model. Some allowed BR formulations that violate this restriction may be transformed into an equivalent NORMA expression by applying modal negation rules, the *Barcan formulae*, and their converses:

$$\forall p \Box Fp \equiv \Box \forall p Fp$$
$$\exists p \Diamond Fp \equiv \Diamond \exists p Fp$$

For instance, the BR *For each customer, it is necessary that he provides a credit card* is transformed into *It is necessary that each customer provides a credit card*[7]. However, BR rules emerging from business modeling cannot always be transformed into rules where the only modal operator is the main operator. To support such cases, in principle there is no alternative but to adopt a semantics based on a modal logic; but the choice of the "right" modal logic is by no means a trivial exercise, due to tractability and expressive power problems [Linehan, 2006]. Modal logics engines do exist; for instance, MOLOG [Fariñas del Cerro, 1986]

[7] Another transformation that could be used in this context is the one based on Barcan formulae's *deontic variations*, i.e. $\forall pOFp \equiv O \forall pFp$. We shall not discuss here the application of these transformations to normalizing modal logic formulas; the interested reader can refer to [Linehan, 2006].

has been developed by the Applied Logic Group at IRIT from 1985 on, initially supported by the ESPRIT project ALPES. MOLOG is a general inference machine for building a large class of meta-interpreters. It handles Prolog clauses (with conjunctions and implications) qualified by modal operators. The language used by MOLOG can be multi-modal (i.e., contain several modal operators at the same time). Classical resolution is extended with modal resolution rules defining the operations that can be performed on the modal operators, according to the modal logics that are chosen. However, instead of choosing a modal logics and applying MOLOG-style modal reasoning, most current approaches to BR semantics are based on Horn rules, which is the basis of Logic Programming and counts on many robust implementations. Actually, DLs also provide all the necessary constructs to evaluate multi-modal formulas (by mapping operators to universal and existential quantifiers) but, for DL as for Horn FOL, processing modal formulations in conjunction with the OWL and SWRL constructs introduced so far can easily lead to undecidability. Consequently, much work has to be done to translate BR into models that can be safely executed by reasoners preserving most of the semantics of the original definitions.

3 Declarative Process Flow Description

An important component of a successful business strategy is related with the organization of process work flows. To this purpose, a business process is viewed as the sequence of activities and decisions arranged with the purpose of delivering a service, assuring security and effectiveness, in accordance to the service life cycle. Due the the procedural nature of notations typically used for these descriptions, process flows are usually validated against process termination, verifying the absence of interferences and procedural inconsistencies. Violation metrics based on this validation can be easily devised; still, they do not exhaust the possible metrics that can be calculated on a flow. This Section discusses process flow description, underlining the role played by declarative representations in supporting model cross-checking or sharing a process description among a community of users.

3.1 Workflow Languages

Graphical notations aimed at describing process flows are one of the most widespread tools for supporting business process modeling. Their popularity is due to the capability of supporting both immediate reading and rigorous formalization. Another important advantage is that graphical notations are understandable by all business users (e.g., business analysts designing a process, technical developers implementing it, business people monitoring the process, etc.). A broad range of standards allowing formalization of process flows exist. A partial list of more relevant standards includes: UML Activity Diagram, UML EDOC Business Processes, IDEF, ebXML BPSS, Activity-Decision Flow (ADF) Diagram, RosettaNet, LOVeM, and Event-Process Chains

(EPCs)[van der Aalst et al., 2003] In May 2004, OMG proposed a standard aimed at reducing the fragmentation of notations and methodologies. This standard was named Business Process Modeling Notation (BPMN) [Bauer et al.] and was designed as a tradeoff between simplicity of notation and expressivity. Another very diffused standard is constituted by UML Activity Diagrams. Currently, these two standards are gaining a large diffusion: on the one hand, the first is more popular in the business analysts community; on the other hand, the latter is more popular in the software analysts community. A recent OMG initiative [BMI] is aimed at reconciling UML AD with BPMN by means of an integrated metamodel.

In general, the properties of a flow of transaction cannot be captured by a declarative formalization. This is primarily due to the dynamic nature of transactions that describe dependencies among events and may require the specification of complex processes with the presence of event-driven behaviors, loops, real-time evaluation of actions, and parallelism. Model checkers for declarative theories require a finite state space whereas dynamic process, in general, have an infinite state space. Nowadays, Petri nets are widely adopted for workflow modelling and they have a formal semantics by means of which model checkers can be implemented [Grahlmann, 1997]. Another widely adopted formalization is π-calculus [H Smith, 2003] and it is a process algebra describing mobile systems. Key notions of this formalization are *communication* and *change*. Distinct π-calculus processes may communicate by referring to other processes trough links and pointers. By doing this, the development of a process can be inserted into another and generate a new development cycle. Activities in a workflow are conceptually mapped to independent π-calculus processes. This way, processes use events as the form of communication to determine the behavior of the workflow. Another option is to use formalizations based on higher-order logic, such as situation calculus or temporal logic. Situation calculus was introduced by John McCarthy in 1963, it is a logic formalism designed for reasoning about dynamic domains. Recently, it was used as a base for designing a programming language named ConGolog [De Giacomo et al., 2000]. Temporal logic is a logic aimed at reasoning about propositions qualified in terms of time. Traditionally, temporal Logic formalizes only one of the two paradigms that are required in order to deal with dynamic and concurrent systems. In fact, the information to be derived from the formalization of a dynamic system can involve either the properties a state must satisfy and also the temporal dependencies between events. Some early works, such as [Nicola, 1995], proposed a formalization including both states and events. In [Gnesi and Mazzanti, 2003], such a formalization is applied to system modeling, i.e. UML diagrams.

3.2 A Temporal Logic for UML Statecharts

The main problem in applying model checking to business processes is the state space explosion: for real-life case studies, the state space is usually too large to be efficiently mapped. One solution is to encode the state space symbolically, using predicates, rather than enumerating it. This way, we may work on a more

abstract representation while preserving the structure of the dynamic model. The most common way to adopt a predicate-based model is the adoption of temporal logics. In particular we can mention Linear Temporal Logic (LTL) and Computation Tree Logic (CTL)[Jain et al., 2005]. In order to deal with the infinite states generated by loops, special model checkers have been implemented. These model checkers develop algorithms for strong fairness. A strong fairness constraint is aimed at excluding loops. If p and q are properties, we state that if p is true infinitely often, then q must be true infinitely often as well. Intuitively, a property p can only be true infinitely often if there is some kind of loop in the model in which p is made true. Consequently, the strong fairness constraint on (p, q) says that if there is some loop which makes p true infinitely often, then q must be made true infinitely often by the loop as well. If this is not the case, the loop is not strongly fair and the loop must be exited after a finite number of iterations.

3.3 Declarative Representation

Despite the limitations in describing dynamic and concurrent systems, declarative formalizations are not irrelevant to dynamic and concurrent systems, and can be exploited for some important tasks related to validation such as:

- consistency checking;
- data exportation;
- annotation.

Consistency Checking. This is a task where declarative formalizations play the main role. Traditional formalisms are aimed at verifying performance properties of workflow models. For instance, a typical problem is to identify if a path is terminating or which tasks are in dependency with others. Declarative formalizations cannot support this kind of controls but act very well for evaluating the consistency of the objects acting in the transaction or the data objects exchanged in the transaction, as discussed in [Haarslev and Moller, 2001].

Data Exportation. Basically any notation used for representing process flow rely on an XML format used for exporting data. This is a declarative description of the flow, usually limited to a syntactic description of the elements in the notation, that in principle could be queried for consistency checking purposes. This approach is not straightforward, because it requires reconstructing the semantics of the notation directly in the query step. Since the usual approach is to provide a semantic mapping between XMI and the individual format to be queried [Fox and Borenstein, 2005].

Annotation. In [Melnik and Decker, 2000], a RDF format has been provided for representing UML diagrams. RDF is a language for data annotation that allows to attach complex assertions (in the form of triples subject-predicate-object) to URIs, i.e. any type of resource. Typically this language is used in systems for cooperative design, such as described in [Ceravolo et al., 2007]. The final output of this approach allows to share process flow annotations and cooperatively update the description of a process.

3.4 Consistency Checking

Here we propose an example of consistency check between the structural part of the model and the process flow. Fig. 1 shows a BPMN diagram describing process coordination between distinct actors, represented by different *swim lanes* in the diagram. A declarative description of the flow can describe business transactions in terms of the actors involved in the transaction plus input and output data required for executing the transaction. As an example, the static model may

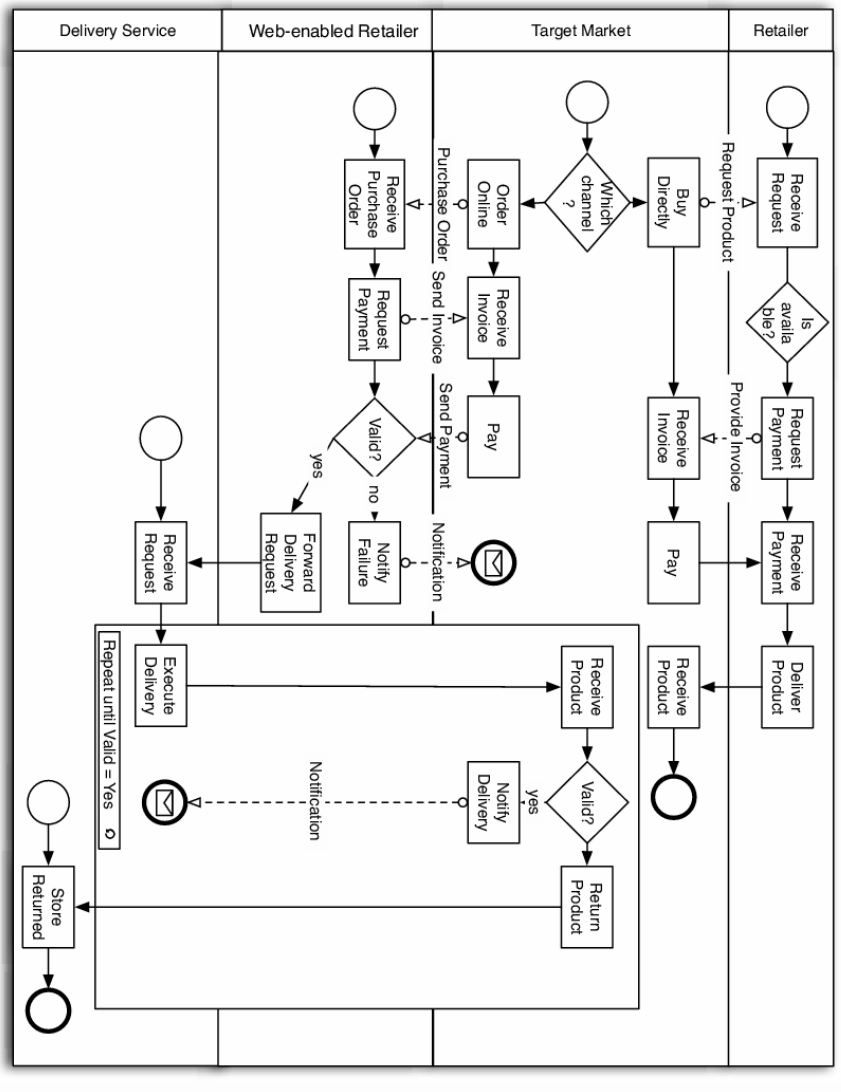

Fig. 1. An example of process flow

feature concept definitions for *Event*, *Activity*, *Gateway*, and all the other BPMN constructs. Entities in Fig. 1, such as the tasks activated by the *Retailer* in the first lane, can then be expressed in terms of assertions (i.e., subject-predicate-object triples), such as the following:

ReceivePayment rdf:type Task

RequestPayment rdf:type Task

Instances of these concepts may be related with each other in order to express general requirements that should be satisfied by processes, such as the following:

ReceivePayment follows RequestPayment

Clearly, a diagram that contradicts the requirement can be spotted at design time. Another possible usage of the static model is to constrain the instances of concept *message* that can be exchanged between *Tasks*. As an example, reference to a specific instance of concept *Message* named *invoice* can be restricted to *Tasks* that are contained in the *Retailer* lane. More interestingly, dynamic requirements that are related with run-time execution of processes can also be expressed. In this case, logfiles produced by the execution of processes are interpreted as concept instances and properties relating them with each other.

4 Declarative Value Model Description

Before implementing and executing a business collaboration, models describing this collaboration can be developed. These models help to analyze a priori the collaboration with different stakeholders. Agreements and clarifications can be made on different levels of the collaboration. A model especially important for describing inter-organizational collaborations is a *value model*, estimating profitability for every actor involved in the collaboration. In a collaboration each stakeholder is *profit and loss responsible*. Analyzing a priori profit opportunities as well as agreeing on the exchanges of value between the different stakeholders is highly important. In this Section, we use a running example for illustration. In our example, a *manufacturer* develops a global value model in order to estimate profitability of his business, before implementing his business, and to *monitor* his business during the life cycle of the collaboration. Several modeling techniques can be used to estimate profitability of a collaboration, e.g. Business Modeling Ontology [Osterwalder and Pigneur, 2002] and REA modeling [McCarthy, 1982]. Although modeling techniques differ, value models depict always which *entities of value* are exchanged between stakeholders. Here, we refer to entities of value as *value transfers*. We discuss two different modeling techniques by means of our running example.

4.1 Graphical Based Value Modelling

REA modelling [McCarthy, 1982] is a widely acknowledged business modeling technique using a graphical representation of the actors and their relations. Here,

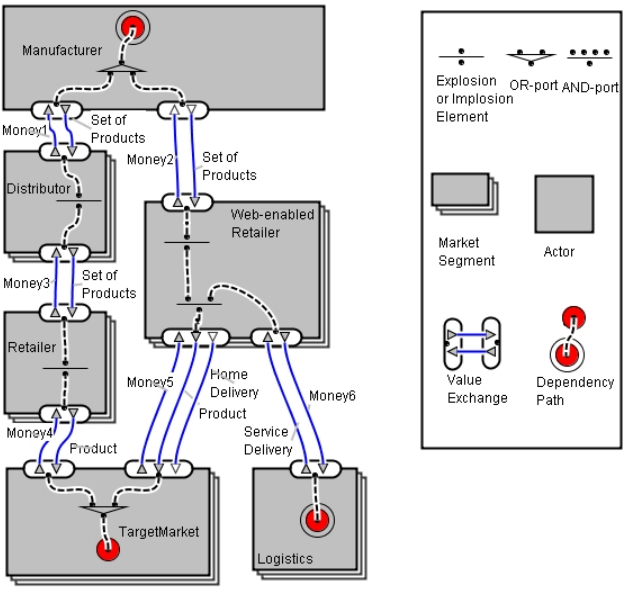

Fig. 2. Business Case as Value Model

Table 1. Estimations

	Money1	**Money2**	**Money3**	**Money4**	**Money5**	**Money6**
Total Value	2000	850	450	50	48	3
# of Products	50	20	10	1	1	1
Euros per Product	40	42.50	45	50	48	3

we use another modeling technique which also has a graphical representation, e^3-value modeling [Gordijn and Akkermans, 2003]. Figure 2 depicts our example as an e^3-value model. The manufacturer sells products to *distributors* and to *web-enabled retailers*. Distributors sell products to *retailers* who, in turn sell products to clients in the *target market*. Web-enabled retailers sell products directly to clients in the target market. Furthermore, they need *logistics* to deliver products to clients. A *consumer need* is fulfilled by one or more value transfers. Value transfers dependent on each other are connected through a *dependency path*. Quantifications of constructs in a dependency path influence the quantification of other constructs in that path. For example, the manufacturer gets higher profits from selling to a target market constituted by two hundred clients than to a market of one hundred only. This model also supports quantification of relations (not shown in the picture). Furthermore, the manufacturer gets higher profits from selling to web-enabled retailers than from selling via distributors. Table 1 depicts estimations the manufacturer made on the number of sold products and the value of each product. These estimations, together with the graphical model, provide an estimation of profitability.

4.2 Logic Based Value Modelling

Another modeling technique for value based modeling, closer to the other ones used in this chapter, is using a *logical formalism*. Here, we depict the running example as a value model in Prolog style in FOL. Again, *value transfers* as well as *dependencies* between these value transfers are modeled. Each construct in e^3-value can be mapped to a *predicate* in this model. Each construct has several arguments, at least denoting the *actor* to which they belong, their *name* and estimated *quantification*, e.g. *number of occurrences*. A connection element has a unique *id* and connects two constructs. In e^3-value this is the part of a dependency path connecting two constructs. Furthermore, each construct in the model captures an *equation*. We demonstrate the general approach by formalizing two predicates. The first is modeling a consumer need and its dependencies on other constructs. In e^3-value this is modeled as a start stimulus and corresponds to predicate *Start*. The second is modeling a *choice* in fulfilling a consumer need and dependencies related to this choice. For example, the consumer need is to buy a product which might be fulfilled by either buying it online or by purchasing it in a store. In e^3-value this is modeled with an *OR-port* and corresponds to predicate *Or_Split*. To create the equation denoting dependencies of a consumer need in a value model, a distinction between equations resulting from market segments (*Equation_StartMS*) and actors (*Equation_StartActor*) is made. To derive the equation, first the estimated number of *occurrences* of the start stimulus (consumer need) is required. When the equation depicts a market segment, also the number of customers in the market segment (*count*) is necessary. Furthermore, the *id* of the connection element, indicating dependency on other constructs, is necessary. For the equation of a choice, again the *id* of the connection element is needed. Furthermore, the estimated number of times a choice is made (*fraction*) is needed.

$$\begin{aligned}
&\text{Equation_StartMS}(id, occurrences, count) \quad \leftarrow \\
&\quad \text{Start}(actor, name, occurrences) \ \wedge \\
&\quad \text{Connection}(id, name, name_2) \ \wedge \\
&\quad \text{Market_Segment}(actor, count)
\end{aligned}$$

$$\begin{aligned}
&\text{Equation_StartActor}(id, occurrences) \quad \leftarrow \\
&\quad \text{Start}(actor, name, occurrences) \ \wedge \\
&\quad \text{Connection}(id, name, name_2) \ \wedge \\
&\quad \text{Actor}(actor)
\end{aligned}$$

$$\begin{aligned}
&\text{Equation_Split}(id_1, fraction_1, fraction_2, id_2) \quad \leftarrow \\
&\quad \text{Or_Split}(actor, name_1, fraction_up, fraction_down) \ \wedge \\
&\quad \text{Connection}(id_1, name_1, name_2) \ \wedge \\
&\quad \text{Connection}(id_2, name_3, name_1) \ \wedge \\
&\quad ((fraction_1 = fraction_up \ \wedge \ fraction_2 = fraction_down) \\
&\quad \vee \\
&\quad (fraction_2 = fraction_up \ \wedge \ fraction_1 = fraction_down))
\end{aligned}$$

Now, we illustrate the creation of the equations based on the gathered data. There are two sets of equations calculated. The first set is the set of equations

without instantiations. The second set of equations consists of equations which are instantiated with the values as estimated. In our example these are the estimations made by the manufacturer in Table 1. Predicate *Value* contains two arguments representing these estimations. The first argument is the corresponding name in the equation and the second argument is the estimated value. Next, the derivations of both sets of equations are depicted.

$x = y \quad \leftarrow$
Equation_StartActor(x, y)
$\vee$
(Equation_StartActor$(x, occurrences)$ $\wedge$ Value$(occurrences, y)$)

$x = y \cdot z \quad \leftarrow$
Equation_StartMS(x, y, z)
$\vee$
(Equation_StartMS$(x, occurrences, count)$ $\wedge$ Value$(name, y)$ $\wedge$
Value$(count, z)$)

$y = \frac{r}{r+s} \cdot x \quad \leftarrow$
Equation_Split(y, r, s, x)
$\vee$
(Equation_Split$(id_1, fraction_1, fraction_2, id_2)$ $\wedge$ Value$(fraction_1, r)$ $\wedge$
Value$(fraction_2, s)$)

Next, formalization for the market segment *TargetMarket* is depicted, illustrating formalizing our running example. When formalizing a value model, all market segments and actors, as well as each construct and connection element, are described. Furthermore, the estimations made by the manufacturer are added with the *Value* predicate. Predicate *Value_Transfer* represents the actual value transfer where the last argument is the estimated average value which are also represented in Table 1.

Market_Segment$(TargetMarket, Count_1)$
Start$(TargetMarket, Start_1, Occurrences_1)$
Or_Split$(TargetMarket, Or_1, S, R)$
Value_Transfer$(Targetmarket, Transfer_1, Value_1)$
Value_Tranfser$(Targetmarket, Transfer_2, Value_2)$
Connection$(Id_1, Start_1, Or_1)$
Connection$(Id_2, Or_1, Transfer_1)$
Connection$(Id_3, Or_1, Transfer_2)$
Value$(Count_1, 100)$
Value$(Occurrences_1, 5)$
Value$(S, 2)$
Value$(R, 1)$
Value$(Value_1, 50)$
Value$(Value_2, 48)$

Expressing the value model in a logical formalism and quantifying it with estimations enables profitability calculations on the collaboration. These calculations are used for decision making. In this Section we showed different representations of value models. Both representations, graphical as well as logic based, enable reasoning on the profitability of the collaboration. Although these are two distinct representations, both model explicitly *actors* and *value exchanges* between those actors. Essentially, this is the important part for a priori evaluating profitability of a collaboration.

5 Relations between Models

Models are per se an abstraction of the real world focusing on a particular aspect like, for example, the flow or the value aspect. Dependent on the model, the analyses that can be performed differ: in case of the process flow model it can be checked whether there are dead ends in the execution or whether deadlocks can occur; on the other hand, in the value model, profitability can be investigated. However, all these models describe the same system and therefore the different models can be related with each other. The aim would be to have a description of the system according to the different aspects. To check whether different models fulfill this property, relations between the different models have to be investigated. Well known relations are equivalence checking, whether two models describe exactly the same, and consistency checking, whether two systems are contradicting each other. Equivalence of two different model types (like for example process flow and value model) will never be given since the different models focus on different aspects while neglecting other aspects. With regard to process flows and value models, the former focus on the order of message exchanges, while the latter disregards these aspects and focuses on the occurrences and values of exchanged value objects. As a consequence, consistency[8] seems more appropriate, since it focuses on contradiction free models which can be checked on the communalities of the involved models.

There are plenty of potential pitfalls that could make models inconsistent, like for example different understandings of the architects of the different models on how the actual system really works (in case it has been already implemented), modifications of a single model without maintaining the remaining models, or the discrepancy between the modeled behavior and the behavior of customers in concrete business situations. Furthermore, there are plenty of cases where consistency between two models cannot be checked at all like, for example, in case models are representing the system on a different level of granularity. In case of the process flow and the value model, this means that the value model considers the modeling at the granularity of companies, while the process flow model is based on the notion of business units within companies. However, the investigation of relations between the models, as well as the actual behavior of

[8] In some literature the term compatibility or soundness is used instead of consistency but addressing the same problem.

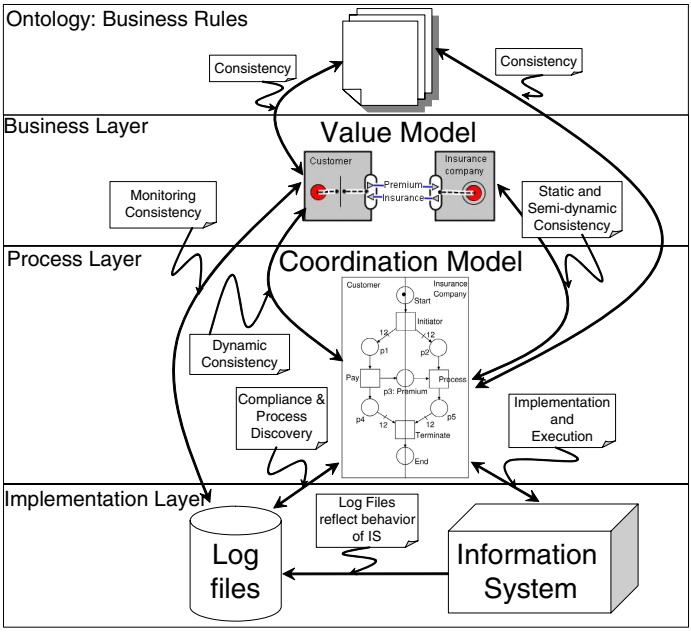

Fig. 3. Consistency Relations

the system, are valuable to improve, correct, or adapt the models according to a continuously changing environment[9].

In the following of this Section, different relations between the models introduced in Sections 2-4 (see Figure 3) are mentioned and some references are provided. Afterwards, an overview of two sample consistency relations are presented.

5.1 Overview

Figure 3 is structured in four layers according to the usual information system life cycle: specifically, the business rules, business, process, and implementation layers. First, understanding how a particular business works and what the dependencies between the different parties and business objects are is important. Second, a business idea containing a business model showing the profitability of the business is constructed. Third, it is specified how you want to do business and how to coordinate the different parties involved. Finally, the information system is implemented deployed and and becomes operational, resulting in log files representing the execution of the information system to a specific level of detail.

[9] We want to point out that, in the following, the focus is on relations of different model types. There exist quite some work on relations between models of the same type.

The relations depicted in Figure 3 are defined between the different layers. One set of relations is between the business rules and the remaining layers. In particular, the business rules terminology is used in the value model to describe the different actors or value objects. In our examples we used terms like "target market" or "product" in business rules and value model. Also, the process flow model relies on the terminology provided in the business rules, for example, to name swim lanes or to express what business objects are related to which messages exchanged between the swim lanes. As an example, "target market" and "product" represent, respectively, a swim lane and a business object. This common terminology is essential for direct comparison between value and process flow models. Since the process flow describes the exchange of messages sent and received by an information system, the log file generated by an information system uses the same messages. This relation indirectly allows to relate some messages to business objects in the value model, which in there turn can be related to terminology in the business rules. As a consequence, value model, process flow model, as well as the log files can be related to the terminology in the business rules. This set of relations allows a first checking whether the dependencies in each underlying layer is consistent with the rules specified in the business rules declaration. Further, the terminology provided by the business rules and the relation to a value exchange, a message exchange, and a log file entry as well as to an actor, a swim lane, and a communication partner enables checking of relations between the different models.

Figure 3 shows three consistency relations between value and process flow model: static, semi-static and dynamic consistency.

- **Static consistency.** Checks whether the set of exchanged values in the value model corresponds to the business objects exchanged in the process flow model represented by message exchanges. Here the aim is to relate the value objects and business objects exchanged, as well as the dependencies between value exchanges and business object exchanges. A more detailed description can be found in Section 5.2.
- **Semi-dynamic consistency.** Checks whether the forecast of a value model can be accomplished by the process flow model. This is mainly focusing on investigating expected changes in a business scenario, like for example the increase of the amount of expected customers using the system or an expected change of user behavior for example because of price changes for a particular modeled product group compared to other modeled product groups.
- **Dynamic consistency.** Checks whether the execution of the process flow model is consistent with the expected behavior represented in the value model. This relation cannot be evaluated directly since the process flow is not executed directly. However, the information system implementing the process flow model is executed directly. Therefore, dynamic consistency can be derived from investigating the three relations between value model, log files (hence implicitly the information system) and the process flow model depicted in Figure 3: first this is *monitoring consistency*, second the relation

between log file and information system representing the *behavior of the information system* in Figure 3, and third the relation between process flow model and information system describing the *implementation and execution* of the process flow model by the information system. The *dynamic consistency* relation is essential since it gives an a-posteriori evaluation of whether the process flow and the value model are indeed describing the implemented information system. In case there are inconsistencies, the models have to be adapted and re-evaluated with regard to the expected profitability and behavior respectively. A more detailed description of the dynamic monitoring relation can be found in Section 5.3.

The **compliance and process discovery** relation has not been mentioned so far. One aspect of this relation is also known as process mining [van der Aalst and Weijters, 2005], where the aim is to derive a process model from log file data. This relation can be applied for example in case of exceptional behavior in information systems, which is not reflected in the process flow model. It is closely related to the dynamic monitoring relation between value model and log files.

5.2 A Priori Model Evaluation: Static Consistency

During the modeling phase it can be checked whether the value model and the process flow model are still consistent, that is, are not contradicting each other. This check is performed on the commonalities between the two models, which are the actors and swim lanes, the value exchanges and message exchanges, and the dependencies between exchanges in the corresponding model. Relating swim lanes and actors to each other is done using business rules. Considering the *Delivery Service* swim lane in the process flow model (see Figure 1) and the *Logistics* actor in the value model (see Figure 2) different terms are used describing the same concept. A business rule describing that a logistics company provides a delivery service is necessary to indicate that the *Delivery Service* swim lane and the *Logistics* actor are not contradicting. The same requirement of a business rule applies to the message exchange *payment* and the value exchange *money*. Obviously money is used to perform a payment.

Based on this concept relations of value models and process flow models, a consistency checking can be performed. A value model and a process flow model are consistent if for every set of message exchanges representing an execution sequence in the process flow model there exist a set of value exchanges representing a single dependency path in the value model and vice versa. However, there exist value exchanges which do not result in a message exchange, like an experience or a knowledge gain. Further, there exist message exchanges in the process flow which do not have a corresponding value exchange. In the example depicted in Figure 1, the message exchanges *request product* and *provide invoice* do not have a corresponding value exchange since these message exchanges are for coordination purposes only. A more detailed description of the approach has been discussed in [Zlatev and Wombacher, 2005].

5.3 A Posteriori Model Evaluation: Monitoring Consistency

After the modeling phase, the system is implemented and executed. It is important to check during the life-cycle of the collaboration whether estimations in the a priori value model are met by the running system. One approach for checking this dynamic consistency is to *monitor* the running system. This is the *monitoring consistency* relation in Figure 3 between log file and value model. Here, we show monitoring of the running system based on log files produced by the information systems. This approach has been formally introduced in previous work by one of the authors of this chapter [Bodenstaff et al., 2007].

In our running example, the manufacturer monitors the collaboration. His main interest is monitoring the *ratio* between products sold to distributors and web-enabled retailers, since he gets higher profits from selling to web-enabled retailers than from selling to distributors. The tool used for the monitoring shows the realized number of value exchanges with distributors as well as web-enabled retailers. The manufacturer can now compare these realized values with estimations made in the original value model. Furthermore, the tool enables reasoning over the relation between the realized values and the different constructs in the value model. The equations derived from the value model are added to the monitoring tool, showing all quantified values and their relations. Figure 4 depicts

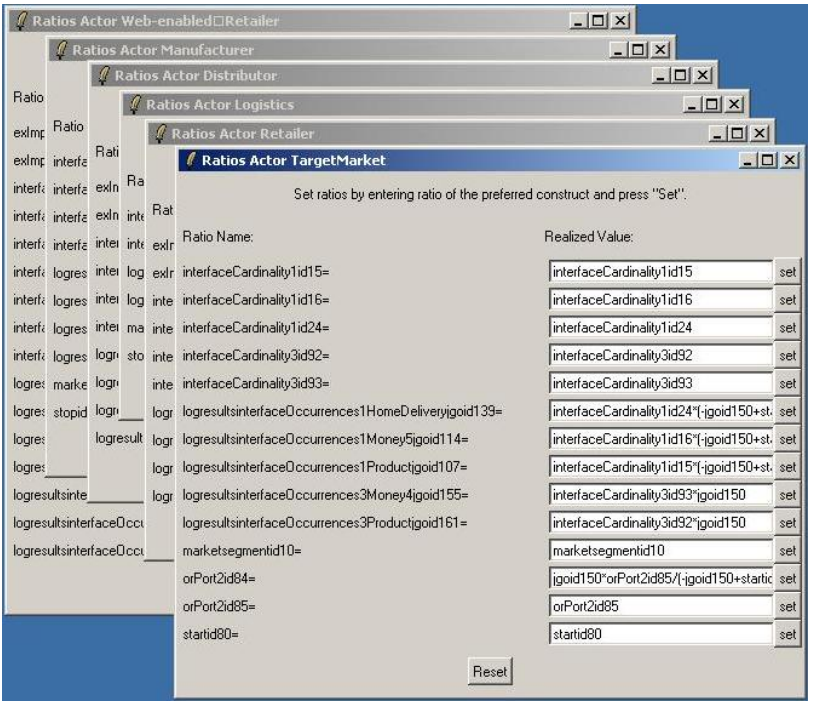

Fig. 4. Monitoring Collaboration - Screen Shot

a screen shot of the tool where the realized values and their relation to the different constructs are shown. Showing these relations enables the manufacturer to reason over effects the constructs have on the value exchanges and on which constructs are influenced by the realized values.

6 Conclusions

In the last few years, the concept of the business process model has become increasingly popular. When designing a new business process, modeling accuracy is likely to be a crucial factor for being able to assess its strength and weaknesses. The Web infrastructure and the availability of Semantic Web-style metadata have enabled new types of inter-organizational business processes, which in turn need new business process models. Despite the surge of interest in representing business process models, reasoning on process descriptions still faces many open issues, mostly due to lack of a shared notion of business process representation. Many approaches focus on producing taxonomies or categorizations, or on stating what aspects of business process models should be included in (or excluded from) modeling. The capability of representing multiple facets is indeed an important requirement for business process models. In this chapter, we discussed the different facets such a representation must possess, starting from standard ones like workflow representation and arriving to the capability of defining the structure of a companys value chain and describing the position of the process actors within the value network. Also, evidence has shown that initial business process models are often unsuccessful and need to keep being modified until a viable model is found; therefore, a multi-faceted representation of the process needs *i)* to support a priori evaluation of feasibility *ii)* to be able to evolve based on run-time evidence. While we believe a visual approach to modeling to be preferable from the modelers point of view, in this chapter we focused on the logics-based models underlying each facet of the process representation. Namely, we developed on the idea that the different facets of a business process model require different formalizations, involving different inference techniques, some of them Semantic Web-style, other closer to classical Prolog reasoning. Using a simple business process model, we discussed how well our multi-faceted representation can be integrated by a hybrid approach to reasoning, transferring knowledge and metrics obtained reasoning on each part of the model to the other parts. While we believe results to be encouraging, more work is needed. Specifically, we wish to highlight the following open issues;

- *Critical areas detection*: A multi-faceted business process model needs to include a way of finding necessary changes, when the original model does not work as envisaged. This might be done by including more options in the original model, plus decision points on when to start using these options based on constraints expressed un the value-model.
- *Representing pragmatics*: Reasons for stakeholders: actions should be part of the model. Our sample model shows what is meant to happen, but not why. Research on agent-based systems has shown that reasons for actions

are difficult to express using logics-based modeling. Also, current business process models focus on the point of view of the business owner. It might be useful to enlarge the model including the point of view of potential customers, e.g. based on market research data.

Acknowledgments

This work was partly funded by the Italian Ministry of Research under FIRB contract n. RBNE05FKZ2_004, TEKNE and by the Netherlands Organisation for Scientific Research (NWO) under contract number 612.063.409. The authors wish to thank Claudio Pizzi for his valuable comments on modal logic reasoning.

References

[Aßmann et al., 2005] Aßmann, U., Aksit, M., Rensink, A. (eds.): MDAFA 2003. LNCS, vol. 3599. Springer, Heidelberg (2005)

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)

[Bauer et al.] Bauer, B., Müller, J., Roser, S.: A model-driven approach to designing cross-enterprise business processes. In: Meersman, R., Tari, Z., Corsaro, A. (eds.) OTM-WS 2004. LNCS, vol. 3292, pp. 544–555. Springer, Heidelberg (2004)

[BMI] BMI. Business modeling & integration domain task force, http://bmi.omg.org/

[Bodenstaff et al., 2007] Bodenstaff, L., Wombacher, A., Reichert, M., Wieringa, R.: Monitoring collaboration from a value perspective. In: Proceedings of 2007 Inaugural IEEE International Conference on Digital Ecosystems and Technologies IEEE-DEST 2007, pp. 134–140 (2007)

[Boley et al., 2004] Boley, H., Dean, M., Grosof, B., Horrocks, I., Patel-Schneider, P.F., Tabet, S.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004)

[Boley et al., 2001] Boley, H., Tabet, S., Wagner, G.: Design rationale of RuleML: A markup language for semantic web rules (2001), http://citeseer.ist.psu.edu/boley01design.html

[Ceravolo et al., 2007] Ceravolo, P., Damiani, E., Viviani, M.: Bottom-up extraction and trust-based refinement of ontology metadata. IEEE Transactions on Knowledge and Data Engineering 19(2), 149–163 (2007)

[CLT] CLT. Controlled Natural Languages, http://www.ics.mq.edu.au/~rolfs/controlled-natural-languages/

[Curland and Halpin, 2007] Curland, M., Halpin, T.A.: Model driven development with norma. In: HICSS, p. 286 (2007)

[De Giacomo et al., 2000] De Giacomo, G., Lesperance, Y., Levesque, H.J.: ConGolog, a concurrent programming language based on the situation calculus. Artificial Intelligence (2000)

[Fariñas del Cerro, 1986] Fariñas del Cerro, L.: Molog: A system that extends prolog with modal logic. New Gen. Comput. 4(1), 35–50 (1986)

[Fox and Borenstein, 2005] Fox, J., Borenstein, J.: XMI and the many metamodels of enterprise metadata. In: XML conference and exhibition (2005)

[Fuchs and Schwertel, 2003] Fuchs, N.E., Schwertel, U.: Reasoning in Attempto Controlled English. In: Bry, F., Henze, N., Małuszyński, J. (eds.) PPSWR 2003. LNCS, vol. 2901, pp. 174–188. Springer, Heidelberg (2003)

[Fuchs et al., 1999] Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto Controlled English (ACE) Language Manual, Version 3.0 (1999), http://attempto.ifi.unizh.ch/site/pubs/papers/ace3manual.pdf

[Gnesi and Mazzanti, 2003] Gnesi, S., Mazzanti, F.: A mu calculus for temporal logic. In: ACM Specifying and Verifying and Reasoning about Programs (2003)

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

[Grahlmann, 1997] Grahlmann, B.: The PEP tool. In: Proceedings of CAV (1997)

[H Smith, 2003] Smith, H., Fingar, P.: Business Process Management The Third Wave. Meghan-Kiffer Press (2003)

[Haarslev and Moller, 2001] Haarslev, V., Moller, R.: Description of the RACER system and its applications. In: International Workshop on Description Logics (2001)

[Jain et al., 2005] Jain, H., Kroening, D., Sharygina, N., Clarke, E.: Word level predicate abstraction and refinement for verifying rtl verilog. In: DAC 2005: Proceedings of the 42nd annual conference on Design automation, pp. 445–450. ACM Press, New York (2005)

[Jena] Jena. Jena, A Semantic Web Framework for Java, http://jena.sourceforge.net/

[Linehan, 2006] Linehan, M.: Semantics in model-driven business design. In: Proc. of 2nd International Semantic Web Policy Workshop, SWPW 2006 (2006)

[McCarthy, 1982] McCarthy, W.E.: The REA Accounting Model: a Generalized Framework for Accounting Systems in a Shared Data Environment. Accounting Review 57, 554–578 (1982)

[Melnik and Decker, 2000] Melnik, S., Decker, S.: A Layered Approach to Information Modeling and Interoperability on the Web. In: Semantic Web Workshop (2000)

[Nicola, 1995] Nicola, R.D.: Three logics for branching bisimulation. Journal of the ACM (1995)

[OMG, 2006] OMG. Semantics of Business Vocabulary and Business Rules Specification (2006), http://www.omg.org/cgi-bin/apps/doc?dtc/06-08-05.pdf

[Osterwalder and Pigneur, 2002] Osterwalder, A., Pigneur, Y.: An e-business model ontology for modeling e-business. In: Proceedings of the 15th Bled E-Commerce Conference - Constructing the eEconomy (2002)

[Parsia et al., 2003] Parsia, B., Sivrin, E., Grove, M., Alford, R.: Pellet OWL Reasoner. Maryland Information and Networks Dynamics Lab (2003), http://www.mindswap.org/2003/pellet/

[Ross, 2003] Ross, R.G.: Principles of the Business Rule Approach. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)

[van der Aalst et al., 2003] van der Aalst, W., Hofstede, A., Weske, M.: Business process management: A survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)

[van der Aalst and Weijters, 2005] van der Aalst, W., Weijters, A.: Process-Aware Information Systems: Bridging People and Software through Process Technology. In: Process Mining, pp. 235–255. Wiley & Sons, Chichester (2005)

[W3C] W3C. Semantic web activity, http://www.w3.org/2001/sw/

[W3C, 2004] W3C. OWL Web Ontology Language Overview (2004), http://www.w3.org/TR/owl-features/

[Zlatev and Wombacher, 2005] Zlatev, Z., Wombacher, A.: Consistency between e^3-value models and activity diagrams in a multi-perspective development method. In: OTM Conferences, vol. (1), pp. 520–538 (2005)