Business Processes in Virtual organizations: an Ontology-Based Conceptual Model

Johny Ghattas

THESIS SUBMITTED AFTER CONFERRAL OF THE MASTER DEGREE

University of Haifa

Faculty of Social Sciences
Department of Statistics

February, 2006

Business Processes in Virtual organizations: an Ontology-Based Conceptual Model

By: Johny Ghattas

Supervised by: Dr. Pnina Soffer, Prof. David Pargi

THESIS SUBMITTED AFTER CONFERRAL OF THE MASTER DEGREE

University of Haifa

Faculty of Social Sciences
Department of Statistics

February, 2006

Approved by: _____ **Date:**_____
(Supervisor)

Approved by: _____ **Date:**_____
(Supervisor)

Approved by: _____ **Date:**_____
(Chairperson of M.A. Committee)

Acknowledgement

I would like to thank my Tutor Dr. Pnina Soffer for her support and advice during the whole research process.
I would like to thank also Prof. Yair Wand, for his valuable advices during the thesis research subject and research question definition
Lat but not the least I would like to thank my Wife, Children and my Parents for their support and patience during the whole process.

TABLE OF CONTENT

Business Processes in Virtual organizations: an Ontology-Based Conceptual Model

By Johny Ghattas

Abstract

Virtual organizations (VO's), as collaboration between organizations that share business processes for a specific purpose, have received a considerable amount of attention in recent years. This Thesis presents a formal ontology-based analysis of business processes in virtual organizations, aimed at achieving an understanding at the conceptual level of the challenges of business process design in such environment. The model is meant to serve as a theoretical basis for the development of support and implementation approaches for this domain, allowing partner organizations the benefits of trust and coordination without harming their autonomy and privacy. It enhances the Generic Process Model (GPM) to include concepts related to business processes in a virtual organization. In particular, it addresses contractual obligations between organizations and their incorporation into business process design. The model is illustrated by an inter-library loan case study. A proposed BP modeling and assessment approach is proposed and evaluated through the past researches in the field.

List of Tables

List of Figures

1. Introduction

1.1. The virtual organization collaboration model

Collaboration between business organizations is often critical for business survival in today's world. The importance of collaboration stems from the tendency towards specialization and focus on core business while outsourcing or buying capabilities, resources and even business processes from business partners.

In this context, the "virtual organization" concept is being frequently used. There is no single, agreed upon definition of a virtual organization. It is sometimes defined as any alliance, temporary or permanent, between two or more legal entities, that exists for the purpose of furthering business or social objectives, without causing the participants to lose their autonomy [62][63][19][24][70][84]. Nevertheless, virtual organizations, as collaboration between organizations that share business processes for a specific purpose, exist as part of today's business world. An example of such collaboration forms is content provisioning to cellular phone customers, which is performed by a variety of content providers. The content providers operate in collaboration with the cellular telecommunication provider, who acts as the sole contact point with the customer.

Various aspects of virtual organizations (VO's) have been addressed in the literature. The advantages of this collaboration form have been discussed in [24][27]. Key implications of business organization virtuality that are mentioned are in reconfigurable business process forms, considerably more blurred organization boundaries, and relationships between organizations, which are likely to be of contractual forms. Frequent switching of tasks, roles, or work assignments is also typical in virtual organizations. In theory, change management may become simpler, but there is not enough evidence yet to assess the extent to which it is true [24]. Implementation issues have been addressed too, and solutions have been proposed for tasks such as contract negotiation and the design of systems for collaborative business process management [4][25][48][50][53].

However, we believe that solution and implementation details should build upon a conceptual understanding of the problem domain and its unique structure and behavior.

This kind of conceptual understanding is crucial for building models and designing processes, for making sure that any research effort scoping is correct, valid, and complete, and that the proposed solutions are generic enough. To the best of our knowledge, such understanding of the virtual organization domain has not been achieved yet.

1.2. Research Motivation

This research objective is to establish a formal ontology-based analysis of business processes in virtual organizations, aimed at achieving an understanding at the conceptual level of the challenges of business process design in such environments.

We consider that conceptual understanding of the domain is the most basic step before we can do any requirement engineering and posterior implementation of any correct and valid process modeling. This is the identification of the entities that compose the problem domain, their properties, attributes, relations with other entities and dynamic evolution. [32][33][44][46][54][55].

1.3. Research methodology

We depart from the generic well known ontological model of Bunge, Wand and Weber (BWW) [12][13] as specialized for business processes in the Generic Process Model (GPM) [81][82], and extend it by concepts necessary for understanding the virtual organization domain. Later we apply it to a real case study illustrating a typical VO structure and business processes – the Interlibrary Loan Business process (ILL).

Our work focuses on providing a model devised to be formal and generic enough to be then used to help defining different implementation forms for specific scenarios. By this we mean, for example, that workflow models can be used to implement business processes, as implementation models that build upon our conceptual model.

The model is devised by the use of a thorough analysis of a proposed study case, and further evaluated through the reviewed literature.

Once evaluated, we analyzed which of the conclusions that were attained through the case study may be generalized to VO in general, newly through the reviewed literature.

This generalization leads us to present a generic Business process modeling approach that should be applicable to most of the VO's that comply with our model assumptions

(such as the no central governance assumption and the conservation of the total business autonomy and independence of the VO partners).

Finally we summarize the model and future lines of work.

1.4. Thesis structure

This thesis report is structured as follows:

- **Chapter 2** presents BWW's ontology and the GPM framework, which are the main building blocks of our model.
- **Chapter 3** presents related literature review.
- **Chapter 4** presents our virtual organization model.
- **Chapter 5** presents the proposed VO Business Process Modeling (BPM) approach
- **Chapter 6** provides an illustrative case study where we demonstrate our concepts and model in a real world scenario, and analyze the generalization of the attained conclusions to Virtual organizations.
- **Chapter 7** presents the model evaluation based upon the reviewed literature.
- **Chapter 8** presents the Summary, conclusions and future lines of research.

Finally, for the Reader convenience the VO Model and the VO PBM are summarized in Annex 1 and Annex 3 respectively.

2. The model building blocks

2.1. General

Ontology based modeling has been acquiring a lot of focus in many research domains: information systems research, information modeling, Requirement engineering and software engineering, Artificial intelligence, Enterprise modeling, are some of them. Today it is largely believed that ontology design leads us to a true and complete model of the real world we are representing through it.

By the early 1980s, researchers in Artificial intelligence (AI) had realized that work in Ontology was relevant to the necessary process of describing the world for intelligent systems to reason about and act in. This awareness and integration grew, and spread into other areas until, in the latter half of the final decade of the 20th century, the term "ontology" actually became a buzzword, as enterprise modeling, e-commerce, and knowledge management, among others, attracted the focus of research communities. Ontologies have been defined for many domains: information modeling, enterprise modeling, biological systems, medical systems, Artificial intelligence, knowledge management, business modeling, business process modeling, etc. Design, specification, evaluation, validation and comparison of ontologies are currently investigated by many researchers in different research domains.

According to Gruber [37], Ontology can be defined as: *ontology is an explicit specification of a conceptualization.* The term *ontology* is borrowed from philosophy, where ontology is a systematic account of existence. In the realm of information systems and AI, ontology has a somewhat different interpretation: ontology is not a theory of what exists, but what a community of practice believes to exist, that is, ontology specifies things that we must assume to exist in order for our theories to be true. What people believe to exist, is named a conceptualization [38] . It represents an abstract, simplified view on the world. In our situation, the simplified world is the world of business processes within VO's. Modern definitions of ontology (see e.g. [10]) emphasize that there must be an agreement on the conceptualization that is specified: *ontology is a formal specification of a shared conceptualization.* This notion of shared conceptualization is important to us, because we aim at a common understanding of

business processes shared by different organizations. To contribute to a common understanding, we base our ontology on well known business process concepts.
Another possible definition of Ontology is a characterization of the intended subject matter (class of applications) for a given conceptual model.
Ontology modeling requires the ontology designer to establish explicitly the assumptions we are adopting about the problem domain. These Ontological assumptions would, later on, serve for the establishment of the required "ontological commitments" [65]. This is critical for defining the scope that is the boundaries of the problem universe the ontology is meant to cover and would serve for evaluating the ontology validity once a tentative ontology is designed. We represent the adopted assumptions in the next chapter. In the following, we dissert about the "ontology" concept in general, and afterward we present the ontology design process we followed while evaluating tentative solutions for the ontology extension. The process is an iterative one, improving the solution in each step toward the final solution we adopt.

2.2. Ontology modeling process.

Our research objective is to establish a formal ontological model of the domain of virtual organizations business processes.
Some authors have tried to define methodologies for ontology analysis, specification and validation. [38][34][10][37]. We adopted many of the guidelines proposed by the mentioned authors and established the following procedure for our ontology design:

- *Step 1*: First a generic enough ontology is selected from the literature. This ontology must be relevant for the problem domain that is analyzed, though it does not have to provide a complete model of the problem domain.
- *Step 2*: The problem domain is analyzed and basic concepts are collected. This might be done through requirement & system engineering methodologies.
- *Step 3*: These concepts are then mapped to the generic ontology. At this step, several scenarios may take place:
    - *Scenario 1:* A generic ontology concept is mapped to several domain concepts. The ontology designer must analyze these concepts and check if they are not really the same one.
        - If not, these concepts must be refined further and differences between them highlighted. These differences would later (in the

next step) be modeled by other existing concepts of the generic ontology or through the addition of new concepts to the generic one (this is what we called the "ontology extension").

- If affirmative, then this would result in the reduction of several domain problem concepts into one generic ontology concept.

This is an example of one of the advantages of using a formal ontology based analysis of the problem domain: A well defined model would use of a minimal set of concepts to describe the problem domain (enhancing thus the clarity of the basic domain concepts and simplifying the future applications of the model to real scenarios.

- *Scenario 2:* A problem domain concept has several possible modeling options through the generic ontology. The analyst should then refine the problem domain concept and its relation to other problem domain concepts. This is done in order to identify more constraints that would lead him to the most correct ontological form for each concept and relations to other concepts.

  At this step the analyst may discover that a concept defined previously hasn't been defined correctly and further analysis and modeling effort may be needed in order to redesign it. This is yet another advantage of the use of a formal ontology (related to the correctness and validity of the resulting model).
- *Scenario 3:* A problem domain concept has no possible presentation through the generic ontology concepts. Here the analyst has to analyze the generic ontology concepts to define precisely the gap to be filled. The output of this analysis should be a set of new engineered concepts that form an extension of the generic ontology for the problem domain.

At the end of the ontology model definition, the analyst should have a problem domain ontology that is formed by a subset of the generic ontology concepts plus an ontology extension composed of a set of concepts engineered by the analyst to complete the missing gap between the generic domain and the specific problem domain entities model.

2.2.1. Generic Ontology selection

2.2.1.1. The generic ontology selection process and criteria

As explained in the previous section, the ontology based modeling starts by the selection of a generic ontology that would be extended through the problem domain analysis to the specific domain ontology. The selected ontology should be an ontology which was designed for a domain that has as much similarity as possible to the problem domain to which we intend to apply and extend it.
Myolopoulus et al. [44][46] proposed a classification of ontologies in 4 main categories:

Static ontologies: encompasses static aspects of an application domain by describing what things exist, their attributes and interrelationships, where each thing or entity has an immutable identity for its lifetime. Examples of such ontology may be found in [29].

Dynamic ontologies: encompasses dynamic aspects of an application domain in terms of states, states transitions and processes. We tend to classify the BWW ontological model in this category; though the BWW emphasizes static elements, it addresses states which are time-dependent. GPM is a more dynamic extension of BWW.

Intentional ontologies: encompasses the world of agents, and things agents believe in, want, prove or disprove, and argue about. Agents and Multiple agent systems were studied extensively in Artificial intelligence.

Social ontologies: characterized traditionally in terms of concepts such as actors, position, role, authority, commitment, etc, is meant for domain where the focus in on social settings and organizational structures.

Each Category is based upon different ontological assumptions about its intended application domain, which determine the set of concepts included in the ontology model. This means that the model is not appropriate for applications domains that don't respect these assumptions.
Thus the ontology selection for a specific domain must be based upon the proximity of the application domain to the case to be modeled.

2.2.1.2. The selected ontology: The BWW and GPM models

In our case, the problem domain is virtual organizations business process model specification. Using the previous classification, at a first examination, we see that the options we have are using dynamic or intentional ontologies. Yet dynamic ontologies are more proximate to our problem domain as it treats the world of processes.
In addition, Virtual organization business processes, though may have different characteristics from classical organizations, are expected to have much in common with them.
Hence we postulate that our problem ontology can be based upon generic business processes ontology. A known process ontology that has been proposed by Wand and Weber [88][87]- the BWW (Bunge-Wand-Weber model) provides a model for information systems as described in the next section. Their work is based upon a more generic ontology, that of the philosopher Mario Bunge. We present the BWW in the next section. Furthermore, we will use an extension of the BWW model, the Generic process model (GPM), defined in [81], which we present in the following section.
In the literature, some authors preferred to design their specific ontology. As an example, in [47] the Authors model Domain knowledge of legal business contracts using a multi-tier contract ontology. There ontology is composed of three layers, each one being built upon the lower one. The lower layer defines generic concepts such as role, consideration, obligation, etc. The second one defines templates for specific domain contracts and the upper ontology layer is a template level contract ontology, in which the authors put all contractual templates of the specific domain contracts specified in the lower ontology layer. Thus each layer builds upon the lower layer in order to complete the concepts and concepts relations. Though the ontology is in its whole specific for the problem domain the authors are modeling, in each layer, the authors have borrowed already designed models, for example the upper layer uses established contract models, such as the international-chamber of commerce's contract model for international Sale of goods. The author considered that it would be impractical to represent all types of business contractual obligations by a single ontology and postulate that a layered structure provides the scope for defining an individual ontology for specific types yet coherently within one unified framework.

In this example, we must consider two main issues:

(1) If the overhead of the introduced complexity is justified in relation to the extensibility and flexibility of the proposed model.
(2) Furthermore the analyst faces an increasing complexity when trying to identify of overlapping concepts in order to simplify the model; this may result in a greater potential of errors and concepts redundancy in the resulting model.

In our case, the GPM model may be considered as an ontological extension of the BWW more generic model. The major differences with the above mentioned example is that the GPM and BWW models are highly integrated, as the GPM constitutes a direct extension rather than an over-layering of an ontology upon the other. The model concepts are kept minimal in order to ease the identification of concepts redundancy (overlapping) and missing constructs.

2.3. The BWW model

2.3.1. Background

Wand and Weber [88][87] have extended an ontology presented by Bunge [12][13], and applied it to the modeling of information systems. Bunge's ontology presents a set of high-level, abstract constructs that are intended to be a means of representing all real-world phenomena. His work relies on the philosophical foundations of Aristotle, Aquinas, Descartes and others. Wand and Weber have based their work on Bunge's work because of its rigor and comprehensiveness.

According to the ontological framework, the world is made of *things* that possess *properties*. Properties can be *intrinsic* (e.g. height) to things or *mutual* to several things (e.g. a person works for a company). Things can compose to form a *composite* thing that has *emergent* properties, namely, properties not possessed by the individuals composing it. Properties (intrinsic or mutual) are perceived by humans in terms of *attributes*, which can be represented as functions on time. The *state* of a thing is the set of values of all its attribute functions (also termed state variables). When properties of things change, these changes are manifested as state changes or *events*. State changes can happen either due to internal transformations in things (self action of a thing) or due to *interactions* among things, which is achieved through their mutual properties. Not all states are possible, and not all state changes can occur. The rules governing possible

states and state changes are termed *state laws* and *transition laws*, respectively. States can be classified as being *stable* or *unstable*, where an unstable state is a state that must change by law, and a stable state is a state that can only change as a result of an action of something external to the thing or the domain.

2.3.2. The BWW Main concepts

Hereafter, we summarize the main BWW ontological constructs, which are relevant to our scope of analysis. Note that these are a subset of the BWW constructs, a more complete list can be found in [87].

Thing. The world is made of things. According to this fundamental premise of the BWW ontology, things are the elementary real-world units. A thing in the BWW ontology is a concrete substance that exists in the real world.

Property. Things possess properties. A property inherently possessed by an individual thing is termed intrinsic. A property that is meaningful only in the context of two or more things is called mutual.

Attribute. Attributes are representations of real-world properties, and thus properties of the models of things, as viewed by people. Attributes map properties of a thing into values, and may be expressed as functions.

Class. A subset of things is called a class if and only if a property exists, such that the subset is equal to the set of things that possess this property. The common property may be a composite one, encompassing a set of behavioral and characteristic properties of the things.

Functional schema. Let T be a set of things, all possessing a common set of properties. A functional schema Xm = <M,F> is a non-empty set M and a finite sequence F = <F1, . . . Fn> of functions defined on M; that is, Fi:M->Vi is a domain of values, and each attribute function represents a property of the thing. A functional schema, in other words, represents the set of attribute functions, associated with a class. The functional schema concept is the basis for the definition of a state.

State. A state is the set of attribute values of a thing at a specific point in time. Consider a thing X described by a functional schema <M,F>. The function Fi:M->Vi is termed the ith state function of the thing. The set of values F(t) = <F1(t) ... Fn(t)> at a certain time point t represents the state of X at t. (Note, recall time is a part of M.)

Interaction. Thing X acts on thing Y if and only if the states that Y traverses for a given subset of M when X is present are different from the states that Y would traverse if the thing X did not exist. Things X and Y interact if at least one acts on the other. Interaction emerges naturally from mutual properties of the things involved. Unlike intrinsic properties of each thing, which are modeled by its attributes, mutual properties are not necessarily modeled by attributes of the things involved. Rather, they may be observed by tracking interaction patterns. Interactions were defined in terms of state changes, or events, which are brought about by transformations. These concepts are defined as follows:

Transformation. A transformation is a mapping from one state to another one.

Event. An event is a change of state of a thing, affected via a transformation. Events may be further divided into external events and internal events.

External event. An external event is an event that arises in a thing by virtue of the action of some other thing.

Internal event. An internal event is an event that arises in a thing by virtue of a transformation in that thing itself.

Stable state. A state in which a thing will remain unless forced to change by virtue of the action of another thing (i.e. by an external event).

Unstable state. A state in which transformations of the thing will occur until a stable state is reached.

Relating events to states of a thing, an external event may cause an unstable state, which would lead to a sequence of internal events that continues until a stable state is reached. This is the mechanism that drives a process.

State law. A state law is a law which restricts the values of the attributes of a thing to a subset defined by natural or human laws. Note, a state law reflects a restriction on the properties of a thing.

Transformation law. A transformation law is a law which defines the lawful transformations that can happen to a thing, that is, the lawful events of the thing.

Subclass. A subset of things X is a subclass of another set of things Y if and only if X is a class and a proper subset of Y. Subclasses are defined by a conjunction of the property used initially to define the class and an additional property of interest.

Composite thing. A thing is composite if and only if it consists of at least two things. A property of a composite thing may be emergent, i.e. it not possessed by any of its components, otherwise it is hereditary.

Composition. The things comprising a composite thing are its composition.

Decomposition. A decomposition of a composite thing is a set of things, such that every component of the composite thing is either a member of this set or is included in the composition of one of the members.

2.4. The Generic process Model (GPM)

2.4.1. Background

The Generic Process Model (GPM) [[81][82]], which is based on BWW ontology, is a specialized framework that extends the ontology for the purpose of process modeling. It includes criteria for validity evaluation of process models, where process validity is defined as the possibility of the process to achieve its goal. It extends BWW model with concepts that provide a formal basis for expressing processes in ontological terms.

2.4.2. The GPM main concepts

The GPM extends the BWW with the following main concepts:

Domain. According to GPM, a *domain* is a part of the world of which we wish to model changes, representing the scope of our control. In ontological terms, a domain is a set of things and their interactions, and is represented by a set of state variables, which stand for the intrinsic and mutual properties of these things, including emergent properties of the domain itself.

Note, the state of the domain is determined by the states of the things included in it. However, due to interactions, emergent state variables of composite things or of the domain might exist.

A *sub-domain* is a part of the domain, represented by a subset of the domain state variables. A sub-domain may be in a stable state while the entire domain is in an unstable state, meaning that a different part of the domain is currently subject to changes.

A domain state is stable if it can only change as a result of an action of something outside the (sub)domain.

A domain state is unstable if it is a state of the (sub)domain that must change. Whether a state is stable or unstable and how an unstable state might change is defined in terms of the *laws* that govern the states of the domain and their transitions:

A criterion function is a function on the set of states C: S $\rightarrow$ D, where D is a certain domain (of values). A criterion function maps the values of state variables into a domain where a decision can be made on whether the process achieved its purpose or not. Often, the mapping is on a subset of state variables that are considered relevant for deciding whether the process has reached its "goal". The domain mapped into is then a sub-domain of the process domain.

A law is a function from the set of states S into itself.

A transition law *is* a function on the set of possible unstable states S_u into the set of states S. Implied in this definition is that the transition law is fully deterministic. However, the GPM allows for uncertainty in how the process will progress when enacted. This is because external events may affect the state of the domain while the process is in progress. Consequently, state variables that affect the law might change in ways not controlled in the process.

A transition law can be extended to all states as follows: for an unstable state the law is the transition law, otherwise it maps the state into itself. These extended laws are also referred to as domain laws (designated by L).

In general processes may be viewed as sequences of unstable states that terminate on stable states. It is not guaranteed that a domain law will always lead to a stable state. This justifies the need for a condition under which every process will terminate (i.e. the domain will reach a stable state).

Goal. In order to define formally this concept, the GPM establish the following ones: Let S={s | s lawful} be the set of possible domain states. Let $S_{st} \subseteq S$ be the subset of domain stable states. Then a *Goal* (G) is a set of stable states $G \subseteq S_{st}$.

A *process* is a sequence of unstable states, transforming by law until a stable state is reached. A process is defined over a domain, which sets the boundaries of what is in a stable or an unstable state. Events that occur outside the domain are *external events* and they can activate the domain when it is in a stable state.

Moreover, The GPM postulates that a goal G will be said to be a *process goal* if every execution of the process terminates in G.

A process model in GPM is a quadruple <S, L, I, G>, where S is a set of states representing the domain of the process; L is the law, specified as mapping between subsets of states; I is a subset of unstable states, which are the initial states of the process after a triggering external event has occurred; G is a subset of stable states, which are the goal of the process. Subsets of states are specified by conditions defined over criterion functions in the state variables of the domain. Hence, a process starts when a certain condition on the state of the domain holds, and ends when its goal is reached, i.e., when another condition specified on the state of the domain holds. The states in the goal set may differ from each other in the values of state variables such as production time and cost. Nevertheless, they all meet the condition specified. The criterion function defines the set of state variables that are relevant for determining that the process has reached its goal.

Note that by "goal" we relate to an operational goal of the process only, as opposed to business goals (also called soft-goals) of the organization.

A state can be projected on a sub-domain by considering the sub-set of state variables describing the sub-domain. This subset defines the state of the sub-domain.

A law L can be projected on a sub-domain.

Consider a process defined over a domain. Let $\underline{x}=x_1 \ldots x_n$ be the set of state variables of the domain. Let $\underline{x}_0$ be a subset of $\underline{x}$. The projection of a law, L, on $\underline{x}_0$ is the mapping L defines on $\underline{x}_0$ when operating on $\underline{x}$.

In other words, the projection of the law over a sub-domain defines the way the process will progress in that sub-domain not considering other parts of the domain. As an example, consider the law projection of a production process over a quality inspection sub-domain. The sub-domain is in a stable state through most of the process. It becomes unstable when a quality inspection of the product is required, and stable again when the product quality has been determined (either approved or disapproved).

A *condition* is a logical expression E made of simple expressions of the form: R::= C rel g, where rel$\in$ {'>', '=', '<'}, where C a criterion and g is a value from the same domain as C, combined by 'AND', 'OR', 'NOT' and precedence indicated by '( )'.

This leads to the definition of a goal as: G ={s | E(C(s)) is 'true'}.

Soft-goals. While a process in GPM must have a goal on which it terminates, it is likely that not all states in the goal are equally desirable. For example, different paths of a production process, all leading to the same goal, may differ in time or in cost. The most desired state in the goal set is the one whose production cost and time are the minimal.

The desirability of different goal states is addressed by GPM using the concept of a *soft-goal*, which is defined as an order relation on states [82].

By defining a soft-goal as an order relation, states can be ranked according to their desirability. This ranking is called a "soft-goal" since while the notion of "goal" implies something that can be attained, achieving a "better" state is a matter of improvement, not of accomplishment. Such an improvement is referred to in the RE literature as "satisficing" the soft-goal [96]. With respect to business processes, soft-goals are usually a result of some performance measurements thus they express business objectives. The definition of soft-goal is operationalized in a similar way to the goal definition, as an order relation on a criterion function. The soft-goal criterion function would usually be different than the goal criterion function. Furthermore, while a process clearly possesses a single goal, it may be related to several (even contradicting) soft-goals.

3. Related Work

3.1. General

Our work is centered on establishing a formal ontology based model for virtual organizations (VO). A moment before diving in the analysis and design of our objective model, we review existing literature about established models, characteristics and process engineering for virtual organizations. We give a special focus on the contractual obligations and privacy in VO's for its special relevance to our model. Finally, a review of relevant literature about ontology based modeling is provided in the next chapter where we introduce the basic building blocks of our model.

3.2. VO models & characteristics

Porter [70] postulated that a virtual organization is "a collection of business units in which people and business units processes interact intensively in order to perform work which benefits all". Although virtual organizations have become a relatively widespread business approach to structuring business, the underlying concepts of linking capabilities across business units or organizations have existed for some time earlier, too. These inter-business relations enable organizations to more tightly coordinate the transactions and activities across a value chain.

The rationale for forming a virtual organization varies for the different entities involved in each relationship [41]. This desire to excel in a market characterized by increasing competition has motivated a growing number of organizations to search for inter-organization associations to help develop effective strategies. It is largely postulated that virtual organizations are able to generate new products more quickly, decrease the risk of pursuing a new opportunity, by relying on synergies of the core competencies of all their memberships [9].

Porter [69][70] also identified several characteristics of virtual organizations: 1) A web of companies each contributing resources, 2) Virtually vertically integrated, 3) Linked through inter-organization business and production systems, 4) Aimed at reduced business cycle time, and 5) Aimed at one-stop shopping.

Several VO characteristics have been investigated through industry reviews and statistical analysis of the results by [74], differentiating between four structural dimensions: differentiation (based upon Modularity and Heterogeneity), configuration (as a Temporary and Loose Coupled Network), integration (using Trust as Coordination Mechanism), and technology (as a measure of how efficient is the coordination of activities). The differentiation dimension is represented by two factors (a) the Virtual value creation, which measures the extent of modularization of the value creating process, (b) the focus on core competencies, which measures the extent to which firms focus on their core competencies. The temporary and loosely coupled network dimension is represented by (a) General characteristics of the network, which includes descriptive aspects like the duration of the cooperation, the selection and combination of the cooperation partners, the configuration of the cooperation, appearance towards the customer, etc., (b) The Independence of cooperation partners measures the degree of horizontal and vertical independence between the cooperation partners; (c) Formal or contractual commitment between the cooperation partners measures the extent to which contracts, rules or regulations are used. The Integration mechanisms includes (a) Trust as coordination mechanism, which measures the general atmosphere as well as trust and fairness inside the network; (b) the Technology dimension, which is concerned about implementation of information and communication technology, and measures which and to what extent communication and computer systems are used for facilitating the cooperation. The author uses these dimensions to establish what he calls DV (degree of virtuality) of an organization. In summary, two distinct approaches are presented for the purpose of the DV estimation. The first approach suggests a predetermined ICT-driven path through different stages of evolution from a non-virtual to a virtual organization. Organizations can be classified according to their actual stage of development and an associated DV. An alternative approach proposes an “ideal” Virtual organization exhibiting predetermined characteristics as a reference for measuring the DV. The DV is conceptualized independently of a specific evolutionary path. Based on this analysis, the author establishes a VO definition as a "temporary, loosely coupled network of legally independent companies, who combine their individual core competencies to exploit a specific business opportunity by optimizing the value adding business process". Mutual trust between the partners and the extensive use of information and communication technology guarantee the coordination of modularized production.

A distinction has been established by several authors between static VO's and dynamic ones [45] [52]. The two types are basically differentiated by the type of coordination mechanisms that can be applied efficiently. While contract based coordination may function well for static (long-term) VO's it is unfeasible in dynamic ones. In a dynamic VO, in some cases trading partners may rely on mutual trust to regulate and coordinate their activities, simply because there is not enough time for contract implementation prior to the beginning of a shared business process [52].

The human based coordination is another considered coordination form, especially when negotiation is required with unknown partners. A regulation based approach has been proposed by [35], where regulation and policies support effective human based collaboration. This may be adopted in VO's.

Other alternative coordination mechanisms for inter-organization collaboration have been proposed at the process implementation level; an example of such alternatives is the web services based approach (within the Service oriented architecture framework) [91], which is a highly accepted framework whose goal is to achieve loose coupling among interacting systems. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by agents on behalf of their owners.

A similar methodology may be applied for modeling the dynamic VO's business processes as a set of coordinated application services, offered by different companies that can be wrapped and presented as independent services that, in turn, could be further composed to create new application services. Here the responsibility of coordinating the providers participating in composite service execution is distributed across several lightweight software components hosted by the participants themselves. However, the coordination support is still limited.

Referring to dynamic VO's, Mowshowitz [62][63] considers that switching is the key managerial innovation of the virtual organization. The concept of switching consists of a managerial approach that enables firms to identify and select the best or most appropriate means for satisfying a need. A continuous evaluation of needs and the means to satisfy those needs implies that the principle of switching enables an organization to be responsive to changing conditions. According to Mowshowitz, four components of managerial activity are essential to the virtual organization: formulation of abstract requirements (e.g. customer needs or orders); tracking and analysis of

concrete satisfiers (e.g. suppliers); dynamic assignment of concrete satisfiers to abstract requirements, based on an allocation procedure; and exploration and analysis of the assignment criteria.

In their dynamic VO model, Shao et al. [78] argue that what maintains the existence of a specific VO may be characterized by four dimensions: purpose, connectivity, boundary, and technology. Purpose provides the incentive for creating the new organization and serves as the cohesive force to hold the components at least temporarily together. Connectivity defines the leverage of co-operation. It may be a consequence of the sharing of physical assets, resources, intellectual and knowledge assets, or access to markets. Boundary indicates who is part of the virtual organization and who is not, in the absence of any clear visible physical or legal borderlines. It defines who can share its activities and receive benefits. Finally, they consider technology the enabling factor that allows for the virtual organization.

Camarinha-matos et al. [18], postulated that the coordination issues in an infrastructure for industrial VO's require a very flexible and easily configurable solutions. He enumerates a large number of factors that justifies this need. Some of these factors which the authors call "diversity factors": (1) Diversity of VO classes, in terms of duration, composition and topology, coordination policy, visibility scope, etc.; (2) Diversity of roles played by each organization, such as the VO member, VO coordinator, client, supplier, etc. (3) Diverse motivations for creation of the VO: decomposition of a large company into smaller units or aggregation of small firms into a larger (virtual) Organization; (4) Diversity of internal management policies and socio-organizational structures found in each company, which also depend on the size of the companies; (5) Diversity of rights and duties (e.g., to share and exchange organization's local information) that can be associated to every other VO member; (6) Diversity of contract/subcontract forms that regulate the cooperation among the members of the VO;(7) Participation of a company in multiple VO's, with different roles, rights and duties towards each VO; (8) Evolution of support technologies, safety mechanisms, and the legal framework for electronic commerce; (9) Dynamic forms of interaction and cooperation between organizations, that are likely to evolve with time, experience and trust building mechanisms. Based on acquired experiences it is expected that new organizational paradigms for industrial companies will emerge.

gathered the main general functional and information requirements for VO implementation. The main requirements are related to (1) Functionalities related to

Creation/Configuration/Reconfiguration phase of a VO. Steps involved in this phase support both the initial phase when a VO is being created, and any possible future variations, considering its configuration along its life cycle; (2) Functionalities related to operation phase of a VO: once the VO is established and operational, several interaction levels among the members must be supported; (3) functionalities are required to represent and monitor the flow of products/services through the VO. These include: material/services flow management, routing planning, transport assignment, distribution, and forecasting, logistics; (4) Information handling mechanisms and interaction protocols that are required to support the co-working and cooperation among the VO members; (5) Support for new emerging technologies and tools in the market such as electronic commerce.

3.3. Virtual organizations modeling initiatives

A number of projects, worldwide, tried to address diverse aspects of virtual organizations. Some of the most representative ones are the North-American NIIIP [84] and the European VEGA [99], X.CITTIC, PLENT, MARVEL OUS and PRODNET II [[5][6][18][19][20][21][22]] .

NIIIP (National Industrial Information Infrastructure Protocols) is a USA program [84] that intended to support the formation of industrial VO's and to provide technologies that would allow VO's partners to collaborate within a heterogeneous computing environment. In its general scope, NIIIP addresses the complete VO "Life Cycle": identifying market needs; looking for partners; assisting the negotiation process between partners; and supporting VO instantiation, operation and dissolution. Within NIIIP's conceptual model, all organizations would work cooperatively, sharing all kinds of resources, including the human ones, which in the opinion of [6], seems to be too optimistic and not in compliance with the current reality in most Small and medium enterprises (SMEs).

X-CITTIC (Planning and Control System for Semiconductor Virtual Enterprises) is an European project focused on VO's for the microelectronics sector. In this application domain, the manufacturing process associated to a sales order, originated in a customer located anywhere in the world, may be accomplished by manufacturing orders allocated through a globally distributed manufacturing network. X-CITTIC rose to the VO level,

some of the techniques currently available in a modern shop floor. Examples of such techniques are event-driven planning, scheduling, dispatch and orders release.

The MARVEL OUS European project, addressed the identification and harmonization of generic requirements for use of advanced information technology in manufacturing and engineering across the maritime industry, and building up links to other sectors for mutual benefit. The project intended to guarantee consensus on requirements across the whole range of maritime users and to work closely with the technology providers in order to facilitate the formation of VO's. It also tried to ensure that the end-user requirements are feasible and can be translated into product development.

The European VEGA (Virtual Enterprise using Groupware tools and distributed Architecture) project [99] aimed to establish an information infrastructure to support the technical and business operations of Virtual Enterprises. Groupware tools and distributed architectures are being developed in compliance with product data technology standardization activities and the current trends adopted by international industrial groupware specifications coming, for example, from the Object Management Group. The approaches and developments resulting from a number of ESPRIT projects were meant to be extended and the strategy for application integration by the distribution of a concurrent access to STEP databases was explored. A complementary route involves the design of a CORBA Access to STEP models (COAST) infrastructure to support, as a natural mechanism, the distribution of a product data by means of updated object broker technology.

PLENT intended to develop a set of innovative software tools to support coordinated planning in networks of autonomous small and medium size manufacturing organizations. In the context of this project, the most important problem is not the study of a sophisticated decision-making technique but the definition of a planning policy that is based on well defined, completely visible and strictly applicable rules. This is considered necessary in order to remove the historical distrust between organizations traditionally in competition with each other, so as to reach an adequate degree of confidence in the network organization.

Finally, the European project PRODNET II [[5][6][18][19][20][21][22]], aimed to design and develop a VO reference architecture and an open platform to support industrial virtual enterprises/organizations with special focus on the needs of small and

medium size enterprises (SMEs) that may join efforts to form and function as a Virtual Enterprise. The PRODNET platform provides for the partners interoperability and exchange of specialized information, in real time in such a way that the organizations can function together as a single integrated unit, while individually preserving their independence and autonomy. Although the computer network is the basic enabling element for this platform, a large number of challenges and open issues are yet left unresolved, such as the definition of flexible reference architecture, the cooperative information management, supporting tools for the advanced coordination and life cycle support for virtual organizations, etc. The main VO's characteristics that the PRODNET designers considered of special interest to the design and development of PRODNET system are the heterogeneity and autonomy of pre-existing nodes, possible loose- and tight-coupling among the nodes, proprietary vs. sharable node information, incomplete and imprecise orders and order status monitoring among the nodes, virtual organization coordination, need for appropriate support tools for specific file transfer, supporting tools and mechanisms for data exchange and communication safety and authentication.

An important assumption within the PRODNET is that within the VO, typically one node (VO coordinator) assumes the role of the coordination of VO activities, and the others (VO members) will report to the VO coordinator on their performance, order status, and achievements towards their contracts. Node architecture, as defined in the PRODNET system is composed of the “Internal Module”, the “PRODNET Cooperation Layer (PCL)” and an “Advanced VO Coordination” module.

The project designed and developed a prototype of an execution infrastructure for distributed BP's.

Another important design assumption within the PRODNET is that the coordination issues must be analyzed at different levels of abstraction. No proof is presented for the necessity of a multilevel abstraction. This model adds, from our point of view an unjustified complexity to the proposed model. PRODNET proposed a process coordination subsystem that has three levels of abstraction: (1) the Core Cooperation Layer (CCL) which is responsible for the basic interactions among VO members offering support for safe communications, exchange of business messages, sharing and management of cooperation information, federated information queries, etc; (2) The Enterprise Management Functionalities (EMF) which deals with coordinating the responsibilities of the organization towards the accomplishment of its assigned BP's or

contracts with the VO and other VO-partners; (3) The Virtual Organization Management Functionalities (VMF) responsible for the coordination aspects at the VO level. In principle, only the node playing the VO coordinator role will use this layer to monitor, assist, and modify the necessary activities related to the VO goal achievement. The VO Management Functionalities (VMF) enables the services provided by the CCL and EMF of its node to communicate with the other nodes of the VO. The model can be generalized to any number of levels in order to cope with any BP tree. This is possible due to the fact that, in addition to the VO coordination role, responsible for the global business process, some participants of the VO may assume the role of coordinators of sub-business processes that might be decomposed and performed by a sub-consortium of organizations. These sub-consortia are formed for the sole purpose of facilitating the coordination of activities involved in the related sub-business processes. Once a sub-business process ends, the sub-consortium is dissolved and its members may become involved in other sub-consortia dynamically formed as the execution of the VO's business process evolves. Under this model, the formation of a sub-consortium inside a VO follows a similar process as the formation of the VO itself, thus allowing recursively creating internal sub-VO's.

3.4. Contracts & privacy within VO's

The topics of inter-organizational processes, and contracts related to such processes have received considerable attention in recent years.

An example of such effort is [47][49][48][100], which is a series of works that address legal contracts between parties (organizations). They present a three-tier contract ontology, where the upper-level presents generic contract concepts, the middle level specializes the upper level for specific domains, and the third provides templates or patterns for specific obligations, and specific document templates. A key element in the upper-level ontology is an obligation. They differentiate obligation types and states, and define a contract workflow model that follows the life-time of an obligation. Note that an obligation is anchored in the legal document, and may be expressed in rather general terms (e.g., the seller will provide goods to the buyer), unlike the detailed specification of our model. For specific cases they show how a detailed workflow can be derived from the contract. They also provide rules for matching the contract-based workflow with the existing organizational business processes.

[4] proposes private and public domains for modeling the inter-organizational aspects of workflows, while the interacting partners are not exposing their whole business process.

[92] asserts that the collaboration flow design required in inter-organization processes, must not constrain the autonomy of each participating organization in choosing its partners and defining its own private implementation of the needed collaboration infrastructure. The author considers that a complete visibility of the processes where collaboration is required, constrains heavily the autonomy of each partner once the inter-organization flow is defined. From here he defined what he calls a "relative workflow" which allows each VO partner organization to define its collaboration process independently providing to external entities only a partial (relative) view of its internal processes, to each one of its partners on a need to know basis. In this manner each organization is aware of the inter-organization flow progress but in such manner that privacy and autonomy of each partner is conserved.

The privacy issue in inter-organizational process management motivates the workflow views suggested by [50] in relation to contracts. According to their definitions the contract between organizations defines the workflows of the participating organizations and the communication between these workflows. This allows the organizations to reveal only parts of their workflows in run time. However, the entire workflow has to be defined at build time, when the contract is established.

Another project that deals with workflow management in a VO is the CrossFlow project [36], which proposed a comprehensive framework and infrastructure for a variety of phases in a VO life-cycle. The support starts from partner identification, through contract establishment and negotiation, infrastructure configurations, contract enactment, monitoring, and finally termination. The contract specifies not only the interface between the organizations, but also the process of the “service provider” partner, although the specification is at a high level without all the details. When the contract is enacted and monitored the other partner is able to view and monitor parts of the provider’s workflow, but visibility may be limited according to the contract specification. Quality of service issues are also addressed determined as part of the contract and monitored afterwards.

The Workflow Management Coalition (2001) [89] attempted to define standards for workflow interoperability. However, the real issue here is not to connect technical systems but to develop fundamentally new concepts and architectures to support

execution and management of inter-organization processes. In inter-organization workflows, the business partners and all tasks of a shared business process are still specified statically and in advance making this concept more suitable for static virtual organizations.

[53] discussed contracts versus trust as enablers of VO's, where short life-time is a major motivation for speeding up the contract negotiation. They propose a workflow system for negotiating a contract, but while providing details of the implementation of such workflow system, the content of the contract is not addressed in detail. The parties agree on a work product to be delivered, but a business process is not defined. [25] proposed contract templates to facilitate e-negotiations and speedy agreements. The templates relate to obligations, permissions, and prohibitions and identify contract parameters that need to be agreed on through negotiation. The templates do not relate to the business process aspect of the relationship between the organizations.

3.5. Business process modeling approaches

3.5.1. Business process characteristics

Business process modeling has been investigated a lot in the last decades from different perspectives: business management perspective [40][39][69], organizational structure [70] perspectives and strategic planning perspectives, social, cognitive and organization sciences are only some of them.

In the 1990s, the work of Michael Hammer and James Champy, [40] authors of the widely read Reengineering the Corporation, focused attention on business processes, both as a root cause of inefficiency and as the source of potential competitive advantage. They advised a deep and radical redesign of the business to root out waste and increase the focus on the customer. What is different today is the novel use of computing technology to drive the analysis and automation of business processes. In Beyond Engineering, a follow-up to Reengineering the Corporation, Hammer defines a business process as "a complete end-to-end set of activities that together create value for the customer."[39]. The notion of "customer" in this context refers to the recipient of the value provided, not necessarily a paying customer in a commercial transaction.

Medina-Mora et al. [72] categorize processes in an organization into material processes, information processes, and business processes. The scope of a material process is to assemble physical components and deliver physical products. That is, material processes relate human tasks that are rooted in the physical world. Such tasks include, moving, storing, transforming, measuring, and assembling physical objects. Information processes relate to automated tasks (i.e., tasks performed by programs) and partially automated tasks (i.e., tasks performed by humans interacting with computers) that create, process, manage, and provide information. Typically an information process is rooted in an organization's structure and/or the existing environment of information systems. Database, transaction processing, and distributed systems technologies provide the basic infrastructure for supporting information processes. Business processes are market-centered descriptions of an organization's activities, implemented as information processes and/or material processes. That is, a business process is engineered to fulfill a business contractual obligation or satisfy a specific customer need. Thus, the notion of a business process is conceptually at a higher level than the notion of information or material process. Once an organization captures its business in terms of business processes, it can reengineer each process to improve it or adapt it to changing requirements. Reasons cited for business process redesign include increasing customer satisfaction, improving efficiency of business operations, increasing quality of products, reducing cost, and meeting new business challenges and opportunities by changing existing services or introducing new ones. Business process reengineering involves explicit reconsideration and redesign of the
business process. It is performed before information systems and computers are used for automating these processes.

According to Bider [8], the most common methods of modeling business processes are grouped in four categories based on the way they reflect the business process dynamics: The first one is an Input/output flows oriented approach, where the focus is on passive participants that are being consumed, produced, or changed by the activities. A typical notation to represent this kind of flow is IDEF0 [43]. The second, a Workflow oriented approach, which focus is on partial time ordering of activities performed by active participants. Typical notations to represents this kind of flow are IDEF3 diagrams [60], Petri-nets [73], and activity diagrams of UML [76]. The Agent-related workflows build on agent cooperation, adding an agent dimension to the time dimension of an ordinary workflow. A typical notation to represent this kind of flow is Role-Activity Diagrams

[67]. Finally, the State-flow approach models changes produced by activities executed in the frame of a given process instance. Some changes may concern the state of passive participants, e.g., their form, shape, or physical location. Other changes may concern the state of active participants, e.g. a state of the mind of a human agent trying to find a solution for a complex problem. An example of a state-flow notation is IDEF3 state-transition diagrams [60]; these diagrams are complementary to the IDEF3 workflow diagrams. However, the state-transition diagrams exploit the state-flow view only partially.

Other authors postulate that there are two basic categories of process modeling methodologies: communication based and activity-based.

The communication-based methodologies stem from Winograd/Flores "Conversation for Action Model" [83].This methodology assumes that the objective of business process re-engineering is to improve customer satisfaction. It reduces every action in a workflow to four phases based on communication between a customer and a performer: (1) preparation - a customer requests an action to be performed or a performer offers to do some action; (2) negotiation - both customer and performer agree on the action to be performed and define the terms of satisfaction; (3) performance - the action is performed according to the terms established; (4) acceptance - the customer reports satisfaction (or dissatisfaction) with the action. Each workflow loop between a customer and performer can be joined with other workflow loops to complete a business process. The performer in one workflow loop can be a customer in another workflow loop. The resulting business process reveals the social network in which a group of people, filling various roles, fulfill a business process. Since this methodology assumes that the objective of business process re-engineering is to improve customer satisfaction, the emphasis is on the customer. However, there are business processes where the customer emphasis may be superficial, e.g., if the objectives are to minimize information system cost or reduce waste of material in a process. Therefore, this methodology is not appropriate for modeling business processes with objectives other than customer satisfaction. Another limitation is that this methodology by itself does not support the development of workflow implementations for specifications.

Activity-based methodologies focus on modeling the work instead of modeling commitments among humans. Unlike communication based methodologies, activity-based methodologies do not capture process objectives such as customer satisfaction.

3.5.2. Information and its relationship to process and organization

Process, information and organization are inexorably linked; one can approach an architectural model from any of the three dimensions but for coherence all three must fit together. Process-based architectures tend to emphasize process as the dominant dimension; processes consume, generate or transform information, behaving in accordance with a set of corporate governance rules. By contrast, information based architectures emphasize the information dimension, viewing processes as operations that are triggered as a result of information change.

3.5.3. The business process management lifecycle

The BPM application to real scenarios includes four major steps in the BPM lifecycle: Process discovery is the beginning of any BPM solution and is necessary to ensure that the solution matches the real business needs. The second phase includes the activities of design, analyze, and simulate business processes. Several Process languages, such as BPEL & BPML [15] have been proposed though the lack for process laboratory makes them less useful than predicted for this phase. The third phase is integration that links the business process design and business process execution. Newly business process languages such as the BPEL and BPML may play a key role in this integration. The last phase is the Business process monitoring [61], which closes the lifecycle loop, generating valuable performance statistics from executing business processes. Businesses need to monitor these execution statistics, organize them into their process context, and present them in the form of alarms, reports, and executive dashboards.

The software packages market offers a lot of software tools for modeling processes, designed for analyzing, designing, simulating and executing automatically processes. Process modeling (ARIS-SHEER, for example [79]) tools allow business users to coordinate business activities, people and applications, and to model routing of work requests within a process and across processes. The model can depict various aspects of a process, including automated and manual process activities, decision points and business rules, parallel and sequential work routes, and how to manage undesirable states to the normal business process.

Furthermore, major software packages like SAP R/3 (and posterior releases) and BAAN ERP use process models to visualize the predefined processes that companies

implement. As a result several process modeling languages have been developed, such as the BML- Business modeling language [86], EPC - Event driven Process chains [79], IDEF0 [43], IDEF3 [60], Specification and description language [7], Role –Activity diagram (RAD) [98], Task structures [3], UML activity, sequence and state diagrams [76].

3.5.4. Business process management standardization

3.5.4.1. Process execution languages: XPDL, BPML, BPEL

Process execution languages are generally not used directly in analysis and design phases. Being expressed in XML syntax, they have a native exchange format. None of these languages offers a standardized graphic notation. By definition, they are not designed to cover the levels of value chain and organization analysis.

The first execution standard was developed by the Workflow Management Coalition (WFMC). A new XML version of the WFMC language was released in 2002 under the name XPDL.

The Business Process Management Initiative group (BPMI) released a competing language in 2001 called the Business Process Markup Language (BPML). This initiative restarted the work on process execution languages and made many contributions to its successor, BPEL.

BPEL (Business Process Execution Language) was initiated by Microsoft and IBM in response to the BPMI initiative. Since that time, this language has received the support of most market players, including BPMI. BPEL has become the de facto standard for business process execution. It lies on top of the web service specification stack. Since 2003, the standardization organization OASIS has been in charge of the evolution of the BPEL language.

3.5.4.2. UML 1.x

Proposed by OMG for object-oriented design, UML 1.X (1.1-1.4) provides an activity model that offers limited functionalities for business process modeling. It also offers a meta-model, a notation and an exchange format for models with XMI 1.x. However, its scope remains limited to object design and its meta-model has some semantic errors (activity = state) that reduce its operational effectiveness. These weaknesses have been recognized by the OMG, which has reviewed this model in depth in UML version 2.

3.5.4.3. UML 2.0

The Open Management group's (OMG) Unified Modeling Language (UML) 2.0 specification has completed at the end of 2004, and is the product of a lengthy development period. Activity models in UML 2.0 have been completely reworked from the 1.X versions. The main errors in the 1.X specifications have been corrected, and the new model offers a robust base for process analysis. However, its technical nature makes it still primarily suited for business process automation. In its current state, UML 2.0 activity models cannot provide a comprehensive support for dedicated business process analysis. OMG is aware of these limitations and has launched a complementary initiative, BPDM, specifically to handle business processes (see paragraph below on BPDM).

3.5.4.4. BPMN – Business Process Modeling Notation

The Business Process Modeling Notation (BPMN) specification is one of the most modern standard options that provide a (graphical) notation for expressing business processes in a Business Process Diagram (BPD).

Following the bpmi.org, the drivers for the development of BPMN is to create a simple mechanism for creating business process models, while at the same time being able to handle the complexity inherent to business processes.

BPMI released the process notation (BPMN 1.0) specification [42] on May 2004, after the BPML execution language release. BPMN provides a graphical representation of business processes. It specifically introduces the notions of messages and information flows that were lacking in most traditional process representations (IDEF, SAP EPC). The objective of BPMN is to support business process management by both technical users and business users by providing a notation that is intuitive to business users yet able to represent complex process semantics. Though, BPMN 1.0's scope is limited to business process automation excluding organization analysis and value chain analysis. One of BPMN's primary goals in its version 1.0 was the possibility of representing executable processes; therefore, the BPMN specification was designed to provide a mapping between the graphics of the notation to the underlying constructs of execution languages, particularly BPEL4WS. Furthermore, rules are provided for correlation with the BPEL execution language complete schema.

The BPMN 1.0 specification addresses only the issue of notation: BPMN includes neither a meta-model nor an exchange format. It would thus be difficult to discuss exchanging BPMN models. Therefore, a new version, BPMN 2.0, has been released, in order to satisfy the need for organization and value chain analysis, by providing the definitions for a meta-model and exchange format.

3.5.4.5. Process meta-model specification: BPDM, BSBR

The OMG developed a general Model Driven Architecture (MDA). MDA is a framework for defining meta-models, transforming them, defining the accompanying notations and exchanging the models and their diagrams in a standardized exchange format (XMI). For business modeling, the OMG created two specifications: BSBR, business semantic for business rules, and BPDM, business process document management. The latter is based on UML 2.0.

3.5.4.6. Business process monitoring and reporting

A major advantage of computer based management of business processes is the degree of monitoring available in the process. At the micro level, it provides the ability to track and monitor individual work requests and at the macro level, review resource productivity and work volume analysis. The ability to quickly search for and identify a work request within the process allows a business user to quickly respond to customer enquiries, and to possibly extend this functionality to customers for online status query. There has been yet no initiative to standardize the monitoring and reporting models of business processes, though this is a critical issue for any business environment measurement based control.

3.5.5. Requirement engineering approaches

3.5.5.1. Process models building blocks: Actors, roles & rules

The requirement engineering (RE) discipline is a mature discipline that though oriented toward system development, starts normally at gaining an understanding of the organization which is the environment in which the systems to be should be integrated. In [17], the authors propose to model the business organization as a network of collaborating processes. Each process has its own goals, and is associated with a set of soft-goals.

[55][32][51][56] agree that, Business process modeling is concerned with specifying the actors, their roles, their goals and the rules that determine the behavior of the process; There is a close relationship between all these and BP modeling techniques seek to define an integrated model. By Actor the author refers to the physical entity that plays one or more roles. A Role is defined as a set of responsibilities and related activities. Examples of Actor modeling approaches are: I*[64] , RADs (Role Activity Diagrams- [98]) and RIN (Role Interaction Nets [80]).

The F3 project [11] proposes the integration of a set of models for attaining a formal definition of requirements of the information systems. The requirements specification is represented as a structured description of five interrelated sub-models which provide the context within which the requirements are elicited. Each sub-model represents a particular concern or view in requirements acquisition, and these sub-models help separating the different concerns in a workable way. The sub-models are not developed in a linear, sequential manner. Although the process usually starts with an objectives model and progresses through actor and activity models to information systems requirements this is not always the case. For instance, with existing systems the activity and concept models may be developed first by reverse engineering previous designs. The objectives sub-model describes the why component of a requirements specification. It is a graph with components such as goals, problems, opportunities and weaknesses as nodes connected through relationships of the type 'motivates'. The objectives sub-model is related to rationale models such as IBIS [71] but it contains a goal decomposition hierarchy close to other proposals such as [30][75] and [33].
The concept sub-model is used to define the Universe of Discourse that concerns requirements engineers. It may serve as a dictionary of user and customer defined concepts. The actors sub-model is used to define the actors in the domain and their relationships with activities and objectives. Actors may be individuals, groups, roles, organizational units, systems, etc. Actors in the sub model are related to goals in the objectives sub-model and therefore represent stakeholders who are responsible for achieving goals through activities described in the activities sub-model. The activities sub-model describes the organizational activities, i.e. the processes and tasks of the organization. Components in this sub-model are created to achieve goals in the objectives sub-model, referring to components of the concepts sub-model, and resources required to carry out these activities described in the actors sub-model. The information

system requirements sub-model is meant to be derived from the other models. It includes both functional and non functional requirements. The former typically indicate needs for establishing objects, defining operations and services in the Object Oriented terminology or functions in top-down decomposition such as Structured Systems Analysis [93]. The latter are related to the environment, performance and quality of the required system.

3.5.5.2. Goal oriented approach requirement engineering

As said before, many authors agree that business process goal should be a major component of any business process model. While these concepts have been applied in general to intra-organization process analysis, their generality and further relevance and applicability to VO business process is obvious.
Within the requirement engineering researchers community, it is largely agreed that that the use of a goal driven approach leads to improved understanding of the problem realm for decision makers, requirements holders and customers as well as for developers. There is convergence on the view that goal modeling is an effective way to identify requirements [71], to elicit high-level goals to be achieved by the envisioned system [31][11], to help in the refinement of these goals [31][94][95][16] and their operationalisation into system requirements specifying how goals should be accomplished by the proposed system [2].

According to the nature of a goal, Yu [95] distinguished between *hard* goals and *soft* goals For a hard goal, the achievement criterion is sharply defined (e.g., "send an order"); for a soft goal, it is up to the goal originator, or an agreement between the involved partners, to decide when the goal is considered to have been achieved (e.g., "delivery a high quality product"). In comparison to hard goals, soft goals are highly subjective and strictly related to a particular context (what is meant by "high quality?"). [23] postulates the *soft goal modeling* aims at producing operational definitions of the soft goals sufficient to capture explicitly the semantics that are usually assigned implicitly by the user, and highlight the system quality issues from the outset. A soft goal is refined in terms of subordinate soft goals, hard goals, tasks, and constraints. Constraints are associated with hard goals and tasks to specify the corresponding quality attributes. So, for example, the soft goal "deliver a high quality product" will spawn the

hard goal "deliver a high quality product" and a set of associated constraints, that specify the product quality attributes according to the stakeholders' perception.

Other researchers declare that while goals (also called "hard-goals") are achieved by the set processes, the soft-goals can be only "satisficed". This gave birth to different models that try to map soft-goals to other soft-goals and hard-goals with the ultimate objective of designing the optimal organization and processes.

Mylopoulos et al. [64][94] proposed the Non-Functional Requirements (NFR) approach and its associated techniques (Tropos, i*, Telos, etc.) [46][44] , which is based on the notion of soft-goals rather in combination with (hard) goals. A soft-goal is "satisficed" rather than achieved.

Goal "satisficing" is based on the notion that soft goals are never totally achieved or not achieved. Therefore Mylopoulos et al . [64][94] state that, “soft-goals are satisficed when there is sufficient positive and little negative evidence for this claim, and that they are unsatisficeable when there is sufficient negative evidence and little positive support for their satisficeability.”

The NFR approach evolved into the Goal-oriented Requirement Language (GRL). GRL is part of the ITU-T URN standard draft which also incorporates Use Case Maps (UCM).

Rolland et al.[16], in the ESPRIT CREWS project focuses more on goal definition and the linking of goals to stakeholders’ actual needs by linking goals and scenarios. The KAOS ((Knowledge Acquisition in autOmated Specification) – Dardenne's [31] approach consists of a formal framework based on temporal logic and AI refinement techniques where all terms such as goal and state are consistently and rigorously defined. The main emphasis of KAOS is on the formal proof that the requirements match the goals that were defined for the envisioned system.

The GBRAM (Goal-Based Requirements Analysis Method- [2]) defines a top-down analysis method refining goals and attributing them to agents starting from inputs such as corporate mission statements, policy statements, interview transcripts etc.

The concepts of hard goals and soft goals are generic concept that are independent from the process scope (be it intra-organization or inter-organization process), and is directly adoptable for VO processes modeling.

4. The Virtual Organization Model

4.1. General

The proposed model builds upon BWW's ontology and the GPM model and extends it to incorporate the relevant concepts for modeling business processes in virtual organizations. The extension focuses on the concept of contractual obligation (that we shall name, from here on, "VO obligation" or simply "obligation"), the elements that are included in it, and its incorporation into the Business process lifecycle management.

4.2. VO model assumptions

We start by presenting the VO business process related assumptions we adopt in our model design. These assumptions are the minimal set of ontological assumptions we need in order to scope of the real world problem we are considering. As explained earlier minimizing these assumptions would lead us to minimal ontological commitments and would help us conserve the model generality and applicability to a wide range of real life scenarios. These assumptions have been established using a two step procedure: (1) Analysis of the research work done within the field of VO BPM, as presented in the chapter 3; (2) Synthesis of the VO business process characteristics and identification of the needs we considered critical for modern VO's lifecycle management, that weren't enough complied with till now, as far as we know. As a result of this procedure, we observed that the issues of conservation of the autonomy and privacy of each VO partner are of major importance. We consider that one of our major contribution consist in that, we gave a major focus on preserving the partners' autonomy for the whole BP lifecycle (from the specification, design, implementation, monitoring and further change), in contrast to other work done within this field, where as we discusses in the related work review (Chapter 3), the focus was not set enough on it (e.g. we consider that PRODNET II project resulting model application, may preserve the partners autonomy during the implementation while constraining it drastically during the design phase due to the high level of iteration required between partners). Another assumption we consider of critical importance is the no central governance existence in

the VO. Without this assumption, the VO would not be possible without being monitored and operated in a centralized way. In addition this assumption allows us to design a much more generic model as it may allow us to model dynamic and static VO's using the same set of ontological concepts.

4.2.1. The Virtual organization concept

We postulate that a virtual organization is a set of organizations that share some business processes according to defined business obligations between them.
Established obligations constitute the mechanism that allows the partners to function as one single organization for specific purposes, by fomenting the needed trust to share processes. Our model should be a conceptual model and as such should be generic enough to be implementation independent. For example, established obligations may be implemented through different collaboration and coordination schemes, such as formal business contracts, workflows and different levels of automations (human coordinated, partially automated or fully automated). The specified model should be implementation agnostic in order not to limit its generality. This is an important issue, for example, in order to conserve our model capability to model static and dynamic VO's.
The model must be also generic enough to serve for such objectives as for example a VO implementation design for a VO to be, the early requirements engineering phase in a VO process RE effort, change management effort in an existing VO, a basis for inter-organizational workflow design for VO processes, etc.

4.2.2. No VO Central governance

We assume there is ***no central governance*** and control of the VO. There may be processes shared by a subset of the organizations, others may be shared by all partners. Thus we assume that no third party entity is in charge of the design of the end-to-end VO process. In fact no end-to-end process design occurs. This does not contradict the existence, in some VO's, of central committees which we assume do not impose nor design business processes, nor control the execution of processes within the VO, but rather define major goals and guidelines of the VO collaboration frames.

4.2.2.1. Internal Business process autonomy

We assume that each Partner organization has its own autonomous business processes. These business processes are neither visible nor accessible by other partners.

4.2.2.2. VO Partner autonomy to design, implement, change and monitor

Each partner is not constrained in its own process design and implementation of its internal processes; nor is he constrained to any specific implementation in its design and implementation of its interaction with any other VO partner.

4.2.2.3. Interacting partners selection autonomy

Each partner is aware of existing partners. Each partner has a complete autonomy to choose with whom between the existing partners he interacts based upon his own private, independent and autonomous criteria. These selection criteria are based upon partner's own goals, soft-goals and business constraints, and may be held unknown to others, partially shared with others or completely shared with other partners. It may be also shared with selected part of the VO partners.
Also, each partner may change its interacting partners in each process instance, whenever he decides, without any prior notification to any partner.

4.2.2.4. Partial delegation of processes

All partners may share a process so that each partner executes a part of the process and delegates other parts to other selected partners. Each partner receives as an input a set of events from another VO partner, and produces an output that is forwarded within the VO, until the VO process is completed (that is, reaches its goal).

4.2.2.5. Partner business autonomy

No partner is constrained to report to any other about activities and processes not related to the shared VO business processes. Thus, for example, a partner organization may be collaborating within several different VO's, in each one for different business goal accomplishments.

4.2.3. The VO business process views

The VO end-to-end process is a process that includes all partner processes and the inter-partner processes. Although the entire process of the VO exists, no entity has a complete view of all its states and states variables. Partners have only a view of the accorded inputs and outputs of each other as well as their own part of the process.

4.2.4. Constraints on performance parameters

VO obligations can also impose given performance parameters as constraints on process execution, in what is sometimes referred to as Service Level Agreement (SLA) or Quality of Service (QoS) parameters.

VO obligations specify thresholds to such performance parameters, which, in turn, must be met by the process executed by the different partners. Obligations may also address undesirable states, which are cases where the process is forced to deviate from its normal course, and may not be able to attain its original goal.

4.2.5. Non Virtual components of the VO

Virtual organizations may also include parts that are not "virtual", e.g., shared resources, regulating committees, centralized processes, etc. In this work, we address the "minimal" virtual organization, which only shares business processes on the basis of obligations.

4.3. Formalizing the virtual organization business process

4.3.1. Background

Processes in virtual organizations are a specific form of inter-organizational processes. GPM does not explicitly address inter-organizational processes. The notion of domain sets the scope of control over the process in a GPM process model.

Consider an inter-organizational process modeled from the point of view of one of the partners (that is, defined over the domain of the partner organization). The process reaches a state where some request was made for another organization (e.g., supplier) to operate. Since actions taken by other organizations are outside the scope of the domain,

the process domain remains in a stable state, waiting for the results of the other partner's action. These results are conceived as an external event, and are not within the control of the requesting organization.

At the occurrence of the expected external event, the state of the domain transforms from stable to unstable and the process is reactivated. This kind of situation is termed a *discontinuity point* [81], since the process (as defined within the organization) depends on an external event in order to proceed and achieve its goal. According to [81], discontinuity points need to be monitored, and to take into account a possibility of failure in the external event, which may result in a need for undesirable state handling. The monitoring and undesirable state handling are required as means taken by the organization to assure the process validity, namely its ability to progress towards its goal, despite the dependency on events that are not within the organization's scope of control.

Now assume the two organizations operate as partners in a virtual organization (VO). The inter-organizational process occurring between partners may be mapped *to the external events that are expected to result from the actions of the other partners*. Contractual obligations that are defined among the partners allow each partner to be in control of its processes and to react consistently to other partners events sent to him.

We shall formalize these notions below.

4.3.2. The Model

4.3.2.1. Private and shared domains

Definition 1: The *VO domain* is a domain including all the partner organizations in the virtual organization.

The partner organizations are all sub-domains of the VO domain, and shall hence on be denoted as the partner sub-domains. The VO domain is not centrally mandated, but it can be captured as an aggregation of all its partner sub-domains, which are autonomous and *private*.

Definition 2: A sub-domain is said to be *private* if its states, goals, and law cannot be controlled or viewed from outside the sub-domain.

The partner sub-domains in the VO interact with each other. In ontological terms, this interaction takes place through properties that are mutual to at least two partners. When the value of such property (reflected by a state variable value) is changed by one

partner, this event affects the state of the other partner too and triggers a transformation in it.

Definition 3: The sub-domain that includes all the state variables that represent mutual properties of the partners shall be termed the *shared sub-domain*.

A process in the virtual organization, namely, a VO process, is a process that takes place in the VO domain. Since the VO is composed of private sub-domains, a VO process cannot be viewed or defined as a whole, but it is known to exist, as it is the purpose of the VO formation. Nevertheless, the internal processes of the partners, which are well defined and known within each partner sub-domain, are *projections* of the VO process over the partner sub-domain.

Note that VO obligations binding the partners in a VO concern process parts that are not private, namely the shared sub-domain. We would like to be able to say that VO obligations define projections of the VO process over the shared sub-domain. However, since the shared sub-domain includes only mutual properties, its state cannot transform by itself. Rather, every transformation in the shared sub-domain is a result of an event which is external to that sub-domain, be it in a partner sub-domain or external to the VO (e.g., time). This argument leads to the following lemma.

Lemma: All the states of the shared sub-domain are stable.

A state can only be regarded stable or unstable with respect to a (sub)domain. Since all the transformations in the shared sub-domain are a result of external events, all its states are stable. Hence, no meaningful projection of the law exists over the shared sub-domain. Rather, all its states can be considered as discontinuity points, where the (sub)domain is "waiting" for an external event for its state to be transformed. For the VO process to progress towards its goal, these events should be generated by the partner organizations.

Note that the VO process may have various paths, reflected in a variety of possible states in the shared sub-domain. These possible states should be defined as part of the obligation set between partner organizations, where each partner is obligated to the states that depend on events it should generate.

4.3.2.2. Partner obligations

Definition 4: An *obligation* of a partner to a state means that the law defined in its private sub-domain assures the achievement of that state.

The outcome of an obligation is that although an observer is not familiar with the projection of the VO process on a private sub-domain, he knows it is designed so that the state under consideration will be achieved.

Consider a partner organization O_1, whose private part of the VO process includes a discontinuity point, where the stable state eventually becomes unstable as a result of an external event *e*, and assume the generation of this event is an obligation of organization O_2. Since both O_1 and O_2 are part of the VO, O_1 knows that the law is designed so that *e* will occur. Then the O_1 process will be able to progress towards its goal, which is either the goal of the VO process or a state in the shared sub-domain to which it is committed by the VO obligation.

In other words, the implications of VO obligations on the process design in each partner's organization are the following:

(a) A given set of states that the process must reach. These may be the goal of the (local) process or some mandatory states within the process that constrain its course.

(b) A higher level of certainty (or trust) that specific external events, expected in discontinuity points in the process, will occur. This trust allows the process designer to take less caution measures (monitoring, reminders, etc.) than would be necessary otherwise.

Note that the VO process can interact with entities outside the VO domain, and may have discontinuity points dependent on events outside the VO domain. Hence, undesirable states, where the process fails to reach its original goal, may still occur. To address such situations, states related to undesirable state handling paths should be included in the associated VO obligation definition.

4.3.2.3. VO Obligations and soft-goals relations

Another element in the contractual obligation between the partner organizations is related to the soft-goals (or business performance measures, also denoted as QoS parameters) of the VO and of each of the partners. The obligation may include constraints on the performance measures related to the generation of the events in the shared sub-domain. For example, response to a request (event) should be made within a given time limit (performance measure). These constraints may take one of two forms:

(a) The performance measure is part of the state definition in the shared sub-domain, to which the partner is obligated. Then performance is a part of the (hard)goal, and

failing to meet the constraint leads to an undesirable state. The VO process should then take a path that is different than the "normal", and may fail to reach its goal.

(b) The performance measure is not part of the obligated state definition, but there may be "punishment" defined for not meeting the required value. For example, if a partner's delivery falls behind schedule he should pay a predefined sum of money to the other partner. The delivery schedule in this case acts as a soft-goal, because the (hard)goal is that delivery is made, as is specified in the shared sub-domain state definition to which the interacting parties commit through an established obligation.

Note that the commitment to the performance parameter values may serve three different purposes. *First*, these values may be derived from a commitment made to the end customer of the VO. For example, a commitment to deliver the customer order within a week requires partner organizations participating in the delivery process to perform accordingly. Depending on the terms of the commitment made to the customer, delivery time may be defined as a hard or a soft-goal for the partners. *Second*, performance measures may be constrained in order to prevent failure in achieving the VO process goal. For example, if a partner has not responded to a request within a given time another partner will be requested instead. In this case, time is clearly defined as part of the hard goal. *Third*, constraints on performance measures can also serve for reducing the uncertainty related to the event. Considering, again, delivery time, the other partners know with a high level of certainty not only that delivery will be made, but also when it will be made. Commitments serving this purpose are usually in the form of soft-goals, since no undesirable states are involved.

4.4. Model summary

The VO process occurs within a VO domain that is partitioned into private partner domains and the shared domain.

Each partner has its own private domain, which is invisible to other partners. Partner private processes' design, implementation and change management are unknown to any other partner.

The interaction of partners is modeled through the shared domain states. Within the shared domain, all states are stable, that is the shared domain state transformation is triggered only through events coming from private partner domains, or domains that are external to the VO. Within the shared domain, there is no transformation law that

occurs, as there is no entity in the shared domain that could potentially trigger any transformation. Hence, all sources of events and associated transformations are external to the shared sub-domain. This means that there is no internal process in the shared sub-domain, as no continuous internal law can be defined within it. This conclusion stands in line with the initial assumption that no central governance and/or control exist within the VO. The existence of a law in the shared sub-domain would have implied that there is an entity external to all partners that governs the sequencing of states between partners and would contradict our initial assumption. Soft-goal constraints imposed on the partner's processes are well demonstrated too.

Each interaction between partners is governed through a pre-specified obligation which specifies the relevant shared domain states and the relevant soft-goals constraints (performance constraints) associated with each shared domain state, and undesirable state handling related states.

Thus, the construction of a VO obligation in the course of a VO formation should address the states in the shared sub-domain to which the partners will be obligated, and the constraints on the soft-goals that are implied in these states. It may be possible that a partner organization already has existing (local) business processes that should be matched to the pre-established VO obligations. Thanks to the obligation specification, this matching becomes rather simple, verifying that the required states are reached by the existing process. The constraints implied on soft-goals should be assessed and evaluated for feasibility and cost-effectiveness.

Finally, we consider that one of the main contributions of a VO formation to inter-organizational processes is the *reduction of the uncertainty related to the external events that are expected to result from the actions of the other partners, with minimal abstraction of partners' privacy and autonomy*. Such reduction in the uncertainty related to external events is achieved through the formation of a virtual organization and the contractual obligations that exist among the parties. The reduction of uncertainty is a major trust fomenter between partners. This is much in line with the trust based collaboration mechanisms that we mentioned in the related work chapter.

In summary, as we will demonstrate through the thorough case study in the next chapter, we consider our work is novel in providing a formally defined set of concepts that form a conceptual model of the VO domain, formalizing the notions of private and shared domains to be included in obligation definitions.

5. VO's BPM approach Proposal

5.1. Introduction

In the following chapter, we propose a BPM approach based upon the VO conceptual model (VOM) we presented along the previous chapters of this thesis. A summary of the model concepts is provided in Annex 1.

As discussed in the related work chapter of this thesis, the need of better coordination mechanisms for VO is an issue that though has been researched, is an open issue. In the previous case study analysis, several characteristics of such mechanisms were identified. The proposed mechanisms, though were illustrated on our specific case study, are generic enough to be applied to many other real world scenarios: specifically, it is obvious that order processing, partner selection, delivery processes and payment processes are generic processes that exist in most of the VO form, though what may change is the process soft-goals.

We consider that coordination mechanisms should be as simple and minimal as possible, allow the parties to be synchronized to the current state of the shared processes, assure the awareness of each party to the current process state, conserve a high level of process resiliency (process freezing, duplication of tasks due to duplicate messages, etc.), simplify the undesirable states handling mechanisms (should not require to build specific undesirable state handling processes such as is done in typical workflow models); further they should not limit the process automation and implementation possibilities. At the business level they should allow different level of service in each interaction and different negotiable options for each interaction.

Our proposition is to engineer, design and further implements better collaboration mechanisms through the use of sets of VO obligations, where in each collaboration scenario, several obligations may be triggered in a sequential way. Each VO obligation should specify a triplet composed by (1) a set of shared states that are a subset of the whole set of shared states; (2) a set of options that are represented by state variables; (3) Performance constraints that are represented by conditions over criterion functions, which parameters are a subset of the state variables.

Apart of the partner private domains and the shared domain states, we add the definition of the VO obligations between partners, which are defined over the shared domain. Not all partners are obligated to support/implement all obligations but the implementation of a specific obligation is a necessary condition for providing services that make use of it. In addition in each VO obligation, each partner may choose which options he supports and which not; for example a partner may support a request delivery but may choose not to support the delivery to home option, or a specific item type format, or an email delivery option. Another partner may support delivery of electronic items while not supporting delivery of hard copied items.

Each of the supported VO obligations parameters (that represent different service options and performance parameters values) is included in the relevant states as part of their state variables. In addition, performance constraints are added as conditions over criterion functions that are parameterized by the value of state variables such as timeout values, item quality level, delivery time, etc.

5.2. The proposed BPM approach

Using the concepts presented in chapter 4 and summarized in Annex 1, let us consider how can we establish, change and validate a VO business process model.

We shall consider two major scenarios:

(a) VO Creation BPM scenario. How do we create a valid model for a new VO BPM ?

(b) VO BP change scenario. For an existing VO, how do we change the model in order to adapt it to new change requirements, while conserving the model validity ?

5.2.1. VO creation BPM scenario

We propose the following BPM modeling steps summarized in Table 1 and detailed afterwards.

Step #	*Step objective*	*Step tasks*
1	***Partner sub-domain BP design***	(1) Identify the 4 components of its processes: {S, L, I, G} (2) Identify stable and unstable states (3) Define state variables semantics and allowed values. (4) Identify external incoming events and outgoing events.
2	***Sub-domain interaction design***	(1) Shared state variables identification (2) External events mapping. (3) Shared sub-domain shared states identification.
3	***Performance constraints mapping***	(1) Map expected performance constraints for each mapped shared state (e.g. timing, quality, etc.) (2) For each constraint, private constraints are mapped within each partner's domain over its interaction state variables.
4	***VO process validity checking***	(1) Partner domain process validity check. (2) Shared domain validation.
5	***VO undesirable states handling***	(1) Identification of shared states that contribute negatively to performance constraints compliance (2) Minimize the occurrence of these states (3) Rechecking the validity of the Partner and shared sub-domains.

Table 1. BPM approach steps summary

Step 1. Partner sub-domain BP design. Using the GPM set of concepts, each partner maps its own private domain, identifying the 4 components of its processes: {S, L, I, G}. For each identified state variable (within the S and G sets), the partner should specify the semantic and set of allowed values. Unstable and stable states must be identified. Incoming External events (coming from other domains) and outgoing events (to other domains) should be identified. As an illustration of this step, refer to Figure 2 and Figure 3, which represent the graphically the process models of the 2 interacting partners.

Step 2. Sub-domain interaction design. This step includes several tasks:
(1) Shared state variables identification. Each sub-domain must externalize required interaction state variables; as we will see later these are the building blocks that would

serve for the definition of shared state variables once the shared states are identified. We illustrated this step through the current and changed state tables of the VO process (Table 3 and Table 4). In these tables the definition of the states, given in the second column, includes variables that represent domains' mutual properties.

(2) External events mapping. Each domain should identify events that would be required from other sub-domains and events that would be sent to other domains.

(3) Shared sub-domain shared states identification. Observing the patterns of events exchanged between partners' domains, the partners define the shared states by grouping identified shared states variables in accordance with the partners' interaction states.

We illustrate the identified shared states and their associated exchanged events patterns in Figure 4. A detailed definition of these states is given in Table 1 for the current process and in Table 4 for the future process model.

Step 3. Performance constraints mapping. For each mapped shared state expected performance constraints (e.g. timing, quality, etc.) are mapped. For each constraint, specific constraints are mapped within each partner's domain over its private state variables that reflect projected shared state variables. In the illustrated case study we mapped the required performance constraints within the state tables (Table 3 and Table 4). We see that many of the shared states have been associated some constraints in the form of a transformation function over a specific condition. Such is the payment maximum time associated to the "payment made" shared state or the maximum response time required in the "request received by supplier" shared state.

Step 4. VO process validity checking. This step includes two major tasks:

(1) *Partner domain process validity check.* Following the GPM, A process model {S, L, I, G} is said to be a "valid model" if every process path leads to a goal

state. It is the responsibility of each partner to check that its process model design is valid. The GPM identifies three possible cases of invalidity of processes:

Case (1): Incomplete law definition: States for which the law is not defined. In particular, with respect to state variables whose value is determined by an external event and realized in the course of the process. In such a case the defined law must be modify in order to include all missing states.

Case (2): Law/goal inconsistency may occur if there are infinite loops, deadlocks or Exceptional termination. Exceptional termination refers to the existence of stable states for which no conceivable external event is expected to make it unstable. In such case, it is necessary to modify law definition, or include exceptional termination in the process definition.

Case (3): Discontinuity of the process: the application of the law leads to a stable state not in the goal, and the process is "waiting" for an external event. In such case, we must add time-dependent event that makes the state unstable and connect it to a process path.

As an illustration of the validity check of the private processes, examining each one of the states in the requester process model and the supplier process within the case study, we see that both the models have no law inconsistency, law/goal inconsistency or process discontinuities and hence may be said valid.

(2) *Shared domain validation.* Note that the shared domain includes no Laws or transformation and all its states are stable and are defined by state variables that are projected into the different partner domains (as interaction state variables) to allow the representation of domain properties. Note also that VO process goals

may be visible (defined in the shared domain) and invisible (internally defined within one or more partner domains. Hence, validation the shared domain is equivalent to assure that, for each Partner interaction state variable:

a. The partner interaction state variable must be a valid projection (following the" state projection" VOM definition) of a shared state variable defined within the shared domain, that is:
    i. The set of allowed interaction state variable values include all required values to allow the correct interpretation of the events associated with the shared state.
    ii. The Partner interaction state (which are stable by definition) must have an associated law that allow it to transform into an unstable state through a condition applied to the interaction variable.
b. Performance constraints associated with the shared state are mapped into adequate internal constraints over interaction states. These private constraints include necessary conditions and condition values as specified within the VO obligation.

Step 5. VO undesirable states handling. Includes three major tasks:

(1) *Identification of shared states that contribute negatively to performance constraints compliance.* These states are states that may cause for example delays or reduced quality in goal completion and thus affect the soft-goals "satisficing" level.

(2) *Minimize the occurrence of these states* by changing if needed the definition of performance constraints, shared variables semantics and/or set of values. Any change introduced at this level should be associated with changes in the private domain models ***(internal constraints, state variables semantics and values).***

(3) *Rechecking the validity of the VO process as specified in step 4.*

5.2.2. VO BP change scenario

Consider now the case of an existing VO BP assessment. Our objective is to improve or change and existing BP. We propose the following procedure, where steps (1) to (3) are the same procedures though their objective is to document the current process specification. Step (4) consist in assessing and identifying the changes requirement engineering; Step (5) and (6) – mapping these change requirements in the partner and shared domains model, much as was done in step (1)-(3); Steps (7) – validating the models as explained in the pervious section; Step (8) is equivalent to step (5) in the previous section.

We summarize the BPM approach steps in the following table.

Step #	*Step Objective*	*Step tasks*
1	***Current Partner sub-domain BP model mapping***	(1) Identify the 4 components of its processes: {S, L, I, G} (2) Identify stable and unstable states (3) Define state variables semantics and allowed values. (4) Identify external incoming events and outgoing events.
2	***Current Sub-domain interaction model mapping***	(1) Shared state variables identification (2) External events mapping. (3) Shared sub-domain shared states identification.
3	***Current Performance constraints mapping***	(1) Map expected performance constraints for each mapped shared state (e.g. timing, quality, etc.) (2) For each constraint, private constraints are mapped within each partner's domain over its interaction state variables.
4	***Process Change Requirements assessment***	(1) Identify the change requirements for the VO process.
5	***Sub-domain BP model change mapping***	(1) Map change requirements into the partners' sub-domains models.
6	***Shared domain model change mapping***	(1) Map change requirements into the shared sub-domains model.
7	***Model validity check***	(1) Partner domain process validity check. (2) Shared domain validation.
8	***VO undesirable states handling***	(1) Identification of shared states that contribute negatively to performance constraints compliance (2) Minimize the occurrence of these states (3) Recheck the validity of the partner and shared sub-domains models.

Table 2. BPM approach to VO process change modeling

6. Illustration of the VO VBPM approach to the ILL case study

6.1. General

Hereafter we present the inter-library loans process (ILL) as an illustrative case study. We base ourselves on a real case study of a university library processes for managing the inter-library loans.

We shall apply the proposed BPM model to this case study. We commence by an analysis of current process state of implementation, analyzing what are the current methods of design and analysis of the overall ILL process.

6.2. Background: Generic ILL business process description

Libraries partner with each other in order to share items, collections, journals and thus provide their customers maximum accessibility to interesting items. This process must be as transparent as possible to all customers (except for inevitable costs and delivery time issues).

Libraries nation-wide and internationally have formed since the 1980s consortiums and associations, which permit the item sharing between partners of the association. Examples of such organizations are the OCLC consortium, and Nation-wide associations such as the American Library Association (ALA) in USA. The advances in Information technology and mostly in data-base technologies allowed the implementation of virtual catalogues that are shared unified catalogues between partners' libraries.

A high-level description of the main modern library processes is presented schematically in Figure 1.

The ILL process may be triggered by students, researchers or research centers, universities, colleges etc. The requester entity asks for an item from the virtual catalogue that includes all available items locally and within the association. If the item is available locally, the lending process is a simple local one and involves no transaction except the update of the status of the item in the catalogue. If not, the business process support system of the library (Called ILS- integrated library system) searches for

tentative suppliers through the catalogue and rank them according to a set of parameters, such as delivery time, quality, price, etc. Afterward, the ILS sends a request to the first ranked tentative supplier and waits for response. Different scenarios may occur: a normal scenario, in which the supplier accepts the request, notifies the requester, and then payment is made by the requester followed by an activation of the delivery process. An alternative scenario is when the tentative supplier response does not arrive within a given period of time. The request is timed-out and the requester may initiate a request to another tentative supplier.

All the process logic is normally established at the level of the consortium/ association and each partner that joins the consortium agrees to comply with it. Partners' systems communicate through a standardized ILL protocol for requests, responses, cancellation, status queries, and other transactions.

All major processes may be triggered (manually) through the library desk by librarians. Different ordering, billing & delivery options depend on:

- Item type: journal, book, Multi-media item (Video), collection, rare collection, etc.
- Item format: electronic, non-electronic.
- Item format (for electronic items only).
- Item location: local library national library, international library/provider.

The Virtual catalogue may be composed of one/several union catalogues in addition to several catalogues of purchased collections, e-journals, etc. The Union Catalogues are generally managed centrally by a shared entity (For example: MELAMED is such an entity in Israel). Several Union catalogues may exist (In Israel, at least 3 exist currently). The central entity is in charge of collecting updates, creating an updated version of the catalogue and broadcasting the catalogue to all partners.

The delivery process can change a lot depending of item type and format. As an example, for item-type= e-journal, delivery is by email or hard-copy. Hard-copy has a different fee and can be delivered to home or at the library desk. Delivery to home has an additional fee, and different delivery times. (Some additional conditions about availability of the requester at the delivery address at specific hours are required; but, for item-type= book, delivery is only at library desk or at home. A warranty fee paid at user registration to the library is retained till the item is re-checked in. Special collections are dealt with more precautions; additional warranty fees are generally

required. For international loans, additional shipment fee is required. Delivery times are generally longer, expect for e-journals.

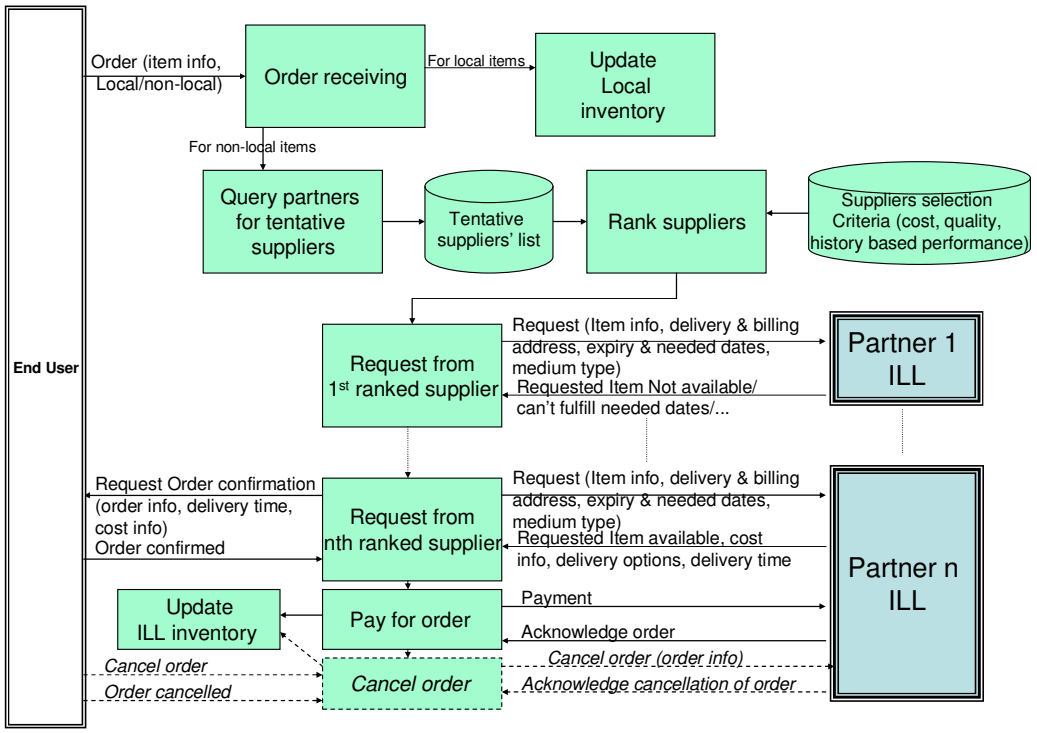


Figure 1. High level description of the ordering related processes for ILL.

6.3. Case study specific ILL business process analysis

A detailed analysis of the ILL business process is provided in Annex 1.

6.4. Modeling of the current ILL process using the proposed BPM

6.4.1. Assessing the current processes using the proposed VO model

Due to the complexity of the case study, we will focus on a set of processes within the ILL- the order and delivery processes.

We propose to model the association of libraries as a VO which is composed of partner libraries, each having its own private domain and its own implementation of the ILS business processes. Aside of the "opaque" private domains, there is a shared domain in which we currently can identify some shared states that the partners are aware of, that is, though the collaboration is human based and informally defined, there are still a set of informally defined shared states within the shared domain

Within the VO, each partner may play one of two roles in the process: a requester or a supplier, and this role may change in different process instances. The overall process internals are known to no entity within the VO.

In the following we present the current processes modeled using our VO model.

6.4.2. Step 1- Current Partner Sub-domains BP model mapping.

Figure 2 schematizes the states of a requester library business process as depicted in the studied library. Figure 3 presents the states of a supplier business process as depicted in the studied library. Note that the supplier business process is not visible to the requester and the requester business process is not visible to the supplier. Furthermore, note that the only "things" the supplier is aware of from the requester business process are of the messages issued to him by the requester and the requester is aware only of the messages issued to him by the supplier.

The process is triggered by the arrival of a request from a customer. The request includes a set of given parameters, which are represented by a set of state variables associated with the private state “Request received from end-User”. Some of these state variables are “Item ID”, “Item ISBN”, “Item delivery address”, “Requested Item format”, “delivery options”, and “payment media”. The state “request received from end-user” is an unstable state, and it is transformed by the internal ILS system, which creates a tentative suppliers’ list (the transformation changes state variables that include ranking of tentative supplier, each with its associated ID, address, contact form, quality score, pricing score). Once the suppliers are ranked, the ILS triggers another transformation (an internal event), sending a request to the first ranked tentative supplier. In this case, although the event and its associated transformation are internal, it crosses

the boundaries of the domain, causing the initiation of a transformation in the supplier domain to the “request evaluation” private state.

From hereon, as depicted in Figure 3 the supplier evaluates the request (internal "Request in evaluation" state), and either accepts it (leading to the "request accepted" state through an internal event "supplier accepts request") or rejects it by sending an event "supplier rejects request" to the requester, who in his turn cancels the request (through the generation of the adequate event which causes the triggering of a transformation of the supplier state into the "Idle" state), while the requester domain state transforms into the "Select next tentative supplier" state where the requester prepares to initiate a request toward the next ranked potential supplier.

Once the supplier accepts the request, an internal event "supplier prepares item for delivery" causes the supplier state to transform into the "item prepared for delivery". IN this state the supplier waits for the requester to send him the payment within a privately defined period of time "maximum payment waiting time". This timer is unknown to the requester; that is the requester does not know he is expected to send the item within this time interval. If the requester does not comply this constraint, the supplier private timer times-out causing the generation of an event (that is external to the supplier domain; by definition – Time is external to all existing domains), which causes the triggering of an undesirable state "payment waiting timeout", in which the supplier sends an enquiry of payment to the requester which transforms the supplier state into "Payment enquiry sent" state. In this state, newly an internal supplier timer is initiated and if the requester does not supply the payment within the defined period of time the supplier rejects the request (transforming its internal state into "state rejected") and causing the requester state to transform into "request rejected".

Once the requester sends the payment – that is an external event which arrives to the supplier domain and triggers the supplier state transformation into the "payment received state" in which the supplier is meant to deliver the item to the customer following the specified delivery options (home delivery or delivery to the requester). Once the requester confirms the receiving of the item the process attains its goal and the supplier state transforms to "idle". From his part the requester has an internally defined timer for the maximum delivery waiting time. This timing parameter is not known to the supplier and thus the supplier is not committed to it. If the supplier does not supply the item within this time period, the requester state transforms into the "delivery waiting timeout", within which the requester triggers a delivery enquiry event toward the

supplier domain, and thus transforming the requester state to the "delivery enquiry sent" state. In this state the requester initiates newly a timer; in case this timer times-out again the requester sends a request cancellation toward the supplier and reinitiates the whole ordering process toward the next ranked supplier.

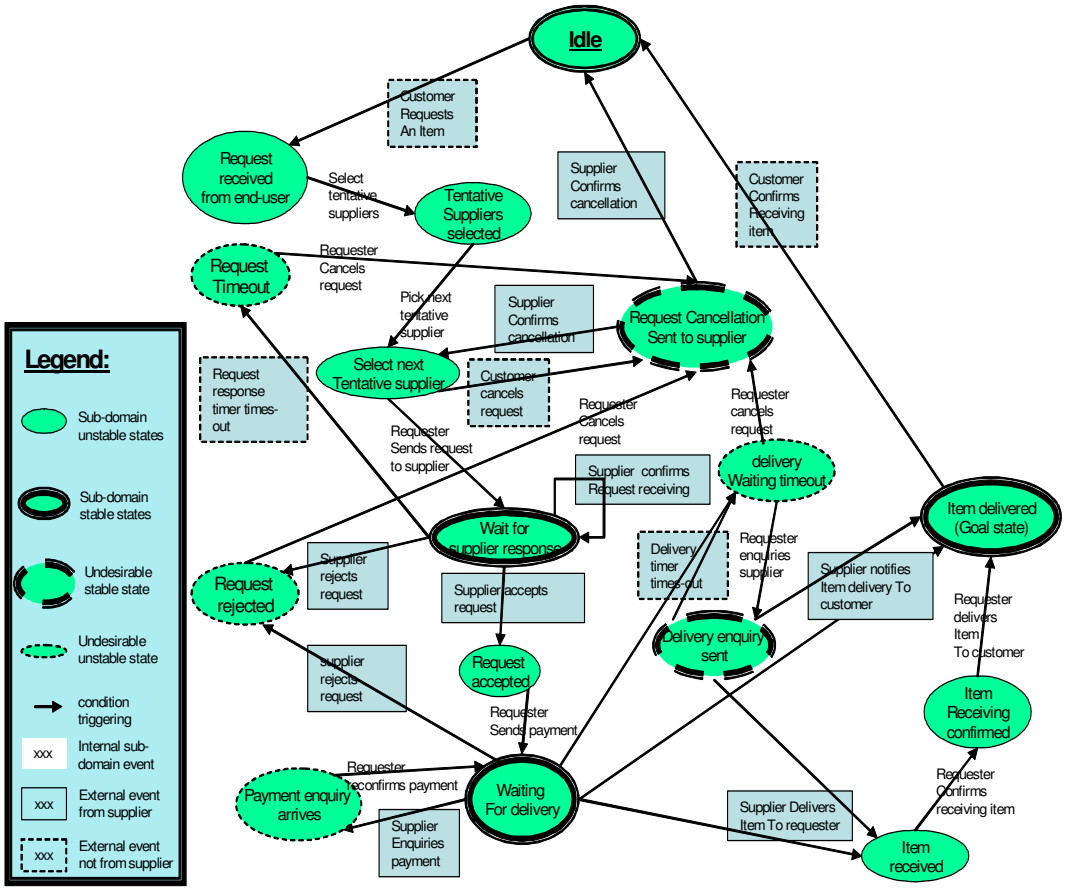


Figure 2. Current Requester sub-domain BP mapping.

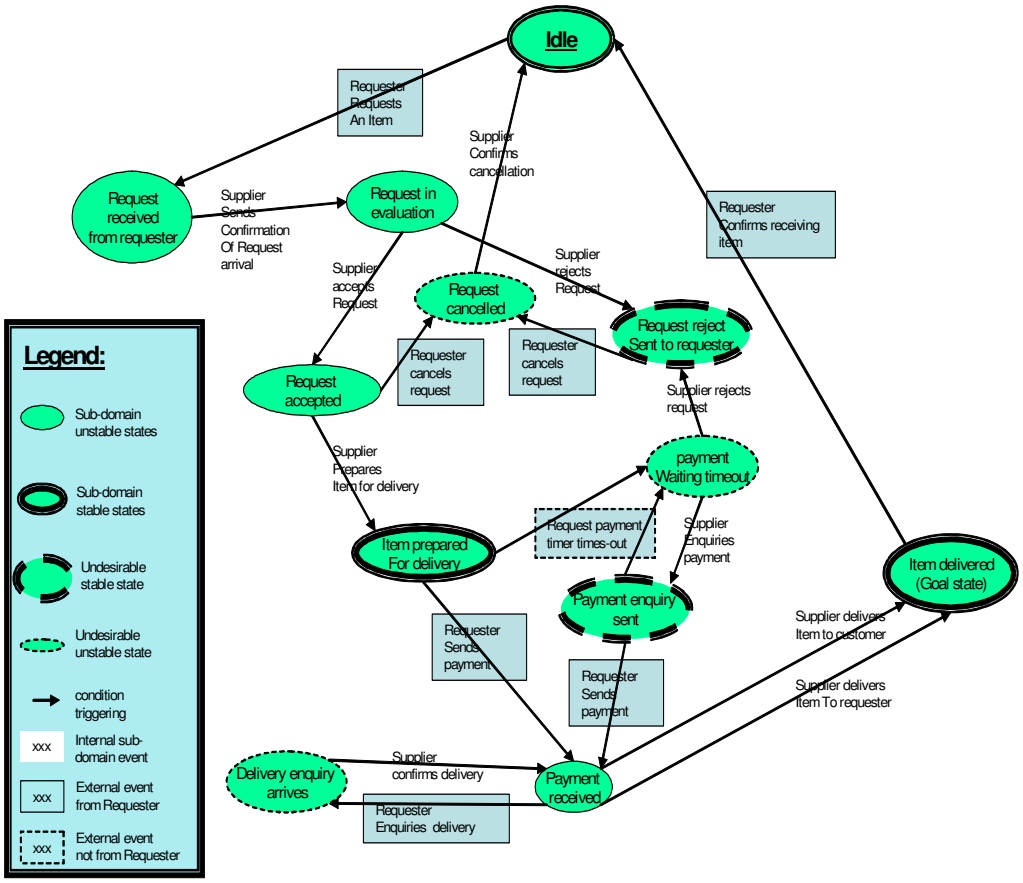


Figure 3. Current Supplier sub-domain BP mapping.

6.4.3. Step 2 - Sub-domain interaction design

Grouping the events exchanged by both sides, and arranging them as depicted in Figure 4, we see that, through these event flows, an external observer (that is an observer that is not part of the requester nor of the supplier domains), can confirm that a request has been sent to a specific supplier from a specific requester when he sees a sequence of 2 events: (a) "Requester send request" flow going from the requester to the supplier followed by (b) "Supplier confirms request" in the opposite direction. That is, without knowing what is occurring in the internals of the domains, the external observer can assure that the supplier has in his hands a request from the requester. This is a state that

is visible to both sides and is what we called a shared state (We called this state by the name of "request received by supplier").

Following the same logic, we mapped a set of shared states that we present in the following Figure 4.

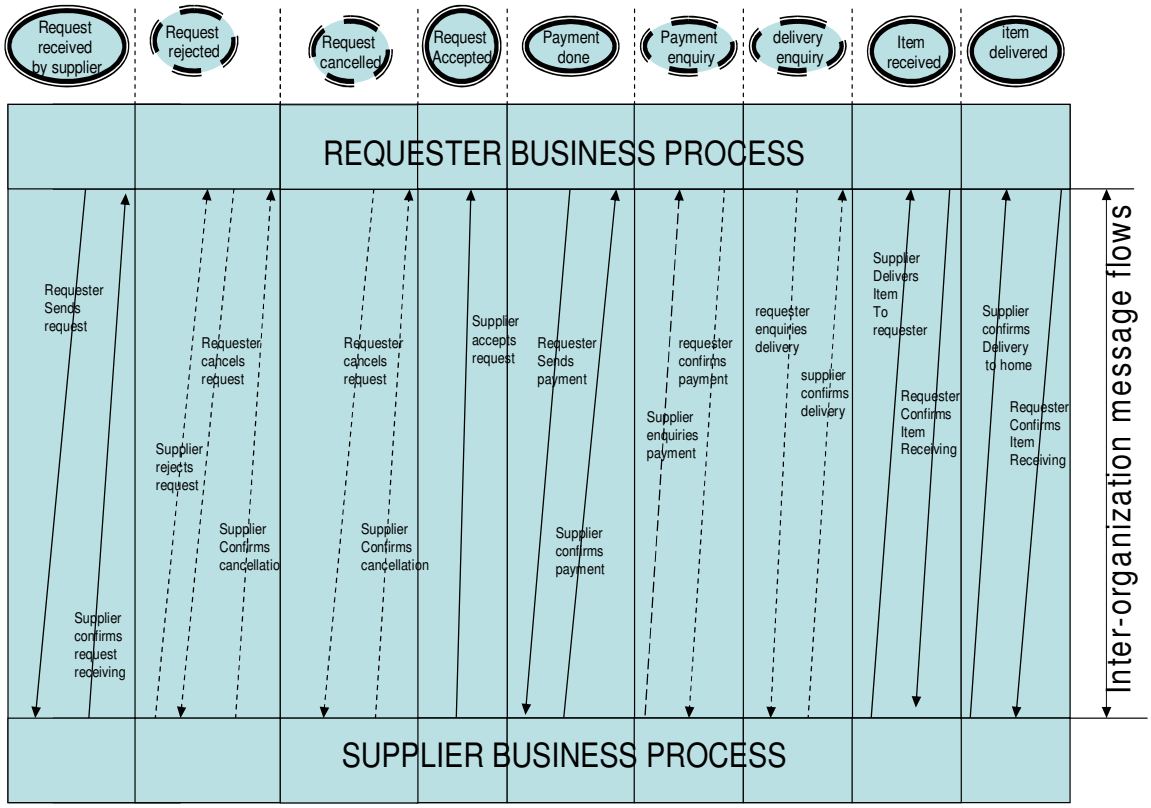


Figure 4. Current Shared Sub-domain model.

6.4.4. Step 3. Current Performance constraints mapping

In the current BP model no performance constraints were established.

6.4.5. Shared States definition summary

In summary we have a set of shared states, which occurrence is specified through the exchange of a sequence of events that are exchanged by the involved private domains. These states exist within a domain that is shared by all other domains. We call this the "shared domain". We summarize the definition of the **current** shared states in the following table:

State	Shared State variables definition	Achieving party	Affected party	Expected action of affected party	Performance constraints
Request received by supplier	Request status = sent to tentative supplier; Supplier details; Request details (order ID, customer details, delivery options);	Requester	Supplier	Accept or reject request.	
Request rejected (undesirable state)	Request status = rejected;	Supplier	Requester	Requester sends a cancellation of the request	
Request canceled (undesirable state)	Request status = canceled	Requester	Supplier	Cancel delivery plans; End of interaction; within his private domain, the requester resends to next tentative supplier.	
Request accepted	Request status = accepted; Delivery details (planned);	Supplier	Requester, supplier	Requester – payment; supplier – delivery plans; Reset payment waiting timer;	
Payment done	Payment status = completed;	Requester	Supplier	Execute delivery;	
Payment enquiry (undesirable state)	Payment_status=enquiry;	Supplier	Requester	Send enquiry to requester;	
Delivery enquiry (undesirable state)	Delivery_status=enquiry;	Requester	Supplier	Send enquiry to requester;	
Item received (goal state)	Request status = received_by_requester; Delivery status = completed; delivery options;	Supplier	Requester	Deliver to requester following the delivery options; requester confirms receiving.	
Item delivered (goal state)	Request status = delivered; Delivery status = pending;	Supplier	Requester	Requester confirms delivery.	

Table 3. Current process states, state variables and performance constraints

We can note several interesting facts about these shared states:

(a) All shares states are stable states. That is, there are no transition from a state to another without the occurrence of a requester and/or a supplier triggered event, which is by definition (see stable and unstable state definitions in the GPM and BWW models description).

(b) Note that we postulated in our model the coordination of the VO is not centralized; in other words, we didn't adopt the assumption of a centralized design nor a centralized coordination of processes as did for example the PRODNET 2 project. Each partner must be free to design his own process as he considers adequate for his business. The states mapping done earlier is done independently for each party; what all parties must know and respect are the events patterns that are required for the specified shared states. We consider this a major strength of our model as this is the basic consideration for a distributed planning, design and further implementation of the VO process.

(c) The shared domain has no transformation law that is outside of the partners there is no continuum process flow; rather than that there is only a set of discontinuities represented as a set of stable states. This is very important result in what concerns the understanding of the inter-organization flow. In the inter-domain space, which we call here shared domain there is no process occurring. The processes occur only in the interior of private domains. In the in-between domains only exchanged discontinued states are visible, through the events that are exchanged between domains.

(d) Undesirable states are dealt with as additional model states and not separate, complex processes, as proposed by workflow models. This is yet strength of our model that is the undesirable state handling process paths become part of all possible process paths. This simplifies the model and as we will prove later on is one of the building blocks for process auditing and improvement.

(e) Each state is ultimately mapped into sets of state variables. A transformation occurs through changes in the state variables values and transformation triggering events are triggers that are activated by conditions set upon criterion functions that are applied to the state variables. Having this in mind, we see that the state variables of each state may be grouped in sub-groups that are relevant for each shared state. That is, a shared state is represented by a sub-set of the state variables of the private states from where the event flows depart. For example, the shared state "request received by supplier" would include

at least the state variables "request ID, Item ID, issuing time, required format", "delivery address", "delivery to home option" from the requester private state "request received from the requester" and the state variables "order ID, Order receiving time" from the supplier private state "request received from requester".

(f) The shared domain state change is always preceded by a sequence of a notification (through an event) that one party sends to another followed by a confirmation of the other side which causes the shared state transformation: for example consider the private requester state "payment sending" (Figure 2), in which the requester is supposed to send the payment for the requested item to the supplier (Table 3**Error! Reference source not found.**- state "payment done"): Event (1)- The requester sends the payment and; Event (2)- in response, the supplier confirms the payment. This sequence causes a new state in the shared domain "payment done". In Parallel, the states of the requester and the supplier domains change adequately.

6.4.6. Illustrations of the end to end VO process flow

Following we provide two illustrations of the VO process enactment.

(a) Consider the requester private state “wait for supplier response”. This is a stable state with respect to the requester sub-domain, which will transform only by an external event now. However, once this state is triggered (that is once the event "requester send request to supplier" occurs), it puts the supplier sub-domain in an unstable state ("request received from requester"), hence, the VO domain is still unstable and the process continues. The supplier (whose process is invisible to the requester) is expected to confirm the request transform the state of the shared sub-domain either to “Request accepted” or to “Request rejected”. Both these states put the requester sub-domain in an unstable state – if the request is rejected, then the requester cancels the request, which causes the transformation of the shared state to "response rejected") before issuing another one to another tentative supplier, while if the request is accepted (causing the transformation of the shared domain state to "Request accepted", payment should be made (the requester send the payment, the supplier confirms it and thus the shared domain transforms into the "payment done" state. In this case the supplier sub-domain also remains unstable (unstable state payment received), preparing for delivery.

(b) Let us consider the requester state “wait for supplier response” state; the requester configured a private timer for a maximum request response waiting time. This timer is a

private partner timer value and is not synchronized with other partners; Once this timer times-out an undesirable state is triggered within the requester, who cancels the request and restart the whole ordering process again, with the consequent rework and delays; as we will prove in the next section, this constitutes a major weakness of the process specification, as it may cause the process to incur in an already invalid request processing, due to message duplication or excessive message delay in the communication infrastructure between partners. Such unwanted situations may cause much damage to the Partners causing in some situation the partners to loose control over their process, causing data inconsistency. This affects a lot the process resiliency and prevents the current process automation, and complicates thus the process implementation and further operation and maintenance. We need a simple solution that does not constraint the process automation opportunities and the process operation. We discuss this issue in the next section.

6.4.7. Step 4 - Process Change Requirements assessment

Two major requirements were assessed by the library team. First, there is an urgent need to commit to delivery times to customers. Today, there is no enough certainty of what are the delivery time the ILL process can commit to; second, there is a need to specify different service levels, according to the requested item type, and set of required options.

6.4.8. Step 5- Sub-domain BP model change mapping

We proceed to analyze these two major requirements using the concepts of our model.

A detailed change analysis and mapping to BP model changes is provided in Annex 3.

As can be seen, the change requirements resulted in a major change of the whole model.

6.4.9. Step 6- Shared domain model change mapping

Following the analysis and implementation of the above requirements, we compile all resulting process modifications in the following table, which is based on the current state definition table, presented earlier. Each row expresses a state; for each state we

specify the main state variables; we also specify the expected action from the affected party and the performance constraints imposed on the expected action.

A detailed discussion of the process assessment is provided in Annex 3. Hereafter we reproduce the modified shared domain definition for convenience of the reader.

State	**State definition**	**Achieving party**	**Affected party**	**Expected action of affected party**	**Performance constraints**
Request received by supplier	Request status = sent to tentative supplier; Supplier details; Request details (order ID, order issuing time item details, customer details, delivery options); Required service level (maximum order processing time, item quality) ; Request_response_Timer= Initiated;	Requester	Supplier	Accept or reject request.	(1) Supplier must either reject or accept request within a specified maximum request response time; if not, a timer triggers a request canceled is triggered; (2) Total order processing_time < maximum order processing time.
Request rejected (undesirable state)	Request status = rejected	Supplier	Requester	Requester sends a cancellation of the request	
Request canceled (undesirable state)	Request status = canceled	Requester	Supplier	Cancel delivery plans; End of interaction ; within his private domain, the requester resends to next tentative supplier.	
Request accepted	Request status = accepted; Delivery details (planned); payment waiting timer = initiated; Order_processing_elapsed_time updated;	Supplier	Requester, supplier	Requester – payment; supplier – delivery plans; Reset payment waiting timer;	(1) Order processing_elapsed_time < maximum order processing time.
Payment done	Payment status = completed; Delivery waiting timer= initiated; Order_processing_elapsed_time updated;	Requester	Supplier	Execute delivery;	(1) Delivery within a maximum delivery waiting time; if not triggers a delivery enquiry state. (2) Order processing_elapsed_time < maximum order processing time

State	State definition	Achieving party	Affected party	Expected action of affected party	Performance constraints
Payment enquiry (undesirable state)	Payment_status=enquiry; Payment_waiting_Timer= initiated;	Supplier	Requester	Send enquiry to requester; Reset payment waiting timer;	(1) Payment must arrive within accorded maximum payment time; if not timer triggers a "request rejected" state. (2) Order processing_elapsed_time < maximum order processing time
Delivery enquiry (undesirable state)	Delivery_status=enquiry; Delivery_waiting_Timer= initiated; Order_processing_elapsed_time updated;	Requester	Supplier	Send enquiry to requester; Reset delivery_waiting timer;	(1) Delivery within a maximum delivery waiting; if not timer triggers a "request rejected" state; (2) Order processing_elapsed_time < maximum order processing time time;
Item received (goal state)	Request status = received_by_requester; Delivery status = completed; delivery options; Order_processing_elapsed_time updated;	Supplier	Requester	Deliver to requester following the delivery options; deliver confirms receiving.	(1) Order processing_elapsed_time < maximum order processing time time;
Item delivered (goal state)	Request status = delivered; Delivery status = pending; Order_processing_elapsed_time updated;	Supplier	Requester	Requester confirms delivery.	

Table 4. Changed shared sub-domain model: shared states, state variables and performance constraints

6.4.10. Step 7- Model validity check

(1) Partner sub-domain validation

(a) We verified that there are all law definitions are complete by identifying the state triggering and exiting conditions for each state in the supplier and in the requester domain model.

(b) We also verified that there are no law/goal inconsistencies through the verification that there are no infinite loops, deadlocks or exceptional termination for each one of the private states.

(c) For each stable state defined we verified that there is at least one external event that renders this state unstable. IN case where there are no external events coming from the interacting partner domain, we verified that there are timing constraints that would render the state unstable.

(2) Shared sub-domain validation

(a) We verified that all interaction state variables were externalized by each one of the partners and included in the definition of one or more shared state.

(b) We also verified that the set of allowed interaction state variable values include all required values to allow the correct interpretation of the events associated with the shared state.

(c) We verified that each Partner interaction state has an associated law that allows it to transform into an unstable state through a condition applied to the interaction variable.

(d) Finally we verified that the performance constraints associated with the shared state are mapped into adequate internal constraints over interaction states and that the private constraints include necessary conditions and condition values as specified within the VO obligation.

6.4.11. Step 8 - VO undesirable states handling.

In Annex 4, within the process assessment we mapped the states that do not contribute positively to process soft-goals, which we called following the VO model definition "undesirable states".

These states are signalized in the different partner sub-domains illustrations and shared sub-domains specification tables.

We also fine-tuned the definitions associated with these states to minimize their occurrence using the timing constraints such as the maximum response time waiting the requester waits for a response, the maximum delivery time the requester waits for the

item delivery, and the maximum payment time waiting for the supplier side. These timings eliminate the necessity for explicit handshakes between partners as explained in Annex 3 and thus reduces the overload and rework due to undesirable states occurrence.

6.5. Summary

We demonstrated through the application of our model that: first, there is a clear need to specify formal performance constraints on each shared state. Note that different performance constraints may be applicable to different interaction options (e.g. item types delivery time depends on whether it is a hard copy or electronic copy); second, for each interaction between partners, formal specification of interaction parameters is needed (allowed values and semantic definition). In the model terms, for each shared state, the partners must accord what are the state variable that would represent the negotiable options such as delivery times, payment waiting times, delivery options and formats, etc., while values for these parameters may be set by default or during the process execution; third, Partners must commit to these constraints and consider them as a VO obligation. The set of all triplets: {shared states, state parameters, performance constraints} form what we defined as the VO obligation.

The correct application of these guidelines would result, as proven earlier, at least in the process improvements in two different aspects. First, at the process operation level, it would lead to a simpler process implementation, higher process resiliency, higher automations opportunities, higher process performance, less undesirable states inducing. Second, at the business level, it would provide the VO with better customer delivery time with resulting improvement of revenues, serve to offer customized service levels, and allow the processing of higher volumes of transactions due to less rework and less undesirable states inducing situations.

7. VO BP model evaluation

7.1. Evaluation methodology

Following the model presentation and illustration through the ILL case study, we intend to evaluate our model with respect to existing research. We will compare our modeling results to other researchers result (related research projects were presented in chapter "related work").

7.2. Evaluation

We established a VO inter-organization process model based upon a generic process model that in its turn is based upon a generic ontology- The BWW information systems ontology.

We extended the above mentioned models by the concepts of VO obligations which we consider is applied in the shared domain between partners and composed of the agreed upon between partners shared states, state options (parameters) and performance constraints.

Our model does not constraint the VO to have a central governance/coordination and/or planning. Most proposed VO models and frameworks are built upon this constraining assumption. Such are the cases of the NIISP American project and VEGA and PRODNET II European projects. Does this assumption really hold ? Let's assume it does: in such a case, all participants must be synchronized by a centralized organism that may be one of the participants, such as the coordinator role in the PRODNET II. This coordinator should be in charge of the end-to-end process management, at least in the triggering, the execution coordination and the monitoring of the process. The end-to-end process design is also centralized. Thus all parties of the VO must introduce heavy changes in their process implementation for supporting the VO; this would further constraint their autonomy and create highly coupled systems. This contradicts the current trend in the IT domain, which is toward open, loosely coupled and distributed systems; moreover it contradicts the main assumption of most of the VO researchers – the autonomy and business independency of the VO partners.

Let us evaluate our model with respect to the characteristics of virtual organizations Porter [69][70] identified: 1) A web of companies each contributing resources, 2) Virtually vertically integrated, 3) Linked through inter-organization business and production systems, 4) Aimed at reduced business cycle time, and 5) Aimed at one-stop shopping. Note that by "resources" Porter referred to added value that each partner adds to the process. Thus characteristic (1) is supported in our model in term of the transformation concept, as state variables may represent relevant characteristics of resources that are transformed though the process instantiation. Characteristic (2) refers to a particular VO model that is a processing chain that transforms a primary material into a more elaborated one through a sequential set of transformation. Our model supports this transformation by the concepts of event, transformation and state variables. Referring to characteristic (3), it is inherent within the definition we adopted at the basis of our model. Characteristic (4) was proven through the previous analysis where we proved how the process is improved at an operational and business performance levels by identifying shared performance constraints and interaction options. Finally our model does not refer to customer facing processes, so characteristic (5) has not been modeled within our model. This would need to analyze customer facing processes and their correlation with the inter-organization processes we modeled.

In difference with [31], we didn't model the integration between processes through trust based mechanism. Our model objectives are to reduce the interaction between partners to minimum, conserving the partners' autonomy and independency to the maximum. Information is minimally shared, with the objective of optimizing the coordination through shared states and formally defined options and performance constraints. Trust is needed here in order to exchange the events as accorded, that is trust is traduced to the commitment of each partner to support the VO process enactment and build his internal processes following the VO obligations. Yet our model didn't refer to the value creation process as we centered our model on business process modeling by business process goals and not by the economic value contributed by the process enactment. Though we can easily model economic goals though the concept of soft-goals and hard-goals, we consider that the issue of economical value is not enough specified- that is, research need to be done in order to see to what extent do VO's share their economic objectives and business plans? (While, obviously, conserving their autonomy, privacy and independency).

Referring to dynamic vis. static form of VO's several authors [35][45][52][74] considered dynamic forms as more advanced forms or more even as inherent characteristic of VO's. Our model do not model time dimension explicitly, but more implicitly as explained earlier. We consider that our model models both forms with no difference, as the process has inherent negotiation of shared states, shared options by the means of shared states variables, exchanged events, shared performance constraints and mutual commitment to implement all of these within each partner organization.

Mowschowitz [62][63] considers that switching is the key managerial innovation of the virtual organization. His model maps four components of managerial activity that are essential to VO's: formulation of abstract requirements based upon customer needs or orders; tracking and analysis of concrete satisfiers – that is selecting adequate suppliers; dynamic assignment of suppliers to customer orders; and exploration and analysis of the supplier assignment criteria. Our model, as demonstrated during the case study, provides the means of mapping the necessary information needed for supplier selection, through options that are mapped to shared state variables. During the selection process, the required options are passed to tentative partners and the partner has to confirm his capability to support the required options. That is our model has built in exchange of options required in each specific transaction. Yet a more thorough study must be done in order to assure the validity of our model with respect to the value creation process the author considers is the main objective of the VO creation.

Most authors agree with that partners' autonomy must be conserved. Our model assure the partner autonomy by (1) reducing the interaction and internal process information exchange needed during the BP design to minimum (only the set of exchanged events and process goals are shared), (2) reducing the interaction during the process implementation mostly to non-existent and (3) reducing the human interaction by reducing the undesirable state volumes to minimum. Each partner, once the process goal defined can proceed to its own process design with no constraint other than the specified performance constraints to link him to the VO.

Obligations as major building blocks of VO coordination mechanisms were proposed previously, but most analysis was rather at a very high level with very few drilling to the granularity we attained in our study.

Our model is based upon generic process ontology. Its generality provide us with a solid yet a simple platform to develop, discuss and evaluate the VO process. Some of the VO research projects adopt the enterprise ontology. While this ontology was created in a

first place to support classical non-virtual organizations, it was extended by different researchers to adapt it to different forms of VO's. The resulting ontology, though considers much more aspects than our models currently support (such as agents, temporal dimension, etc.) we postulate that its complexity make its validation and verification quasi- impossible. Can we offer an alternative with our model? yet we can't affirm it for sure, as our model need to be further extended to support other yet not modeled entities.

Though the VO related issues are complex ones, we preferred to attack it with simple, one-level models and we didn't rush into establishing multi-levels models that though attractive are very hard to verify and discuss due to the high number of concepts they imply.

Our model though is a conceptual model and isn't related to any implementation methodology. In future work it can be done by mapping our ontology for example to workflow implementation models, providing us thus a complete process lifecycle management tool- commencing by the early requirements engineering, through business process modeling, business process verification, execution and monitoring models.

[47][49][48][100] the authors present contract ontologies which address a variety of legal contract issues; compared to our model the established ontologies do not focus on process issues, and is not anchored in a generic ontology like BWW's. The workflow representation is aimed at assisting process designers to design processes that implement the obligation, and also match existing processes with new ones. However, the clear distinction made by us regarding the separate domains (private and shared) does not exist there. Our clear distinction has two advantages: (1) it keeps the autonomy of the separate organizations and their private processes. (2) Matching existing processes with contractual obligations is much simpler, including only verification that the states for which the organization is obligated are achieved.

Private and public domains are present in [4], whose proposed workflow inheritance bears much similarity to our private and shared sub-domain concepts. However, our model provides a conceptual understanding rather than an implementation model, and addresses also soft-goal constraints, which are not included there.

Finally, our model can be extended further to cover open issues such as shared resources, non-virtual aspects of VO (processes), customer facing processes and their impact on inter-organizational issues, supply chains models, the human factor importance. The temporal dimension is not explicitly modeled in our framework; the

necessity of such model is put in question as we don't yet see any necessity to such an explicit entity modeling, whilst an implicit inclusion of the time dimension already exists in our model through the concepts of transformation and events.

8. Summary & conclusions

We studied VO inter-organization business processes and extended a generic process model - the BWW ontology extended by the GPM model.

We realized that the VO process has many similarities with the intra-organization process model and with previously researched inter-organization processes.

Yet we feel that the proposed coordination and collaboration models proposed in the literature has not matured enough in order to allow a formal methodology based requirement engineering and further design of such processes.

We propose to improve the coordination and collaboration mechanisms with in VO's through the identification, and establishment of what we called VO obligations between partners.

We modeled the interaction between partners as occurring within a shared domain, a domain that is visible to all interacting parties, while each one conserves completely the privacy and autonomy of his private domain, which scopes his own private business processes. The shared domain is where the VO obligation's are defined as a triplet composed of shared states, state parameters (including their associated sets of values and semantics), and performance constraints that are imposed on shared states.

We further proposed to assess the shared business process by first identifying the shared states, including the shared undesirable state states, identifying for each state its state variables. Focusing on the undesirable state, we assert that the parties must collaborate in order to analyze shared undesirable state states with the objective of identifying and further mapping previously unidentified private performance constraints and interaction options (that are modeled through appropriate state variables). This would reduce undesirable states volume, reduce errors in appropriate partner selection, better the resiliency of the business process, and provide the partners with the necessary infrastructure to provide differentiated service levels. It would also simplify the implementation and thus allow higher levels of automations.

Drilling down to a higher granularity level (a thing that we consider haven't been done enough in the reviewed research), and focusing on the inter-organizational view of the VO process, we realized we can conserve the autonomy and independency of the VO partners and renounce to a centralized governance and design of a VO, by basing the

VO design on the examination of the exchanged events between partners and defining shared states according to these events. These states exist in the "ether" between organizations, which we modeled as a shared domain, a domain that in contrast to private domains of each VO partner (domains that scope and hide each partner internal processes), is visible to all interacting partners.

The set of shared states, performance constraints and state parameters compose what we defined as the VO obligation that all partner must commit to and implement within their processes.

Furthermore, the VO obligation specification allows resolving undesirable state handling uncertainness and reducing undesirable state occurrences.

We postulate that most of the malfunctions of the VO would come from the shared undesirable state events, where the parties must put a lot of care to map all performance constraints and interaction options in order to make the VO more resilient and less prone to errors and undesirable states.

In the case study based upon the interlibrary loans business processes, we demonstrated the way the formal model depicts the VO situation. We saw how with the proposed process model, each partner knows with much higher certainty what to expect from its partner in each collaboration instance. The dialogue through a specified set of obligations with a specified set of shared states and associated state parameters, performance parameters (that are assured in each partner process implementation and further process instantiation). In each collaboration instance the requester chooses a partner that is adequate for the required end-user service options (as an example we saw the delivery to home options may be supported or not by a specific partner, so if needed a partner that implemented the required obligation parameter need to be chosen).

The new model permits to define the performance obligations of each partner. Each partner has a clear set of performance parameters with internal processes must support. These constraints are really constraints to the partner soft-goals and the each partner must design his internal processes to support and comply with them. Different services levels may be further associated with different performance levels; some partners would support a higher level and others a lower one, depending on their own business goals and willingness. This exemplifies our affirmation that the partner business autonomy and privacy are totally conserved: only the required collaboration related data is exposed to relevant partners, whilst the rest of the partner business remains unexposed (that is "kept private").

We consider that VO coordination mechanisms should be as simple and minimal as possible, allow the parties to be synchronized to the current state of the shared processes, assure the awareness of each party to the current process state, conserve a high level of process resiliency (process freezing, duplication of tasks due to duplicate messages, etc.), simplify the undesirable states handling mechanisms (should not require to build specific undesirable state handling processes such as is done in typical workflow models); further they should not limit the process automation and implementation possibilities. At the business level they should allow different level of service in each interaction and different negotiable options for each interaction.

The simplicity and generic characteristics of our model permit us to model both dynamic and static VO's without any difference. As far as we know, this is a novel approach. We consider that the temporal dimension of a VO is based upon the necessary negotiation between the partners of the VO obligations we presented in this research, in either case this must be done in order to the VO process be executed in an efficient way.

As affirmed by many authors (see chapter 3), certainty foments clearly trust and enables the VO to provide better services. Hence this model provides the basic building blocks to support the roadmap of the VO toward better service quality by supporting different levels of performance, and different obligation options, while each partner supports the obligations and options he wants and is capable to implement.

Referring to future lines of research, we consider that our model may be extended further to cover open issues such as shared resources, non-virtual aspects of VO (processes);

We also consider extending our VO model to include customer facing processes, with the objective of analyzing their impact on inter-organizational issues, supply chains models.

Finally, the human based coordination is an issue that needs to be analyzed further to see where after all there must be a human agent and how does it affect the VO automation capabilities.

Obligations may be implemented through contracts. An interesting possible direction of research is to extend our model to model contracts in general.

Annex 1. Inter-Library Loans (ILL) Business process analysis

Hereafter we present the analysis of the characteristics, strengths and weaknesses of the current process design and implementation.
The analysis is based on interviews with the library general manager and the ILL manager.

1. ILL major characteristics

For the library under study, the library ILL process functions as if it was a business unit with efficiency goals in front of other libraries and Colleges in Israel through a set of national commitments established through a set of national committees.

The library has no revenue related goals toward end users such as students and investigators. Some of its processes such as the Inter-library loans (ILL), has revenue and efficiency goals.

ILL process is concerned with the management of loans between Israel libraries and is basically centered on electronic items (journals, etc.). The Idea is to reduce the costs of purchasing items (specially electronic journals and content of interest in general) nation wide through the consolidation of the purchasing process by a national committee, which is in charge of the whole process of identification of possible providers, negotiation and selection of the best one (service and delivery quality, delivery time, price criterions) and the delivery of the purchased items to the library that are interested in them. The committee is also in charge of selecting through auctions process a nation wide delivery provider.

Major partners are:

- Other libraries (balanced relation: loaning & borrowing items), in Israel.
- Colleges in the North (mainly to provide them with items, in rare cases borrow from them)
- Hospitals, research centers. a customer-provider relationship

These are the three major types of existing mutual commitments

Two types of major customers exist referring to required delivery time:

- More urgent: hospitals & scientific researchers require lower delivery times (few days at maximum).
- Less urgent: arts, philosophy, social studies (up to 2 weeks)

Referring to Customers' service levels commitment: No service level commitment has been established (a default answer made to end users is "2 weeks" delivery time). Though for electronic items, existing university systems enable providing most required items within 2 days. This is thanks to automatic scanning systems and delivery to email and fax that the library acquired.

Prices are set by external national committees, though the service given by the studied library is known to be better.

Purchasing for all the partners libraries in Israel is done by a Central national unit for electronic items. It negotiates prices with providers.

Service delivery is done through one company selected a priori by the purchasing national Unit. The cost of delivery grew a lot because of delivery pricing per packet instead of by travel to the library.

2. ILL business process assessment

2.1. Assessment method

We used guided and unguided interviews to assess the ILL business processes general characteristics, weaknesses and strengths and to identify major roadmap objectives.
For each process the interview included a first unguided part where the interviewee was asked to describe the process in general, and then a guided part based upon previously prepared questions.
Information obtained from the library manager was correlated with the information obtained from the ILL manager in order to identify misunderstandings and thus limit the potential error.
In the following we present the results of the assessment.

2.2. Current methods of process design

Currently, each library implements its processes autonomously, though they all use the same ILS product and the same electronic items scanning and delivering system. Each library is distributed geographically in a different way (some have several sites, a site per faculty/department, others are less distributed), with resulting different organizations structures and roles.

All libraries comply with the guidelines regulated by the national committee. The committee is making a considerable standardization effort for pricing and catalogue issues especially. Though, the committee specifies a very generic guidelines with no constraints on performance or on service delivery quality issues (delivered item quality is not assured – bad item quality is not a rare issue and is unsatisfactory, delivery time is set by default to 2 weeks though electronic items may be delivered in 2 days maximum and even few hours, pricing is at flat rate because of the non-adaptation of the existing ILS to billing requirements). There are no regulated procedures for treating bad quality item delivery, non-delivery on time, payment procedures between libraries, etc. The service is a best effort service. The end-user has no options to get a differentiated level of service (even at a higher price).

An ILL order request to another library is treated as a message that is sent to another library. There is no uniform predefined set of parameters and performance constraints. There is a high level of uncertainty about the options the supplier library can comply with, the time delivery that the supplier can comply with, the quality of the item that can be provided, the availability of the item at the supplier library (specially for non-electronic items - books, journals, thesis, research reports, etc)

That is though there is a commitment between libraries and a willing to collaborate there are no clear mechanisms that allow this collaboration to be efficient neither from a quality nor from a performance points of view.

2.3. ILL business process weaknesses

During the interviews, the following main process weaknesses were identified:

- Low level of automation
- No clear business soft-goals- service quality is a best effort quality level:
- Little effort is invested in marketing ILL services.

- No Clear roadmap.
- Low level of satisfaction of the ILS system, low expectation of bettering it through customizations, low level of satisfaction of the ILL system provider.
- Before the ILS entered, there was an in-house system that satisfied the needs much better.
- Difficulties to customize the ILS due to unsatisfactory interface toward the ILS provider. Basic functionality needed is not supported (Example: Accounting).
- No integration with libraries outside Israel due to missing Hebrew language support. No integration with the OCLC for this reason.
- Cataloging is a project that is not scheduled, nor budgeted.
- Inventory management- is not done with enough frequency due to lack of resources (300K transactions per year, 800K items are being exchanged, per year).
- Giving a high quality service to the partners does not assure that the library would get an equivalent quality of service from its partners.
- In order to improve the service level in the whole network, it is not enough to improve the service in this library. It must be a synchronized effort through existing national comities.
- Cutting operational costs: costs and budgets were cut drastically in the last 4 years (30% in 4 years), with a reduction of 9% in human resources costs. It is a operational goal to cut costs in 10 % per year. In contrast, the University teaching & investigation teams have grown, and many new departments have been created. This had a bad impact on inventory management specially and on the service agility in general.
- There is no clear relation between the fame of the university and the quality of its library processes in general and ILL in particular.

2.4. ILL business processes main strengths

The following ILL business process related strengths were identified; some of them are general library strengths as well:

- The library in question is known to be a high quality library, from a service point of view and variety of "items" offering.

- The library manager considers that the Library has a competitive advantage in social sciences and law study, not in scientific computer and biology studies.
- They have the greatest number of transactions per year: 10 K loans to other libraries (& colleges), 5 K borrowing transactions.
- Their advantage is in their central library, not distributed as in other libraries. This gives them a better human resource and knowledge management and reduces operations delays (item search, item bringing, transport, etc).
- Many items are offered through different providers for redundancy. Though no special effort is done for setting a redundancy policy, many times when purchasing electronic items the partner offers a package of services which includes many features that weren't required, including sometimes items that are also provided in other packages.
- The library has its own team of IT experts and developers, a capability that allows the library to self provide itself with advanced IT solutions with a good cost to investment ratio.
- The library does not operate totally as a business organization as the partnership and reciprocity has a higher priority than revenue or profit goals. This may be seized as a strength with reference to the budgeting approach, where budgets may be dedicated to subject that in the short term do not look rentable.

2.5. ILL business process roadmap

During the interviews, we figured that, at the long run, it would be advantageous for the library if the following functionality would be put in place:

- Committed delivery and service times.
- An explicit code of practice as in the ICLC and that's gives place to a big variance in the service and delivery quality between partners.
- The level of automation in different libraries is not uniform: the studied library is for the moment, the only Israel Library to have the ILS system set, a set of scanning and delivery automation tools. Bettering the service must be supported by better levels of automations and standardization of services.
- Entering the OCLC would significantly improve the processes but the Arabic and Hebrew languages support makes it very difficult.

- Set different prices to different service levels (especially for scientific (physics, computer sciences, etc) & medical research, not in humanitarian studies such as philosophy). This would not change notably internal processes & operation but would help in setting priorities. There is a major difficulty to support such a pricing through the existing ILS system due to inflexibility of the ILS system.
- The ideal is to have mechanisms such as those established in the OCLC (for example, 4 days committed response time. In case it is not complied with, the request passes to another partner library). For these reasons, the preferred way of pricing is a flat rate per request, per service level type.
- Delivery to home feature development: A niche market is growing in this aspect, as document supply business organizations are growing in a fast rate.

Annex 2. VO model concepts summary

The VOM builds upon the GPM/BWW. For convenience we included hereafter all major concepts (GPM and VOM).

Domain. A *domain* is a part of the world of which we wish to model changes, representing the scope of our control. In ontological terms, a domain is a set of things and their interactions, and is represented by a set of state variables, which stand for the intrinsic and mutual properties of these things, including emergent properties of the domain itself.

Note, the state of the domain is determined by the states of the things included in it. However, due to interactions, emergent state variables of composite things or of the domain might exist.

A *sub-domain* is a part of the domain, represented by a subset of the domain state variables. A sub-domain may be in a stable state while the entire domain is in an unstable state, meaning that a different part of the domain is currently subject to changes.

A Domain stable state is a state that can only change as a result of an action of something outside the (sub)domain.

A Domain unstable state. is a state of the (sub)domain that must change. Whether a state is stable or unstable and how an unstable state might change is defined in terms of the *laws* that govern the states of the domain and their transitions:

A criterion function is a function on the set of states C: S $\rightarrow$ D, where D is a certain domain (of values). A criterion function maps the values of state variables into a domain where a decision can be made on whether the process achieved its purpose or not. The

mapping may be on a subset of state variables that are considered relevant for deciding whether the process has reached its "goal". The domain mapped into is then a sub-domain of the process domain.

A law is a function from the set of states S into itself.

A transition law *is* a function on the set of possible unstable states S_u into the set of states S. Implied in this definition is that the transition law is fully deterministic.

Note that External Events may affect the state of the domain while the process is in progress. Consequently, state variables that affect the law might change in ways not controlled in the process.

A transition law can be extended to all states as follows: for an unstable state the law is the transition law, otherwise it maps the state into itself. These extended laws are also referred to as ***domain laws*** (designated by L).

It is not guaranteed that a domain law will always lead to a stable state.

A *process* is a sequence of unstable states, transforming by law until a stable state is reached. A process is defined over a domain, which sets the boundaries of what is in a stable or an unstable state. Events that occur outside the domain are *external events* and they can activate the domain when it is in a stable state.

Goal. If $S=\{s \mid s \text{ lawful}\}$ is the set of possible domain states. Let $S_{st} \subseteq S$ be the subset of domain stable states. Then a *Goal* (G) is a set of stable states $G \subseteq S_{st}$.

A process Goal: A goal G will be said to be a *process goal* if every execution of the process terminates in G.

A *condition* is a logical expression E made of simple expressions of the form: R::= C rel g, where rel$\in$ {'>', '=', '<'}, where C a criterion and g is a value from the same domain as C, combined by 'AND', 'OR', 'NOT' and precedence indicated by '( )'.

This leads to the definition of a goal as: $G = \{s \mid E(C(s)) \text{ is 'true'}\}$.

A process model is a quadruple <S, L, I, G>, where S is a set of states representing the domain of the process; L is the law, specified as mapping between subsets of states; I is a subset of unstable states, which are the initial states of the process after a triggering external event has occurred; G is a subset of stable states, which are the goal of the process. Subsets of states are specified by conditions defined over criterion functions in the state variables of the domain.

A process starts when a certain condition on the state of the domain holds, and *ends when its goal is reached*, i.e., when another condition specified on the state of the domain holds.

The states in the goal set may differ from each other in the values of state variables. Nevertheless, they all meet the condition specified. The criterion function defines the set of state variables that are relevant for determining that the process has reached its goal.

A state projection on a sub-domain is done by considering the sub-set of state variables describing the sub-domain. This subset defines the state of the sub-domain.

A law L projection on a sub-domain. Considering a process defined over a domain; Let $\underline{x}=x_1 \ldots x_n$ be the set of state variables of the domain. Let $\underline{x}_0$ be a subset of $\underline{x}$. The projection of a law, L, on $\underline{x}_0$ is the mapping L defines on $\underline{x}_0$ when operating on $\underline{x}$.

The projection of the law over a sub-domain defines the way the process will progress in that sub-domain not considering other parts of the domain.

A Soft-goal is an order relation on states, which defines the desirability of different goal states, as it is likely that not all states in the goal are equally desirable.

The VO domain is a domain including all the partner organizations in the virtual organization.

The partner sub-domains are all sub-domains of the VO domain. The VO domain is captured as an aggregation of all its partner sub-domains, which are autonomous and *private*.

All partners' sub-domains are private sub-domains, that is their states, goals, and law cannot be controlled or viewed from outside their associated sub-domains.

Partner sub-domains interaction is through properties that are mutual to at least two partners. Each mutual property is reflected by a state variable value. When the value of such property is changed by one partner, this event affects the state of the other partner too and triggers a transformation in it.

The Shared sub-domain *is* the sub-domain that includes all the state variables that represent mutual properties of the partners.

Partner sub-domains processes are projections of the VO domain process. A process in the virtual organization, namely, a VO process, is a process that takes place in the VO domain. Since the VO is composed of private sub-domains, a VO process cannot be viewed or defined as a whole, but it is known to exist, as it is the purpose of the VO formation. Nevertheless, the internal processes of the partners, which are well defined and known within each partner sub-domain, are *projections* of the VO process over the partner sub-domain.

All shared states in shared sub-domain are stable, since all the transformations in the shared sub-domain are a result of external events.

The shared domain includes no transformation laws. The (sub)domain is always "waiting" for an external event for its state to be transformed. For the VO process to progress towards its goal, these events should be generated by the partner organizations.

The VO process may have various paths, reflected in a variety of possible states in the shared sub-domain. These possible states should be defined as part of the obligation set

between partner organizations, where each partner is obligated to the states that depend on events it should generate.

An obligation of a partner to a state is said to be "committed to by a specific partner" if the law defined in its private sub-domain assures the achievement of that state. The outcome of an obligation is that although an observer is not familiar with the projection of the VO process on a private sub-domain, he knows it is designed so that the state under consideration will be achieved.

VO obligations specify a ***set of states that all partners' domain must reach***. These may be the goal of the (local) process or some mandatory states within the process that constrain its course. These states are reflected through the shared states.

VO obligations are part of the shared sub-domain definition.

Undesirable states. Note that the VO process can interact with entities outside the VO domain, and may have discontinuity points dependent on events outside the VO domain. Hence, ***undesirable states***, where the process fails to reach its original goal, may still occur. To address such situations, states related to undesirable state handling paths, which we name "Undesirable states" should be included in the associated VO obligation definition.

Performance constraints. Are constraints on the response of each partner domain to the generation of events through the shared sub-domain. These constraints are mapped in each sub-domain to constraints on soft-goals over relevant states (states which definition includes variables affecting domains mutual properties).

Interaction state variable. A partner domain state variable that represents an inter-domain mutual property and hence enables the interaction between domains. Such state variable is a projection of a shared domain state variable. Each interaction variable

definition must include the allowed set of values which must correspond to a subset of the allowed valued of the shared state variable.

Interaction states. Domain states which definition includes interaction state variables are called interaction states. These states are stable states as they require an external event to transform. Conditions applied to the interaction state variables allow transforming the interaction state into unstable states and thus, allowing the progression of the partner process.

Annex 3. VO BPM approach Summary

For the Reader convenience, we summarize the proposed BPM model in the following tables (reproduced from Chapter 5).

Step #	*Step objective*	*Step tasks*
1	***Partner sub-domain BP design***	(1) Identify the 4 components of its processes: {S, L, I, G} (2) Identify stable and unstable states (3) Define state variables semantics and allowed values. (4) Identify external incoming events and outgoing events.
2	***Sub-domain interaction design***	(1) Shared state variables identification (2) External events mapping. (3) Shared sub-domain shared states identification.
3	***Performance constraints mapping***	(1) Map expected performance constraints for each mapped shared state (e.g. timing, quality, etc.) (2) For each constraint, private constraints are mapped within each partner's domain over its interaction state variables.
4	***VO process validity checking***	(1) Partner domain process validity check. (2) Shared domain validation.
5	***VO undesirable states handling***	(1) Identification of shared states that contribute negatively to performance constraints compliance. (2) Minimize the occurrence of these states. (3) Recheck the validity of the partner and shared sub-domains models.

Table 5. BPM approach steps summary

Step #	*Step Objective*	*Step tasks*
1	***Current Partner sub-domain BP model mapping***	(1) Identify the 4 components of its processes: {S, L, I, G} (2) Identify stable and unstable states (3) Define state variables semantics and allowed values. (4) Identify external incoming events and outgoing events.
2	***Current Sub-domain interaction model mapping***	(1) Shared state variables identification (2) External events mapping. (3) Shared sub-domain shared states identification.
3	***Current Performance constraints mapping***	(1) Map expected performance constraints for each mapped shared state (e.g. timing, quality, etc.) (2) For each constraint, private constraints are mapped within each partner's domain over its interaction state variables.
4	***Process Change Requirements assessment***	(1) Identify the change requirements for the VO process.
5	***Sub-domain BP model change mapping***	(1) Map change requirements into the partners' sub-domains models.
6	***Shared domain model change mapping***	(1) Map change requirements into the shared sub-domains model.
7	***Model validity check***	(1) Partner domain process validity check. (2) Shared domain validation.
8	***VO undesirable states handling***	(1) Identification of shared states that contribute negatively to performance constraints compliance (2) Minimize the occurrence of these states. (3) Recheck the validity of the partner and shared sub-domains models.

Table 6. BPM approach to VO process change modeling

Annex 4. Mapping required ILL Partner sub-domains BP changes.

1. BP model changes mapping for the committed delivery time change requirement

The first requirement formulated as a question: "once the customer orders an item of type X, in time T, what is the minimal time the entire ILL process takes in order to deliver the requirement to the customer"? That is there is a VO soft-goal related to time, constrained to "total ordering + deliver time < Maximum ILL order processing time".

Following our model, the potential reasons why this constraint is not complied with are: (1) either the process is not valid (not attaining its goal); (2) either that the soft-goal was not identified or (3) a constraint was not imposed to an already established soft-goal- that is part of the interacting partners do not know that such a constraint exist of the interacting; (4) Even if a soft-goal has been established the way to measure the total order processing time is not feasible (e.g. only the order receiver knows the time the order arrived and this time value is not communicate to suppliers).

Let us analyze each one of these reasons.

(a) Analysis of Reason (1) – the occurrence of the VO shared domain in an undesirable state is due to the invalidation of the process in one of the private domains- that is the private process is deviated from its normal course and its state transformed to an exception state. In the case study we have four shared exception states. We analyze each one of these separately.

Considering the "request cancelled" exception state: As we see in the internal states mapping of the requester several cases may lead to a cancellation: (1) a request rejection by the supplier – we deal with it in the "request rejected" exception state analysis; (2) the customer cancels its order – this is a situation that though leads to an exception is a "force majeure" situation which is above the control of the VO- hence is not relevant to our discussion; (3) the request times-out and the requester cancels it. Further more, we see in Figure 2 and Figure 3 that the request response waiting is an internal state of the requester and the supplier is not aware of the condition the requester has set to its process. The supplier is not committed to such a soft-goal and has not set its processes to comply with it. This may cause unnecessary exception triggering leading the requester to reselect a partner and redo the whole process, with consequent delays of the

order processing total time. In conclusion, we identified that the supplier is unaware of a law that the requester has imposed to its process and that may trigger unnecessary exceptions, causing time waste and prolonged total order processing time.

Considering the "request rejected" exception state: As we see in the internal states mapping of the supplier, several cases may lead to a rejection which then yields a request cancellation by the requester: (1) a supplier who can't comply with the specified order (e.g. required item format, quality, delivery options, etc.) rejects the request - this yields an exception but it is a situation we can do nothing about- it is inherent to the process of partner selection and is defined as part of the obligation between partners - hence is not relevant to our discussion; (2) the supplier awaits payments which does not arrive though a payment enquiry has been sent (We analyze the "payment enquiry" exception state in the next paragraph); In this case, we see that the supplier has set private payment timeout condition and a timer parameter and value to its "payment enquiry sent" private state. Once this condition is triggered within the supplier domain he rejects the request. Newly, we see in Figure 2 and Figure 3, that the condition that triggers the internal exception "request rejected" within the supplier domain from the "payment enquiry sent" private state is unknown and is an internal condition of the supplier; the requester is not aware of it. This, as in the previous illustration case, causes unnecessary exception triggering rework, with consequent delays of the order.

In conclusion, we identified that the requester is unaware of a law that the supplier has imposed to its process and that may trigger unnecessary exceptions which invalidate the process, causing newly time waste and prolonged total order processing time.

Considering the "Payment enquiry" exception state: while the requester may be processing the payment, the chosen supplier internal payment timer may timeout (the timeout period was too short or the requester processing may have taken some additional time). This causes the supplier domain to enter the "payment enquiry" exception state, where the supplier waits another period of time for the payment to arrive, sending an enquiry event to the requester. The requester is unaware of the condition the supplier has set upon his private state "waiting for payment" (e.g. timer value and timeout condition are private to the supplier) and has not constraint his process in order to send the payment in adequate timing conditions. If his processing takes another payment timer period of time the supplier would reject the request, triggering another exception "request rejected". In conclusion, in this particular undesirable state, we yield the same conclusion as the previous ones.

The same conclusions are yield if we analyze the "delivery enquiry" shared state.

In summary, following the analysis of the exception states, we see that exception may be triggered unnecessarily in the current process specification, as each partner sets his own constraints to his own process, such as the time he waits for a confirmation of the receiving of his order, the time he waits for the delivery of the requested item, the time he waits for the payment to arrive before he rejects the order. These non-synchronized parameters between the parties cause an unwanted variance in the service between partners, as each one sets his service standards, or better said "service level" to the level he sees convenient.

These situations are all unwanted, as they disturb the normal process path and causes delays, unnecessary activities, order reprocessing, order reissuing, data inconsistency, unexpected process failures and other situation that may require complex rollbacks and human intervention.

Of more importance, the partners' autonomy is highly constrained as their control on their processes and their capability to optimize them are reduced significantly.

Reconsidering the payment timing example, we add a state variable "time elapsed since order issuing by customer" which contains the current elapsed time since the order was issued to the shared states "payment enquiry" and "request confirmed" states, and establishing a shared constraint (performance constraint) which establishes a condition on these shared state variables that is triggered when a this state variable value crosses a a-priori specified timeout value, common a timer which is visible to both parties; that is a timer that is set within the shared domain. This timing parameter shared between partners, would be represented by a state variable within the "request received by supplier" shared state (the supplier would communicate its value within the "supplier confirms request receiving" event or in some cases a VO-wide parameter value may be set by default.

Similarly, we add shared state variables concerning the needed specific timing parameters, to the shared states "request received by supplier", "request accepted" and "delivery enquiry".

(b) Analysis of reason (2)- an unidentified soft-goal. Returning to our case study, in order to comply with a maximum order processing time all the partners must accord the required soft-goal- that is all partner must accord to share the soft-goal "maximum order processing time" and the shared soft-goal constraint "less than X days". Moreover this

soft-goal constraint may be parameterized by different values according to the item type. Today this required soft-goal and associated constraints parameterized by the item type is not shared by the partners. These are what we called performance constraints. The parameterization of the constraint is based upon the shared state variable "item type" which should exist in each state. Accorded values per each item type must by set a priori between partners.

(c) ***Analysis of reason (3) - a missing soft-goal constraint upon an already defined shared soft-goal.*** This may be considered as a sub-case of the previous one.

(d) ***Analysis of reason (4) – there is no accorded way of measuring the order processing total time***. Considering newly Figures Figure 2, and Figure 3, this situation is due to: (1) the required state variable "time elapsed since order issuing by customer" is not passed by the requester to the supplier; (2) the "formula" by which each partner calculates the elapsed time is not the same between partners- in our model terms, it is not sufficient for the partners to accord between them a shared performance constraint "total order processing time < X (X different for each item type)" ; they have to specify the way the order processing time is measured (which is in the model term the criterion function applied to the state variable " time elapsed since order issuing by customer" must be specified and agreed upon between partners.

Departing from the shared states definition in Table 3, we see that we have at least 4 shared exceptions states. We proceed to analyze these states.
We identify several timing related performance constraints that are currently kept private in partners domains:

1. Maximum Request response waiting time- at the requester side.
2. Maximum payment waiting time – at the supplier side.
3. Maximum delivery waiting time – at the requester side.

These are timing issues that were not accorded between partners in the past and each partner specified his own timing privately. Enquiries in case of time outs are done by human interactions. The levels of exceptions induced were considered excessive especially for the delivery sub-process.
Once these constraints identified, we proceed to the modification of the related states as following:

1. A state variable "maximum response time allowed" is added to the state "request received by supplier". The state variable would be assigned a default value.
2. A state variable "maximum payment waiting time allowed" is added to the states "request accepted", "payment enquiry" and "payment done". The state variable would be assigned a default value.
3. A state variable "maximum delivery waiting time allowed" is added to the "request accepted", "deliver enquiry" and "item received". The state variable would be assigned a default value.

This induces the partners to modify their respective state variables and constrain their relevant processes implementation to commit to this timing.

To illustrate the necessary modifications, we use the following three scenarios presented in the following 3 diagrams:

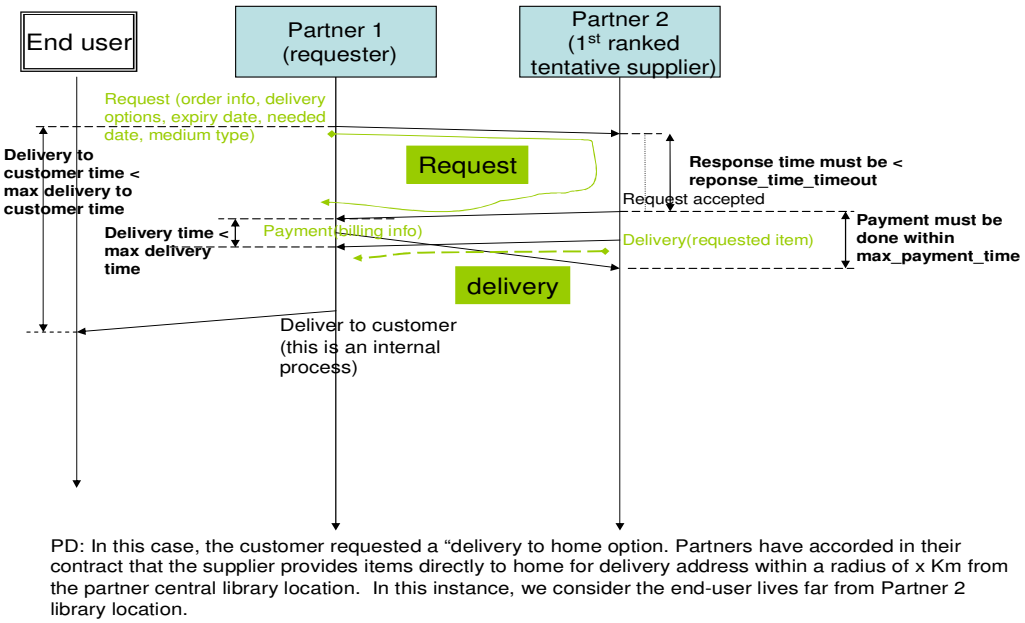


Figure 5. Scenario (a).

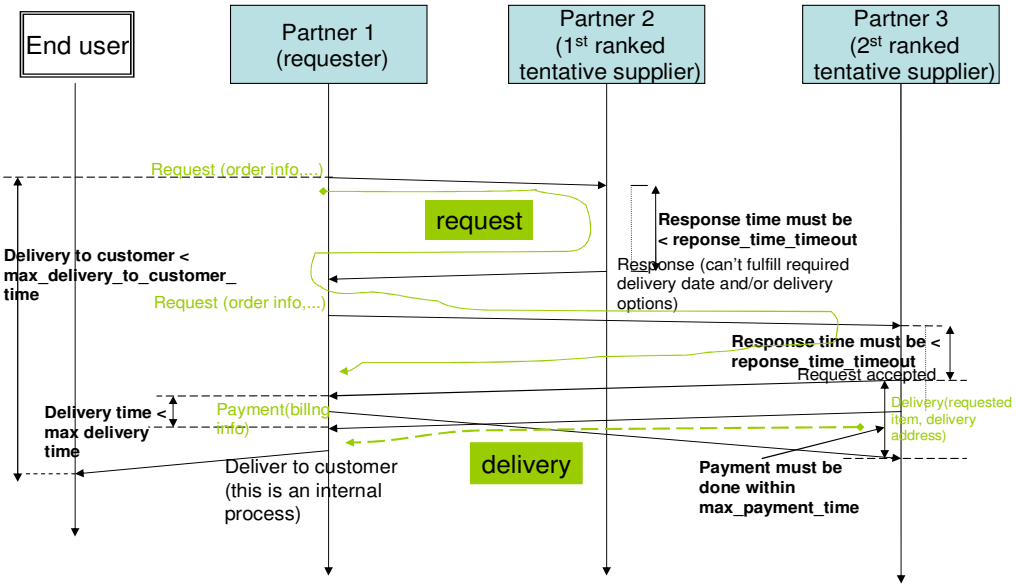
Scenario (a): An end user requests an item, with delivery to home option. Partner 1 library which receives the end user request, ranks the possible providers (ranking is an internal process of Partner 1 which is invisible to the VO partners) and sends the request to the first ranked Partner (Partner 2). The user lives far from Partner 2 library, so the delivery is made by partner 1.

This scenario illustrates the request, payment and delivery related VO obligations (this third one, with a particular path due to home delivery option requirement).

A Request related performance constraint is illustrated: Response time for a request must be less than a defined Response time timeout value.

A payment related performance constraint is illustrated through the maximum payment waiting time the supplier waits for the payment to arrive from the requester.

The delivery sub-process requires the establishment of a constraint upon the maximum delivery time the partner in charge of delivery must comply with. In this case the selection of the partner who delivers depends upon a condition that is the distance of the partner from the end-user address. If the end-user is within a specified number of km from the partner the partner would deliver the item to t he end-user house, otherwise the requester is the one who should take the charge of the delivery to home option.



PD: In this case, the customer requested a "delivery to home" option. We suppose this specific end-user lives far from Partner 3 library location, and thus the delivery to home is made by the Partner 1 library.

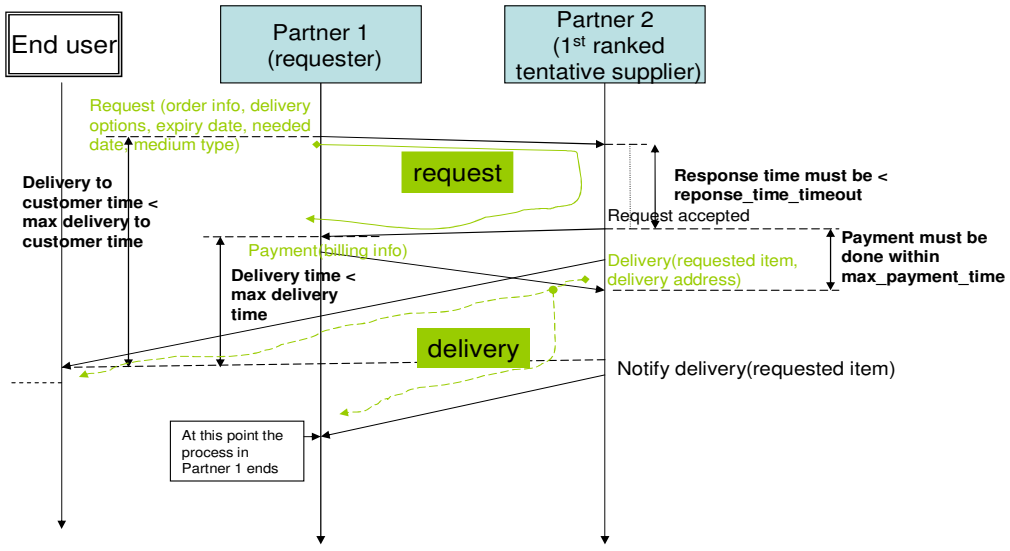
Figure 6. Scenario (b).

Scenario (b). An end user requests an item, with delivery to home option. The Partner 1 library which receives the end-user request, ranks the possible providers (ranking is an internal process of Partner 1 which is invisible to the VO partners) and sends the request to the first ranked Partner (Partner 2). But partner 2 did not answer the query within the maximum allowed response time. The request times out. Both sender and receiver are aware of this fact due to the constraints specified in the VO obligation. Knowing that, Partner 1 resends the request to the second ranked partner that is Partner 3, who accepts

the request and delivers the required item to the Partner 1, as the user lives far from Partner 3 library (this is a constraint specified in the delivery obligation).

Thus, we illustrate here, within the request obligation, an undesirable state handling scenario. For dealing with this exception, a shared state has been defined (see Table 7); whenever this timeout occurs, the request is implicitly cancelled (request timeout shared state) and the requester send a request to the second ranked partner (causing a transition newly to the Request Send shared state).

Once these performance constraints were set, the requester could estimate with a high level of certainty what delivery time he may propose to his customer for different items types. The result would be a great reduction in delivery times of electronic items especially in soft formats (files by email for example). In this case, in addition, a completely automated process would be considered for electronic items, allowing reducing further the delivery time to minutes, though complete automatic delivery with the customer self-ordering portal. This illustrates how clear obligations may enable automation opportunities not possible till then.



PD: In this case, the customer requested a "delivery to home" option. We suppose this specific end-user lives near Partner 2 library location, and thus the delivery to home is made by the Partner 2 library. The delivery process is thus a Partner 2 internal business process .

Figure 7. Scenario (c).

Scenario (c). An end user requests an item, with delivery to home option. Partner 1 library which receives the end-user request, ranks the possible providers (ranking is an internal process of Partner 1 which is invisible to the VO partners) and sends the request to the first ranked Partner (Partner 2), who accepts the request and deliver the required item to the Partner 2, as the user lives near enough Partner 2 library (this is a constraint specified upon the distance between the partner library and the end-user library, within the delivery related obligation definition).

This scenario illustrates a second possible path for the delivery obligation. The path is different from scenario (a) & (b), due to a condition triggered by a state variable value (delivery to home state variable was set to on and the distance of the end-user to partner library was within the range of partner 1 library).

Note that there may be a case where the partner selected does not support the delivery option, this would be identified in the request obligation as the partner would reject the request due to non-compliance of the required delivery to home option (while the end-user is within the range of its library).

The above scenarios require each of the interacting partners to design and implement its processes in such a way that complies with set performance parameters (see as example timers set upon request response, payment and delivery). These constraints impose constraints on internal business processes of each partner. Each partner who is committed to the VO complies with these constraints.

2. BP model change mapping for the committed customer service levels change requirement

Committing a service level to a customer implies that the ILL process must comply with the constraints required by this service level.

The ILL process requirement is to impose service levels per each type of item. The service level is to define the quality of the ordered item, the delivery times and the delivery options depending on the item type.

Different service levels would be priced differently so the billing process would be parameterized also by the item type.

So establishing the requirement for such a capability implies that:

(a) There is a need to define the different item types.

(b) There is a need to define the delivery time per each item type.

(c) There is a need to define pricing parameter per each item type

(d) There is a need to map the different options offered to the customer.

(e) There is a need to assure that all partners share the delivery time soft-goals, the pricing per item schemes.

Let us map these requirements by our model:

Requirement (a): mapping different item types, implies adding required shared state variables to the relevant states and exchanged events. The relevant exchanged events are the "requester send request"- in which we must specify the "item type" state variable transformation (value assignment). This state variable must also be available at all shared states of the model.

Requirement (b): today there is no certainty about what delivery times may be offered to a customer; we discussed this requirement in the previous section.

Requirement (c): the pricing schema can be defined within the VO per each mapped item. These would be VO wide a priori set values. This requires us to set shared state variables which would be assigned default values depending on the item type state variable.

Requirement (d): mapping different options implies a part of adding shared state variables, which triggers different paths within the VO process depending on their values. For example, adding an option such as the delivery to email option of electronic items would imply:

(a) To agree between partners to adding the delivery to email allowed value to the existing shared state variable "delivery option"

(b) Add a shared state variable "email_address" and "required item file format" to the relevant events and states. These shared state variables should be added to the "send request to supplier" event and the shared states "request received by supplier", "item delivered state".

(c) Add additional exchanged events and lawful transformation in each private domain. In this case, the delivery by email would not require additional states or exchanged events, though if we look at the delivery sub-process more in detail, the needed infrastructure is totally different (email capabilities are required, the whole process reduce to sending by email the item file. Though there is a necessity for the supplier to add a transformation law for the "payment received" state, that is: if "delivery option"="by_email", there is a new event that is

"supplier delivers item by email" that lead to the supplier domain private state "item delivered". The requester domain states and transformation laws remain unchanged.

(d) Add necessary shared states – in this case there is no such a need.

(e) Add relevant undesirable states and associated transformation laws and events. In this case, no change is required though this may be required in other scenarios.

(f) To add necessary soft goals and soft goals constraints- in this case there is no need for such modifications.

(g) Map shared performance constraints for the exception states- we didn't find such a need in this requirement mapping

As we see this is a major change to the ILL process as a whole.

3. Process assessment conclusions

Following the analysis and implementation of the above requirements, we compile all resulting process modifications in the following table, which is based on the current state definition table, presented earlier. Each row expresses a state; for each state we specify the main state variables; we also specify the expected action from the affected party and the performance constraints imposed on the expected action.

Note that each state variable has its allowed set of values. In the following table we do not enter this detail; note for example, that the variable delivery options set of values has been modified (we added an allowed value "delivery to email") in order to support the required delivery option. Also the variable required service level specifies the maximum order processing time, following the delivery time committed to the customer and item quality required.

State	State definition	Achieving party	Affected party	Expected action of affected party	Performance constraints
Request received by supplier	Request status = sent to tentative supplier; Supplier details; Request details (order ID, order issuing time item details, customer details, delivery options); Required service level (maximum order processing time, item quality) ; Request_response_Timer= Initiated;	Requester	Supplier	Accept or reject request.	(1) Supplier must either reject or accept request within a specified maximum request response time; if not, a timer triggers a request canceled is triggered; (2) Total order processing_time < maximum order processing time.
Request rejected (undesirable state)	Request status = rejected	Supplier	Requester	Requester sends a cancellation of the request	
Request canceled (undesirable state)	Request status = canceled	Requester	Supplier	Cancel delivery plans; End of interaction ; within his private domain, the requester resends to next tentative supplier.	
Request accepted	Request status = accepted; Delivery details (planned); payment waiting timer = initiated; Order_processing_elapsed_time updated;	Supplier	Requester, supplier	Requester – payment; supplier – delivery plans; Reset payment waiting timer;	(1) Order processing_elapsed_time < maximum order processing time.
Payment done	Payment status = completed; Delivery waiting timer= initiated; Order_processing_elapsed_time updated;	Requester	Supplier	Execute delivery;	(1) Delivery within a maximum delivery waiting time; if not triggers a delivery enquiry state. (2) Order processing_elapsed_time < maximum order processing time
Payment enquiry (undesirable state)	Payment_status=enquiry; Payment_waiting_Timer= initiated;	Supplier	Requester	Send enquiry to requester; Reset payment waiting timer;	(1) Payment must arrive within accorded maximum payment time; if not timer triggers a "request rejected" state. (2) Order processing_elapsed_time < maximum order processing time

State	State definition	Achieving party	Affected party	Expected action of affected party	Performance constraints
Delivery enquiry (undesirable state)	Delivery_status=enquiry; Delivery_waiting_Timer= initiated; Order_processing_elapsed_time updated;	Requester	Supplier	Send enquiry to requester; Reset delivery_waiting timer;	(1) Delivery within a maximum delivery waiting; if not timer triggers a "request rejected" state; (2) Order processing_elapsed_time < maximum order processing time time;
Item received (goal state)	Request status = received_by_requester; Delivery status = completed; delivery options; Order_processing_elapsed_time updated;	Supplier	Requester	Deliver to requester following the delivery options; deliver confirms receiving.	(1) Order processing_elapsed_time < maximum order processing time time;
Item delivered (goal state)	Request status = delivered; Delivery status = pending; Order_processing_elapsed_time updated;	Supplier	Requester	Requester confirms delivery.	

Table 7. Changed shared sub-domain model: shared states, state variables and performance constraints

Considering the payment waiting shared variable: such shared parameter has some additional advantages with respect to the currently specified internal private parameters:

- The requester knows he is expected to send the payment within a predefined period of time once he receives the request confirmation. If he accepts to support such an interaction with his VO partners, he would design, implement and configure his processes in order to comply with this constraint. That is this timer value becomes a performance or, in the model concepts - a soft-goal constraint upon the requester domain. Once this is done, the quantity of process incurring into "payment inquiry" and "request rejected" undesirable states may decrease (in some cases dramatically depending on specific timer values).
- As the requester is aware of the required timing constraints and accepts it as part of its VO obligations, the supplier has a higher level of certainty about the timing of his own process and thus can design and optimize his process independently of his VO interactions. This is partners autonomy is restituted.
- In the current process, the inability of the partners to specify timing requirements of its processes due to unknown performance constraints of other partners, drastically affects the VO parties capability to define delivery times to its customers. Returning to the previous example: as the requester can't assure that the item delivery timing is

bounded in time, he can't assure to its customer how much time it takes for him to deliver the order. What many organizations do, as in our study case, the requester specifies a default very high delivery time- in the case study a default of 2 weeks delivery time is specified. This is due to the fact that even if in some cases it may take 2-3 hours for the item to arrive, in other cases it may take a week, due to unacceptable item quality that cause reissuing the item request or other internal reasons of the supplier organization. In other words the variance is too high due to the uncertainty of the supplier timing. Once the supplier is obligated to performance constraints, the delivery time variance is reduces drastically and the requester can set, is a much higher certainty delivery times.

- Once the shared parameters are committed to within the VO, Premium services may be specified for customers who are ready to pay for better item quality, less delivery time. The issue of Service levels is a hot issue in today highly competitive business market and affects directly the business performance and revenues.
- Whenever it is relevant, different performance timing constraints may be set for different items types. That is done by specifying several state variables instead of one. This opens the possibility to offer customers customized service levels.
- VO partners accord between them that if the supplier does not confirm the request nor rejects it within a specified time let us say 2 hours, the request is automatically cancelled; this would simplify the interaction needed between partners referring to request sending and confirmation procedures, as there would be no need for handshakes between partners to assure the other party got the message; furthermore, this would eliminate the need for the requester to cancel implicitly the request when there is no answer from the tentative supplier within the maximum period of time accorded between partners for the request response time; when this period of time has elapsed all partners within the VO know that the request is no longer valid and even if the request was delayed due for example to unexpected communication delays and arrives later on to the supplier, the supplier who knows the time limits of the request validity would ignore it with no need to further notification from the requester. Thus we see that the process specification may be simplified due to implicit notifications-that is, fewer events must be communicated between domains. This simplifies further more the process implementation and enables additional automation opportunities.

- The formal specification of interaction options makes the VO process less error prone as there is no possibility to select accidentally an inadequate process path supplier due to "misunderstanding" (semantic erroneous interpretation, or incorrect options values).

Hence we affirm that the current VO process model does not impose enough obligations on VO partners; formal definition and commitment to performance constraints and formal definition of the options (and their associated semantics and values) in each interaction is needed. The current process definition causes a high level of uncertainness between partners with respect to the requested collaboration (uncertainness with respect to the current process state, the expected service level, service options, uninsured process resiliency, etc.). It compromises the ability of the partners to specify, design, control and maintain their processes, constraints notably the process automation and renders the process implementation less resilience and more complex. In addition, it does not enable delivering to the customer a committed level and quality of service, as there is no clear service commitment between collaborators. The fact it does not permit to develop different levels of services consequently reduces the opportunity to make higher profits that would be created though higher pricing for customers who wish to receive a higher quality of service (specially for delivery time, delivery options and items quality).

Now once the shared states are identified and a partner sub-domain model is established and validated we examine the undesirable state states. Undesirable state states are by definition caused by unwanted transformation of private processes of either interacting parties (as they deviate the process from its desired coarse). Undesirable states occur in two cases: (1) A private performance constraint has been violated. (2) An illegal shared state variable transformation has been required by an interaction domain.
Let us illustrate it through our case study- consider the shared state (See Figure 2 and Figure 3), "Request rejected". In the current process model, this undesirable state occurs in one of two scenarios: (a) the supplier rejects the request because the specified state parameters values are not supported – for example, the request requires a delivery to home option that he hasn't implemented, or a delivery of an item format that he does not supported both specified in the requester event (suppose this is due to recently implemented new delivery options and formats that were not yet adopted by a partner).

(b) The private supplier payment waiting timer timeout after a payment enquiry has occurred. Case (a) is an unlawful transformation of the request state variable "delivery method" that the requester requires to set to "delivery to home" which is not supported or the "delivery item format" which the requester requires to set to an unsupported value; while case (b) corresponds to a supplier private performance constraint violation.

Either scenario induces unnecessarily undesirable state states.

Thus the VO's BP assessment must have at least two objectives: First it must render visible private performance constraints which were not identified in the past such as timing and quality related, incorrect specification of an existing VO performance constraint, incorrect or incomplete semantics definition between parties that cause incorrect interpretations of events or required transformations, inadequate business process infrastructure use (e.g. unacceptable delays, processing power, low quality communication channels, etc.). Second, it should identify incorrect state variables settings that are due to not previously negotiated interaction options, or semantic incompleteness. This would induce state definition changes- such as the addition of state variables, modification of existing state variables (adding new values or eliminating invalid ones). It may also induce the specification of new shared states (for example, the creation of a new shared state "item delivered" which will support the new service of delivery to home option).

References

[1] A. I. Anton, C. Potts (1998), The use of goals to surface requirements for evolving systems, in Proc. of the 1998 Int. Conference on Software Engineering, pp. 157-166, 1998.

[2] A.I. Anton (1996), Goal Based Requirements Analysis, in Proc. Second Int. Conference on Requirements Engineering. ICRE '96, pp. 136–144, 1996.

[3] Aalst v. d. W., Hofstede A., Kiepuszewski B. and Barros A., 2003, Workflow Patterns, Distributed and parallel Databases, 14(3), p. 5-51.

[4] Aalst v. d. W. and Weske, M., 2001, The P2P Approach to Inter-organizational Workflows, Proceedings of CAiSE'01 (LNCS 2068), Springer-Verlag Berlin p. 140-156

[5] Afsarmanesh, H., Camarinha-Matos, L.M. (1997) - Federated Information Management for Cooperative Virtual Organizations, Proc. DEXA'97, 8th Int. Conf. on Databases and Expert Systems (Lecture Notes in Computer Science 1308, Springer Verlag), Toulouse, France, Sept 97.

[6] Afsarmanesh, H; Garita, C; Hertzberger, L.O.; Santos, V. - Management of Distributed Information in Virtual Enterprises : The PRODNET Approach – in proceedings of the Int. Conf. on Concurrent Enterprising (ICE'97), October 97.

[7] Belina,F., Hogrefe, D., Arma, A.: SDL with Applications from Protocol Specification,Carl Hanser Verlag and Prentice Hall International, UK,1991.

[8] Bider, I. (2002), "Evaluating Adequacy of Meta-models: A Practical Exercise in the Domain of Business Process Modeling", Research Report, IbisSoft AB, Available www.ibissoft.se/English/Howto.pdf.

[9] Black, J., Edwards, S. (2000). Emergence of virtual or network organizations: Fad or feature. Journal of Organizational Change Management, 13, 6, 567-576.

[10] Borst, W. N., Akkermans, J. M. & Top, J. L. (1997), 'Engineering ontologies', International Journal of Human-Computer Studies 46, 365–406.

[11] Bubenko, J., Rolland, C., Loucopoulos, P. and de Antonellis, V. (1994), Facilitating "Fuzzy to Formal" Requirements Modelling, IEEE International Conference on Requirements Engineering, 1994.

[12] Bunge, M. (1977). Treatise on Basic Philosophy: Volume 3: Ontology 1: The furniture of the world. Reidel, Boston.

[13] Bunge. M., Treatise on Basic Philosophy: Vol. 4, Ontology II: A World of Systems, Reidel, Boston, 1979.

[14] Bush, J.B., Jr., and Frohman, A.L. (1991), "Communication in a 'network' organization," Organizational Dynamics, 20(2), 23-36.

[15] Business Process Execution Language for Web Services, Version 1.0. IBM: see http://www.ibm.com/developerworks/library/ws-bpel/.

[16] C. Rolland, C. Souveyet, C. Ben Achour (1998), Guiding goal modeling using scenarios, IEEE Trans. Software Eng., vol. 24, pp. 1055–1071, Dec. 1998.

[17] C. Rolland, C. Souveyet, C. Ben Achour (1998), Guiding goal modeling using scenarios, IEEE Trans. Software Eng., vol. 24, pp. 1055–1071, Dec. 1998.

[18] Camarinha-Matos, L. M. and Afsarmanesh, H., (1998), Flexible coordination in virtual enterprises, in: Proc. of IMS'98, IFAC Internat. Workshop on Intelligent Manufacturing System, Gramado,Brazil, November 1998.

[19] Camarinha-Matos, L.M. (1997) - A platform to support production planning and management in a virtual enterprise, Proceedings of CAPE'97, IFIP International Conference on Computer Applications in Production and Engineering (Chapman & Hall), Detroit, USA, Nov 97.

[20] Camarinha-Matos, L.M., Afsarmanesh, H., Garita, C.; Lima, C. (1997) - Towards an Architecture for Virtual Enterprises, Proc. of the 2nd World Congress on Intelligent Manufacturing Processes & Systems, Budapest, Hungary, June 10-13, 1997.

[21] Camarinha-Matos, L.M., Carelli, R., Pellicer, J., Martín, M. (1997) - Towards the virtual enterprise in food industry, Proceedings of the ISIP'97 OE/IFIP/IEEE Int. Conf. on Integrated and Sustainable Industrial Production, Lisboa, Portugal, 14-16 May 1997, Chapman & Hall, ISBN 0 412 79950 2.

[22] Camarinha-Matos, L.M; Afsarmanesh, H. (Ed.s), (1999), Infrastructures for virtual enterprises –Networking industrial enterprises, Kluwer Academic Publishers, Oct 1999.

[23] Cantone, G., and P. Donzelli. (1999), Goal-Oriented Software Measurement Models. European Software Control and Metrics Conference. Herstmonceux Castle, East Sussex, UK, Apr. 1999.

[24] Chesbrough, H.W., and Teece, D.J., (1996), When is virtual virtuous? Organizing for innovation, Harvard Business Review, 74(1), 65-73.

[25] Cheung S.C., Hung P. C. K., and Chiu D. K. W., (2002), A Meta-Model for s-Contract Template Variable Dependencies Facilitating e-Negotiations, Proceedings of ER 2002 (LNCS 2503), Springer-Verlag, Berlin, p.50-64

[26] Clancy, T. (1994). The latest word from thoughtful executives – the virtual corporation, telecommuting and the concept of team. Academy of Management Executive, 8(2), 8-10.

[27] Clancy, T., (1994), The latest word from thoughtful executives – the virtual corporation, telecommuting and the concept of team. Academy of Management Executive, 8(2), 8-10.

[28] Cornelius Ncube and Neil Maiden (2000), COTS Software Selection: The Need to make Tradeoffs between System Requirements, Architectures and COTS/Components, April 19 2000. ICSE workshop, 2000.

[29] D. Harel. (1987), Statecharts: a visual formalism for complex systems. Science of computer programming, 8, 1987

[30] Dardenne, A., Fickas, S., van Lamsweerde, A. (1991), "Goal-directed concept acquisition in requirements elicitation", Proc. 6th IEEE Workshop System Specification and Design0 , Como, Italy, 1991, 14-21.

[31] Dardenne, A., Lamsweerde, v. and Fickas, S. (1993), Goal-directed Requirements Acquisition, Science of Computer Programming, Vol. 20, 1993, pp. 3-50.

[32] Enterprise Modelling and the Teleological Approach to Requirements Engineering, P. Loucopoulos & E. Kavakli. IJICS.

[33] Eric SK Yu, John Mylopoulos. " From ER to AR modelling strategic Actor Relationships for Business Process Reengineering ". In Proc. of the 13th International Conference on the Entity-Relationship Approach ER'94, Manchester (UK), December 13-16, 1994.

[34] Gangemi, A., Guarino, N., Masolo, C., and Oltramari, A. 2001. Understanding top-level ontological distinctions. LADSEB/CNR Technical Report 04/2001.

[35] Gil Regev and Alain Wegmann (2002). Regulation Based Linking of Strategic Goals and Business Processes, Workshop on Goal-Oriented Business Process Modeling, London, September 2002.

[36] Grefen P., Abere K., Hoffner Y. And Ludwig H., 2000, CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises, International Journal of Computer Systems Science & Engineering, 15 (5), p. 277-290

[37] Gruber, T. R. (1994), Towards principles for the design of ontologies used for knowledge sharing, in N. Guarino & R. Poli, eds, 'Formal Ontology in Conceptual Analysis and Knowledge Representation', Amsterdam, NL.

[38] Guarino, N. 1998a. Formal Ontology in Information Systems. In N. Guarino (ed.) Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. IOS Press, Amsterdam: 3-15.

[39] Hammer, M. (1996), Beyond Reengineering. New York: HarperBusiness.

[40] Hammer, M., and Champy (1993), J. Reengineering the Corporation. New York: HarperCollins.

[41] Harvey, M., Speier, C., Novicevic, M. (2000). An Innovative Global Management Staffing system: A Competency-Based Perspective. Human Resource Management Journal, Vol. 39 (3).

[42] http://www.bpmn.org/

[43] IDEF0 (1993), "Integration Definition for Function Modeling (IDEF0)", Draft Federal Information Processing Standards, Publication 183 (Fips 183). Available http://www.idef.com/Downloads/pdf/idef0.pdf.

[44] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis (1991), Telos: Representing Knowledge about Information Systems, ACM Trans. Info. Sys., 8 (4).

[45] Johan Vesterager, Martin Tølle, Peter Bernus, VERA: Virtual Enterprise Reference Architecture, http://globemen.vtt.fi/book/02_vesterager.pdf

[46] John Mylopoulos, Jaelson Castro (2000). Tropos: A Framework for Requirements-Driven Software Development, Information Systems Engineering: State of the Art and Research Themes, Lecture Notes in Computer Science, Springer-Verlag June 2000.

[47] Kabilan V. and Johannesson P., 2003, Semantic Representation of Contract Knowledge using Multi Tier Ontology, Proceedings of Semantic Web and Databases Workshop, (SWDB 2003)

[48] Kabilan V., 2005, Contract Workflow Model Patterns Using BPMN, EMMSAD'05, Proceedings of CAiSE'05 workshops Vol. 1, Porto, Portugal, p. 557-568.

[49] Kabilan V., Johannesson P. and Rugaimukammu, D., 2003, Business Contract Obligation Monitoring through Use of Multi-tier contract ontology, Proceedings of Workshop on Regulatory Ontologies (Worm Core 2003), Italy, (LNCS 2889), Springer-Verlag, Berlin

[50] Kafeza E., Chiu D. K.W. and Kafeza I., 2001, View-Based Contracts in an E-Service Cross-Organizational Workflow Environment, Proceedings of TES 2001 (LNCS 2193), Springer-Verlag, Berlin, p. 74-88.

[51] Kavakli, V. and Loucopoulos, P. (1998) Goal Driven Business Analysis: An Application in Electricity Deregulation, CAiSE*98, 8-12 June 1998, Pisa, Italy.

[52] Lee, R.M., Bons, R.W.H, Wagenaar, R.W. (2001) "Pattern-directed Auditing of Inter-organizational Trade Procedures", Towards the e-Society: Commerce, E-Business, and E-Government, Proc. of the First IFIP Conference I3E 2001, October 3-5, 2001, Zurich, Switzerland, Kluwer Academic Publishers.

[53] Lenz K., Oberwies A. and Schneider S., 2001, Trust-Based Contracting in Virtual Organizations: A Concept Based on Contract Workflow Management Systems, In: Schmid B., Stanoevska-Slabeva K., and Tschammer V. (eds), Towards the E-Society – E-Commerce, E-Business, and E-Government, Kluwer Academic Publishers, Boston, p. 3-16.

[54] Loucopoulos, P. (1994) The F3 (From Fuzzy to Formal) View on Requirements Engineering, Ingénierie des Systèmes d'Information, Vol. 2, No. 6, 1994, pp. 639-655.

[55] Loucopoulos, P. and Kavakli, E. (1995) Enterprise Modelling and the Teleological Approach to Requirements Engineering, International Journal of Intelligent and Cooperative Information Systems, Vol. 4, No. 1, 1995, pp. 45-79.

[56] Loucopoulos, V. Kavakli, N. Prekas, C. Rolland, G. Grosz, S. Nurcan "Using The EKD Approach: The Modelling Component" Technical Report WP2/T2.1/UMIST/1, Version 2.01 ELEKTRA P, Dept. of Computation, UMIST, UK.

[57] Maiden, N. A. M. & Ncube, C. (1998). Acquiring Requirements for Commercial Off-The-Shelf Package Selection. IEEE Software, 15(2): 46-56.

[58] Malone, T. W.; Crowston, K. – The interdisciplinary study of coordination, ACM Computing Surveys, Vol. 26 (1), 87-119, Mar 1994.

[59] Malone, T., Crowston, K., Lee J., Pentland, B., Dellarocas C., Wyner, G., Quimby, J., Osborn, C., Bernstein, A., Herman, G., Klein, M., and O´Donnell, E. (1999), "Towards a Handbook of Organizational Processes", Management Science, Vol 45, No 3, pp. 425-443.

[60] Mayer, R., Menzel, C., Painter, M., deWitte, P., Blinn, T., and Perakath, B. (1995), "IDEF3 Process Description Capture Method Report", Information Integration for

Concurrent Engineering (IICE), Interim technical report. http://www.idef.com/Downloads/pdf/idef3_fn.pdf.

[61] McCoy, D. Business activity monitoring (BAM)—deeper meanings, Business Integration Journal (Aug. 2003), http://www.bijonline.com/Article.asp?ArticleID=755.

[62] Mowshowitz, A. (1994), Virtual Organization: A Vision of Management in the Information Age, The Information Society, 10, pp. 267-288.

[63] Mowshowitz, A. (1997), Virtual organization, Communications of the ACM, Sep, 40:9, pp. 30-37.

[64] Mylopoulos, J., Chung, L. and Nixon, B. (1992) Representing and Using Nonfunctional Requirements : A Process-Oriented Approach, IEEE Transactions on Software Engineering, Vol. SE-18, No. 6, 1992, pp. 483-497.

[65] N Guarino, M. Carrata, P. Giaretta. Formalizing ontological commitment. In J.Doyle, E. Sandewall, and P. Torasso, editors, National Conference on Artificial Intelligence Conference (AAAI-94), Morgan Kaufman, Seattle, 1994.

[66] Neil A. M. Maiden, Cornelious Ncube & Andrew Moore (1997), Lessons learned during requirements acquisition for COTS systems. Communications of the ACM, 40(12), pp. 21-25, December 1997.

[67] Ould, M. (1995), Business Processes - Modelling and Analysis for Reengineering and Improvement, Wiley.

[68] Pnina Soffer, Boaz Golanya , Golanya, Dov Dori , and Yair Wand (2001). Modelling Off-the-Shelf Information Systems Requirements: An Ontological Approach. Requirements Eng 6:183–199.

[69] Porter, M. (1985). Competitive Advantage. New York: Free Press.

[70] Porter, M. (1990). Competitive Advantage of Nations. New York: Free Press.

[71] Potts C., "A Generic Model for Representing Design Methods", Proc. 11th Int. Conf. on Software Engineering, 1989.

[72] R. Medina-Mora, T. Winograd, and R. Flores, “ActionWorkflow as the Enterprise Integration Technology,”Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, Vol. 16, No.2, June, 1993.

[73] Reisig, W. (1985), Petri Nets: an introduction, Springer, Berlin.

[74] Roland Bauer, Sabine T. Köszegi, Measuring the degree of virtualization, Journal of Organizational Virtualness 2003, 5 (2003) 2, http://www.virtual-organization.net.

[75] Rolland C., Prakash N., "A Contextual Approach for the Requirements Engineering Process", Proc. Int. IEEE Conf. on Software Engineering and Knowledge Engineering (SEKE94), Riga, 1994.

[76] Rumbaugh, J., Jacobson, I., and Booch, G. (1999), The Unified Modeling Language Reference Manual, Addison Wesley.

[77] SEI CMM- Capability maturity model. See www.sei.cmu.edu.

[78] Shao, Y.P., Liao, S.Y. and Wang, H.Q. (1998), A model of virtual organizations, Journal of Information Science, 24: 5, pp. 305-312.

[79] Sheer,A.: ARIS-Business Process Modelling, Springer-Verlag, Berlin (1998).

[80] Singh, B. and Rein, G.L. (1992) Role Interaction Nets (RINs): A Process Definition Formalism, MCC, Technical Report, CT-083/92.

[81] Soffer P. and Wand Y., 2004, Goal-driven Analysis of Process Model Validity, Advanced Information Systems Engineering (CAiSE'04) (LNCS 3084), p. 521-535

[82] Soffer, P., and Wand, Y., On the Notion of Soft Goals in Business Process Modeling, Business Process Management Journal (forthcoming).

[83] T. Winograd and R. Flores, Understanding Computers and Cognition, Addison-Wesley, 1987.

[84] The NIIIP Reference Architecture, 1996, http://www. niiip.org.

[85] Van Der Aalst, W.M.P, Ter Hofstede and A.H.M: verification of workflow Task structures: A Petrinet-based approach, Information systems, vol. 25, no.1,2000.

[86] Wahlander, C., Nilsson , M.,Tornebohm J: Visuera PM Introduction, Copyright Viewlocity 2001.

[87] Wand Y, and Weber R, 1993, On the ontological expressiveness of information systems analysis and design grammars. J. Inform. Syst. 1993;3:217–237

[88] Wand, Y. and. Weber, R , 1990, An Ontological Model of an Information System, IEEE Transactions on Software Engineering, Vol. 16, No. 11, pp. 1282-1292.

[89] Workflow Management Coalition (2001), Workflow Management Coalition Interface: Process Definition Interchange Process Model.

[90] Workflow Management Coalition: see http://www.wfmc.org; Business Process Management Initiative: see http://www.bpmi.org; World Wide Web Consortium: see http://www.w3.org; OASIS: see http://www.oasis-open.org.

[91] World Wide Web consortium, www.w3c.org.

[92] Xiaohui Zhao, Chengfei Liu, Yun Yang. An organizational perspective on Collaboration business processes. BPM 2005, LNCS 3649, pp. 17-31, 2005.

[93] Yourdon E.E, 'Modern structured analysis', Prentice Hall, 1989.

[94] Yu, E. and Mylopoulos, J. (1994) Understanding 'Why" in Software Process Modeling, Analysis and Design, 16th International Conference on Software Engineering, Sorrento, Italy, 1994, pp. 159-168.

[95] Yu, E. Why Agent-Oriented Requirements Engineering? Proc. of 3rd Workshop on Requirements Engineering for Software Quality. Barcelona, Catalonia, June 1997.

[96] Yu, E., and Mylopoulos, J., 1996, Using Goals, Rules, and Methods to Support Reasoning in Business Process Reengineering, International Journal of Intelligent Systems in Accounting, Finance and Management, Vol. 5, pp. 1-13.

[97] Yu, E.S.K. (1993) Modelling Organizations for Information Systems Requirements Engineering, IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, San Diego, California, 1993, pp. 34-41.

[98] Yu, E.S.K. (1993) Modelling Organizations for Information Systems Requirements Engineering, in Proceedings of IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, San Diego, California, pp. 34-41.

[99] Zarli, A. et al – Integrating Emerging IT Paradigms for the Virtual Enterprise: Ghe VEGA platform. In proceedings of the Int. Conf. on Concurrent enterprising (ICE'97), Nottingham, England, October 97.

[100] Zdravkovic J. and Kabilan V., 2005, Enabling Business Process Interoperability Using Contract Workflow Models, On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, Cyprus (LNCS 3760) Springer-Verlag, Berlin, p. 77-93.

תהליכים עסקיים בארגונים וירטואליים: מודל קונספטואלי מבוסס אונטולוגיה

ג'וני גטאס

תקציר

ארגונים וירטואליים (VO's), כשיתוף פעולה בין ארגוניים אשר חולקים תהליכים עסקיים למטרות ספציפיות. הינו נושא שזכה לתשומת לב מחקרית רבה בשנים האחרונות. עבדת טיזה זו מציגה ניתוח מבוסס אונטולוגיה (ONTOLOGY) פורמלית לתהליכים עסקיים בארגונים וירטואליים, אשר מטרתו להשיג הבנה ברמה הקונספטואלית של האתגרים שבניתוח תהליכים בסביבה מעין זאת. המודל המוצג אמור לשמש במודל טאורטי לפיתוח מודלים לתמיכה וליישום בעולם זה, בד בבד שהוא אמפשר לארגונים השותפים להנות מהיתרונות של אמון וקואורדידציה ללא פגיעה באוטונומיה והפרטיות העסקית של כל אחד מהם. המודל מרחיבאת מודל ה- GPM (General Process Model) על מנת להוסיף מושגים אשר קשורים לתהליכים עסקיים בארגונים וירטואליים. במיוחד, המודל מתיחס להתחיבויות חוזיות בין ארגונים ושילובם לתוך ניתוח התהליך הארגוני. המודל מודגם באמצעות תהליכי השאלה בין-ספריית. כמו כן, מוצע מודל לניתוח ובדיקה של תהליכים עסקיים, אשר נוערך באמצעות עבודות מחקר אשר בוצעו בעבר בתחום זה.

תהליכים עסקיים בארגונים וירטואליים: מודל קונספטואלי מבוסס אונטולוגיה

ג'וני גטאס

עבודת גמר מחקרית (תיזה) המוגשת לאחר קבלת התואר "מוסמך"

אוניברסיטת חיפה
הפקולטה למדעי החברה
החוג לסטטיסטיקה

פברואר, 2006

תהליכים עסקיים בארגונים וירטואליים: מודל קונספטואלי מבוסס אונטולוגיה

מאת: ג'וני גטאס

בהנחיית: דר' פנינה סופר

פרופ' דויד פרג'י

עבודת גמר מחקרית (תיזה) המוגשת לאחר קבלת התואר "מוסמך"

אוניברסיטת חיפה

הפקולטה למדעי החברה

החוג לסטטיסטיקה

פברואר, 2006