# Business Process Flexibility in Virtual Organizations

Pnina Soffer and Johny Ghattas

University of Haifa, Carmel Mountain 31905, Haifa, Israel
spnina@is.haifa.ac.il
ghattasjohny@gmail.com

**Abstract.** Virtual organizations are perceived as a means for achieving flexibility. However, shared inter-organizational business processes may pose additional constraints on the internal processes of an organization and reduce their flexibility. The paper builds on a conceptual model of business processes in a virtual organization. The model aimes at identifying a minimal process definition to support the collaborative process while allowing flexibility of the internal processes. The model is informally presented through a case study of an inter-library loan process, and its implications on flexibility are discussed.

## 1 Introduction

Virtual organizations are perceived as a means for achieving flexibility. The formation of a virtual organization allows a partner organization to focus on core competencies while outsourcing various operations [6]. Flexibility is achieved by the ability to expand the variety of products and services offered to the customer, the ability to switch partners and select the appropriate partners for a given task.

However, shared business processes may pose additional constraints on the internal processes of an organization. The necessity to perform in coordination with other organizations and the resulting obligations may lead to a higher rigidity of the possible processes.

Various mechanisms at various levels of detail have been proposed for achieving interoperability or shared business processes among organizations. Most of them focus on implementation details (e.g., [2]). We claim that an implementation solution must rely on a solid conceptual model, depicting the essence of shared business processes and interoperability.

A key issue discussed in the literature with respect to virtual organizations and inter-organizational processes is the required balance between trust and control, visibility and privacy. Addressing this delicate balance, solutions vary from complete central control to pure distribution. [11][12][13][21][14][10] propose models where an inter-organizational workflow is defined as part or as result of a contract between organizations. The organizations are then contractually committed to the defined workflow (or to a partial definition). Different levels of visibility of a partner's internal process by the other partners at run time are also proposed and supported. In general, a high degree of central control and required visibility imposes constraints on the internal operations of an organization, thus reduces its flexibility.

This paper builds on a conceptual model of shared processes in a virtual organization, proposed by [8]. The conceptual model aims at identifying the minimal definition required to enable a smooth operation of shared processes, while allowing the partners a maximal degree of privacy and flexibility. The model is based on the formal Generic Process Model (GPM) and Bunge's ontology. In this paper we informally present it through a case study of an inter-library loan process, and discuss its implications on business process flexibility.

The remainder of the paper is structured as follows: Section 2 briefly introduces the main concepts of GPM, as a basis for the analysis of the case study, which is presented in Section 3. Section 4 discusses the model with respect to the flexibility it enables, and conclusions are presented in Section 5.

## 2 The Generic Process Model

GPM is based on Bunge's ontology [3][4], as adapted for information systems modeling (e.g., [18][20]), for conceptual modeling, and for modeling business process concepts.

According to the ontological framework, the world is made of *things* that possess *properties*. Properties 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. 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.

A *domain* is a part of the world, namely, a set of things and their interactions. It is represented by a set of state variables, whose values represent the state of the domain at a moment in time. 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 *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.

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 over 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.

## 3 The Inter-Library Loan Case Study

This section presents a case study of an inter-library loan process, as an example of a virtual organization (VO) business process.
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).

The Inter Library Loan (ILL) process is triggered by customers, i.e., students or researchers in research centers, universities, colleges etc. The customer asks for an item from a virtual catalogue that includes all available items locally and within the association. If the item is available locally, it is provided by the local library. If not, the information system of the local library shall search for tentative providers through the catalogue and rank them according to a set of parameters, such as delivery time, quality, price, etc. The system sends a request to the first ranked tentative provider and waits for response. Different scenarios may occur: the provider may accept the request and notify the requester, who should pay for the service before delivery is made by the provider. An alternative scenario is when the tentative provider does not respond within a given period of time. The request is timed-out and the requester may initiate a request to another tentative provider. All this process logic is normally established at the level of the consortium / association and each partner that joins the consortium agrees to comply with it.

Every partner library can play one of two roles in each occurrence of the ILL process: a requester or a provider. Figure 1 and Figure 2 present example state flows of a requester and a provider in the ILL process, respectively.

Each of the parties has its own private process that takes place within its domain of control, and has a defined (local) goal. These private processes entail states where interaction with a partner takes place. Figures 1 and 2 distinguish “internal” from interaction states, where each can be stable or unstable. We take a special interest in the interaction states. In GPM terms, a stable interaction state is a *discontinuity point* [16], where the process domain is in a stable state waiting for an external event to reactivate it so it can progress towards its goal. Specifically, the expected external event should be a result of an action of the other party. An unstable interaction state is a state that follows an external event, originated by the other party. Note that every stable interaction state in one of the figures has a corresponding unstable interaction state in the other figure.

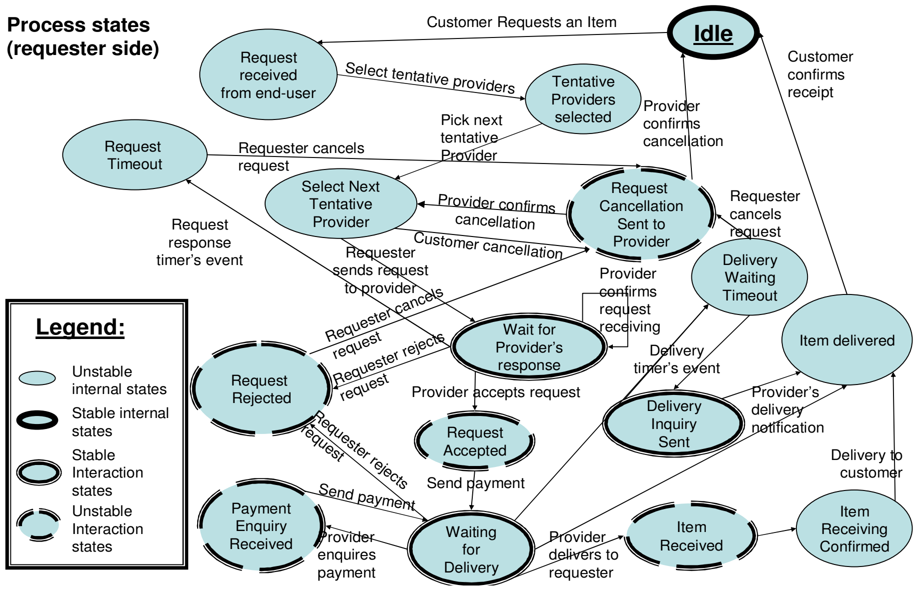
Process states (requester side)
Customer Requests an Item
Idle
Request received from end-user
Select tentative providers
Tentative Providers selected
Customer confirms receipt
Provider confirms cancellation
Pick next tentative Provider
Request Timeout
Requester cancels request
Select Next Tentative Provider
Provider confirms cancellation
Request Cancellation Sent to Provider
Requester cancels request
Customer cancellation
Request response timer's event
Requester sends request to provider
Delivery Waiting Timeout
Provider confirms request receiving
Item delivered
Legend:
Unstable internal states
Stable internal states
Stable Interaction states
Unstable Interaction states
Requester cancels request
Wait for Provider's response
Requester rejects request
Request Rejected
Delivery timer's event
Provider accepts request
Delivery Inquiry Sent
Provider's delivery notification
Request Accepted
Requester rejects request
Delivery to customer
Send payment
Payment Enquiry Received
Send payment
Waiting for Delivery
Provider enquires payment
Provider delivers to requester
Item Received
Item Receiving Confirmed

**Fig. 1.** State flow on the requester side

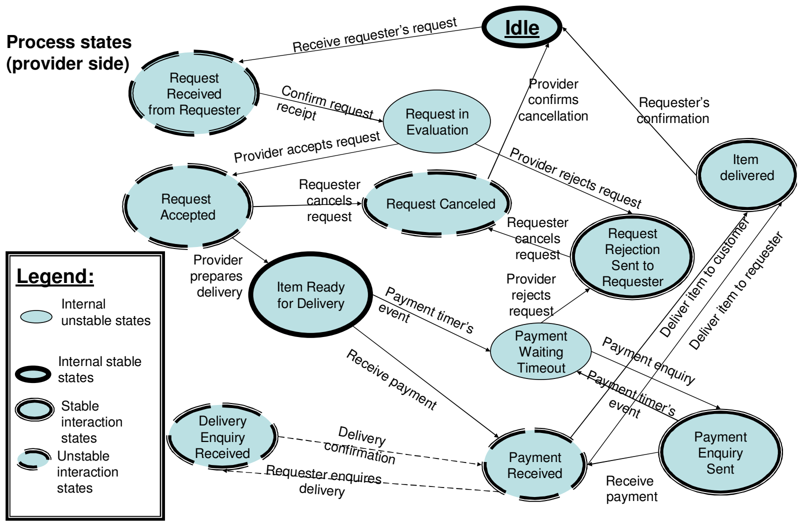

**Fig. 2.** State flow on the provider side

In order to streamline the overall VO process and to assure its validity as well as the validity of the private processes of each partner, these interaction points need to be defined and coordinated. Coupling the corresponding interaction states of Figures 1 and 2 yields the states specified in Figure 3, which are the shared states of the requester and the provider.

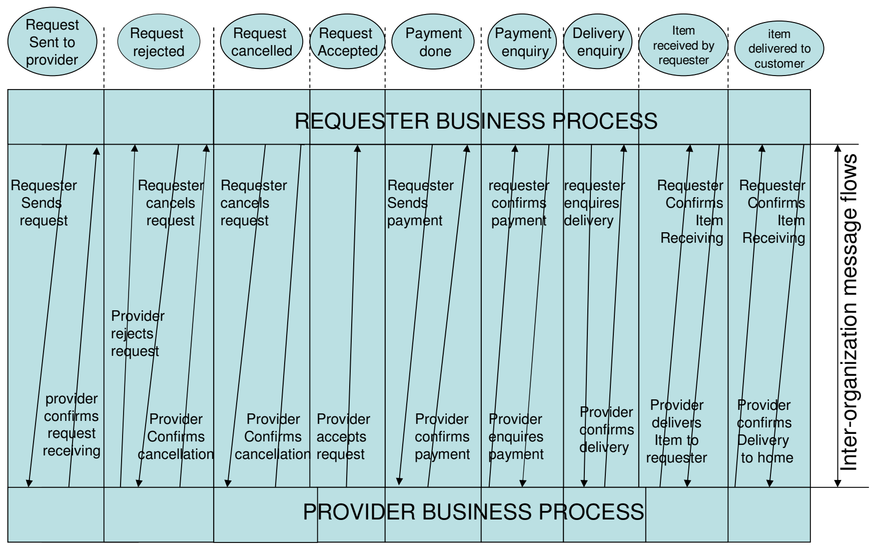
Request Sent to provider
Request rejected
Request cancelled
Request Accepted
Payment done
Payment enquiry
Delivery enquiry
Item received by requester
item delivered to customer
REQUESTER BUSINESS PROCESS
Requester Sends request
Requester cancels request
Requester cancels request
Requester Sends payment
requester confirms payment
requester enquires delivery
Requester Confirms Item Receiving
Requester Confirms Item Receiving
Provider rejects request
provider confirms request receiving
Provider Confirms cancellation
Provider Confirms cancellation
Provider accepts request
Provider confirms payment
Provider enquires payment
Provider confirms delivery
Provider delivers Item to requester
Provider confirms Delivery to home
Inter-organization message flows
PROVIDER BUSINESS PROCESS

**Fig.**
**3.** Shared view of the VO process

These states are shared in the sense that they are visible to both parties, whereas internal states are not viewed from outside the organizational domain. Each shared state is brought about by the action of one party and triggers action of (at least) the other party. Note that in general these states may also be a result of an event which is external to the VO (e.g., time). Figure 3 also specifies the events fired by each party in relation to each state. The events can be viewed as messages passing between the parties.

In order to streamline the VO process:

(a) These states should be defined in terms of state variable values.
(b) Constraints on Quality of Service (QoS) parameters can be defined and agreed upon (e.g., time to delivery).
(c) The parties should make an obligation to take the required actions in order to achieve the defined states.

Each one of these shall be discussed below.

## Shared state definition

The specification of the shared states is intended to define the state variables that are known to both parties and their required values. In fact, it sets the format of the message to be passed between the parties. For example, the shared state of *Request sent to provider* is specified by the following state variables: *Request status* whose value is "sent to tentative provider", state variables holding the provider details and the details of the request, which are *Order ID*, *Order issuing time*, *Item details*, *Customer details*, *Required delivery options*, and a state variable indicating the status of the *Request response timer*, which is initiated once the order is sent. The values of these state variables are set by the requester. This definition includes all the information needed for the provider to process the request and respond to it. The provider is expected to respond by changing the value of the *Request status* state variable (to “rejected” or “accepted”).

A complete definition and agreement of both parties regarding the shared states is necessary in order to facilitate the collaboration between the organizations. Consider, for example, a situation where the request is for a soft-copy of an item to be sent by email, but the provider’s process does not consider the *Required delivery options* state variable, and is capable only of sending hard copies. Including this state variable in the shared state definition and specifying its possible values should be a result of the negotiation between the parties during the VO formation.

**Constraints on Quality of Service parameters**

While the above discussed definition of the shared states is necessary for achieving the goal of the overall process (and of the internal processes of the parties), it is not enough for this process to achieve a desired quality of service. QoS relates to state variables which can indicate the desirability of different states where the process has achieved its goal. For example, consider two possible states where the customer has confirmed receiving the item, namely the process has reached its goal. However, one state is where the customer has received the item within two days, and the other is where it took a month for the item to arrive.

Setting constraints on the QoS of the entire process constrains the values of specific state variables in the internal processes of the parties. These constraints should be negotiated and agreed upon. As well, the definition of the shared states should include state variables which are relevant for these constraints.

In the ILL case study, the main QoS parameter identified is order processing time. Hence, an upper threshold, *Max order processing time*, was defined, depending on possible service levels offered to the customers. As a result, constraints were defined with respect to time taken for specific parts of the process. In particular, these constraints relate to phases where one party is in a stable “waiting” state while the other party is active. Furthermore, both parties share the overall constraint on the order processing time.

Note that in this case no penalty was set for not meeting the constraint. However, it is possible to define such penalty as part of the shared state definition.

Table 1 specifies the shared states of Figure 3, including QoS constraints and their related state variables. The table specifies the relevant state variables, their required values, the party responsible for achieving them, the party triggered to action as a result, and QoS constraints (where $T_1 - T_5$ are defined time thresholds). Note that the last two states (*Item received by requester* and *Item delivered to customer*) are identical in their shared definition, but different in the internal state variables of the requester, whose expected action in response to each state is different.

**Obligations**

The overall VO process, spanning at least two parties, is not centrally mandated, nor can it be entirely viewed by a single party. The partner organizations of the VO should commit themselves to their required parts of the process in order to establish the necessary trust among the partners, so a commitment to an end-customer can be made. Once negotiation and shared state definitions are completed, including agreement on QoS constraints, the partners should make an obligation to these states.

**Table 1**: Shared states definition

| State | State definition | Achieving party | Affected party | Expected action of affected party | QoS constraints |
|---|---|---|---|---|---|
| Request sent to provider | Request status =sent to tentative provider; Provider details; Request details (order ID, Order issuing time, customer details, delivery options); Required service level (Max order processing time, item quality); Request response Timer= Initiated; | Requester | Provider | Accept or reject request | Provider's response must be made within $T_1$ time; Order Processing elapsed time < Max order processing time |
| Request rejected | Request status = rejected | Provider | Requester | Requester sends request cancellation | |
| Request canceled | Request status = canceled | Requester | Provider | Cancel delivery plans; End of interaction | |
| Request accepted | Request status = accepted; Payment waiting timer= initiated; Order processing elapsed time= updated | Provider | Requester, provider | Requester – payment; provider – delivery plans | Requester must pay within $T_2$ time; Order Processing elapsed time < Max order processing time |
| Payment done | Payment status = completed; Delivery waiting timer= initiated Order processing elapsed time= updated | Requester | Provider | Execute delivery | Provider must deliver the item within $T_3$ time; Order Processing elapsed time < Max order processing time |
| Payment enquiry | Payment status= enquiry; Payment waiting Timer= initiated; | Provider | Requester | Payment | Requester must pay within $T_4$ time; Order Processing elapsed time < Max order processing time |
| Delivery enquiry | Delivery status= enquiry; Delivery waiting Timer= initiated; Order processing elapsed time updated | Requester | Provider | Execute delivery | Provider must deliver the item within $T_5$ time; Order Processing elapsed time < Max order processing time |
| Item received by requester | Request status = delivered; Order processing elapsed time updated; | Provider | Requester | Delivery confirmation | Order Processing elapsed time < Max order processing time |
| Item delivered to customer | Request status = delivered; Order processing elapsed time updated; | Provider | Requester | Customer receipt confirmation | Order Processing elapsed time < Max order processing time |

In GPM terms, an obligation means that the law operating in each private organizational domain is designed to achieve the agreed upon states. Based on the obligations made, although an observer cannot see the details of the entire end-to-end VO process, he has some information about it, which reduces uncertainty. An observer knows the entire process is designed so that certain events will take place, complying with certain constraints, leading to the obliged shared states, and eventually to the goal of the process. This is in spite of the fact that the internal process of each partner is completely private, and independently of the means by which the obligation is made (e.g., contract, human agreement).

## 4 Discussion

The proposed model, as demonstrated through the ILL case study, provides a minimal definition that facilitates the operation of a VO business process. At the same time it allows maximal privacy and flexibility in the internal processes of the participating organizations.

The literature dealing with inter-organizational business processes addresses both infrastructure and process models. Infrastructures, such as XRL/flower [2], may support flexibility and autonomy of the participating organizations. However, a specific process model should be designed and operated on top of the infrastructure, and constraints may be formed through this design. Comparing our model to other models proposed for inter-organizational processes in general and VO processes in particular, most of these models impose stricter constraints on the internal processes of the participants.

[11][12][13][21] address legal contracts between parties (organizations), and show how a detailed workflow can be derived from a contract. They also provide rules for matching the contract-based workflow with the existing organizational business processes. The contract-based workflow relates to the entire process, thus it allows no flexibility to a single organization for changing the process or deviating from it in specific cases.

In [14] privacy of organizational processes is maintained at run time, limiting the visibility of the internal process. However, the entire workflow has to be defined at build time, when a contract between the parties is established. Being contractually obligated to the workflow model, the flexibility of the parties is limited.

In [10], the contract specifies the internal process of the "service provider" party and required interface between the organizations, and allows limited visibility of the provider process. Here the process flexibility is reduced only for the "provider" party, and not for the other side.

The PRODNET project [5] is aimed at facilitating autonomy and heterogeneity of the organizations. However, their solution is based on a central mandating and coordinating party, to whom all parties must report. This requirement forms a limitation on the privacy and autonomy of the participants.

A model that facilitates the autonomy of partners through workflow views and inheritance is proposed by [1]. This model allows a relatively high level of flexibility to the partners. However, our model, unlike [1], addresses QoS parameters and constraints as well as the process flow.

Autonomy of the partners is also facilitated by the ebXML BPSS model [7]. However, this model is more detailed and less generic than ours. Its evaluation on the basis of Bunge's ontology [9] indicates a lack of ontological completeness as well as clarity, which may lead to modeling and interpretation difficulties. Specifically, the identified construct redundancy may imply that some ebXML BPSS constructs can be generalized and yield a more concise and clear model. The over-specification of ebXML can also be viewed as being rigid. For example, specific QoS parameters are part of the model, while there is no construct that allows the inclusion of others.

Clearly, when more constraints are imposed by the inter-organizational process, less flexibility can exist in the internal processes of an organization [18]. This rigidity applies to both process types and process instances [15]. The rigidity of the process type relates to (a) process design, when a new process type is designed in collaboration with the other parties; (b) the transformation of an existing process to an inter-organizational one, which requires matching the process details with the agreed-upon process [21]; and (c) modifications made to an existing process type, which must be coordinated with the other partners. The rigidity of the process instances is a result of the lack of freedom to deviate from the agreed upon process in exceptional situations, besides predefined agreed-upon exceptions.

Our model, in contrast, allows a partner organization to design and modify its internal process type and to deviate in specific instances, as long as the obligation to the shared states is kept. Within the boundaries of the obligations, any kind of change is possible, relating to all possible subjects (perspectives [15]) and having different properties (e.g., extent, duration [15]).

## 5 Conclusion

Flexibility, among many other benefits, is frequently associated with the formation of a virtual organization. However, shared inter-organizational processes may impose constraints on the internal processes of an organization and reduce their flexibility.

The case study presented in the paper demonstrates a minimal definition of a VO collaboration. It facilitates the achievement of the VO process goal, while allowing maximal flexibility in the partner's internal processes. We show that a process definition is possible despite the complete privacy and autonomy of the partner processes.

The underlying model to our approach is GPM, whose formality enables a precise definition of the terms involved, and process analysis possibilities.

Future research will address the implications of our models in terms of possible implementation solutions and systems supporting inter-organizational processes.

## References

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

[2] Aalst, W. M. P, van der. And Kumar, A., 2003, XML-Based Schema Definition for Support of Interorganizational Workflow, *Information Systems Research* 14(1), p. 23-46.

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

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

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

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

[7] EbXML BPSS specification, www.ebXML.org, Feb. 2006.

**[8]** Ghattas, J., 2006, Business Processes in Virtual organizations: an Ontology-Based Conceptual Model, MA Thesis, University of Haifa

**[9]** Green, P. F., Rosemann, M. and Indulska, M., 2005, Ontological Evaluation of Enterprise Systems Interoperability Using ebXML, *IEEE Transactions on Knowledge and Data Engineering* 17(5), p. 713-725.

[10] 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

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

[12] 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

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

[14] 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.

[15] Regev G., Soffer P. and Schmidt R., 2006, Taxonomy of Flexibility in Business Processes, http://lamswww.epfl.ch/conference/bpmds06/taxbpflex.

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

[17] Soffer, P., and Wand, Y., On the Notion of Soft Goals in Business Process Modeling, *Business Process Management Journal* 11(6), p. 663-679.

[18] Soffer, P., 2005, On the Notion of Flexibility in Business Processes, *Proceedings of the CAiSE'05 Workshops*, p. 35 – 42.

[19] 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.

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

[21] 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.