Design and Evolution of e-Business Models

M. Weiss*, D. Amyot†
*School of Computer Science, Carleton University
Ottawa, ON, K1S 5B6 (Canada)
Email: weiss@scs.carleton.ca
†SITE, University of Ottawa
Ottawa, ON, K1N 6N5 (Canada)
Email: damyot@site.uottawa.ca

***Abstract*— Companies need to adjust their business models constantly to changes in their environment. In this paper we propose a lightweight approach for evolving business models that allows for a quick evaluation of alternatives, while preserving investments in existing business processes. The approach is based on the User Requirements Notation (URN) for modeling and analysis of early requirements in the form of goals and scenarios. URN models help us model the strategic options available to a business for evolving its business model, and determine when the right moment to apply them has come. We illustrate the systematic and incremental evolution of business model alternatives for an e-business case study.**

I. Introduction

In today's rapidly evolving world, companies need to adjust their business models constantly to changes in their environment. However, they also need to do so in a controlled manner. An approach to evolving business models needs to strike a balance between capitalizing on new opportunities, and entering uncharted territories by mitigating the risks involved with such a change. The approach must be lightweight in order to quickly evaluate alternative models, but also be reliable.

We argue that the User Requirements Notation (URN) [1], [3] enables such an approach. While it allows us to explore alternative business models, it addresses the need to preserve investments in existing business processes. Business processes can be expressed as scenarios, which are defined separately from the participants in the business model that perform them. This allows us to experiment with different business models without changing the underlying business processes.

URN has many concepts relevant for business process modeling, such as behavior, structure, goals, and non-functional requirements. URN combines two complementary notations: the Goal-oriented Requirement Language (GRL) [5], and the Use Case Map (UCM) notation [6]. GRL captures business or system *goals*, alternative means of achieving goals, and the rationale for goals and alternatives. The notation is especially good for the modeling of non-functional requirements.

A UCM model depicts *scenarios* as causal flows of responsibilities that can be superimposed on underlying structures of components. Responsibilities are scenario activities representing something to be performed (operation, action, task, function, etc.), and can be allocated to components. Components are generic enough to represent software entities (e.g., objects, processes, databases, or servers) as well as non-software entities (e.g., actors or hardware resources).

Our example is based on a WS-I (Web Services Interoperability) case study [11], [12]. These documents describe a simple supply chain management system in terms of use cases defining the use of Web services in structured interactions and identifying basic interoperability requirements. The use case model integrates high-level functional requirements, a set of simplifying assumptions, and eight use cases and activity diagrams. The main functional requirements are:

- Retailer offers electronic goods to Consumers.
- Retailer must manage stock levels in Warehouses.
- Retailer must restock a good from the respective Manufacturer's inventory, if the stock level in one of its Warehouses falls below a certain threshold.
- Manufacturers must execute a production run to build the finished goods, if a good is not in stock.

In a recent contribution [8], we showed how a UCM model could be extracted from such use cases and informal requirements. We argued that URN offers suitable and useful features for modeling and analyzing business processes, and meets the goals of a business process modeling language. In other work [9], we explored how a similar UCM model could be designed given a set of business goals and informal requirements expressed in GRL as a starting point.

In this paper, we first summarize our existing work on business process modeling (Section II). We then focus on business model design and evolution. Various ways to evolve the business model from the Manufacturer's point of view are explored in Section III. Section IV concludes the paper.

II. Designing e-Business Models

A. Business Goal Model in GRL

Business goals describe *why* particular activities are performed in a business process. Figure 1 shows the GRL model for a manufacturer that sells to stock via warehouses (a.k.a. distributors), and retailers. Given the dominant role that the intermediaries (retailer, as well as warehouses) play, we will also refer to this e-business model as the **R** (Retailer) option. Like [7] we define an *e-business model* as a set of participants (including the firm of interest) and the value flows between them. This model represents each participant in the business

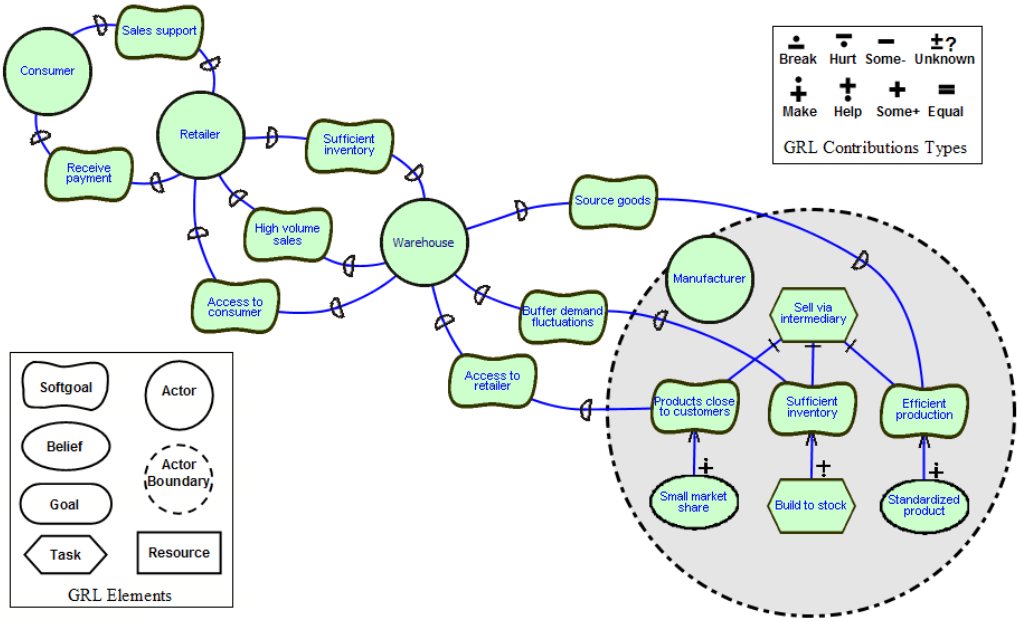

Fig. 1. GRL actor diagram: Sell to stock via warehouse and retailer (R) strategy.

model (consumer, retailer, warehouse, and manufacturer) as an *actor* (circle), and indicates their *dependencies* (–D–).

The Manufacturer actor is expanded (dotted circle) to reveal its internal *goals*. There are two *tasks* (hexagons) that the manufacturer performs, Sell via intermediary and Build to stock. The Sell via intermediary task is decomposed into three *softgoals* (where softgoals, shown as clouds, are goals that can never be fully satisfied). Tasks, goals, and softgoals can be recursively refined via such decomposition.

We can also model the preconditions under which the manufacturer should consider this business model. They are represented by *beliefs* (ellipses). One belief is that the manufacturer has a Small market share, and relies on the warehouses to position its Products close to customers. Another is that it can enjoy Efficient production levels as long as the market demands a Standardized product. Therefore, the levers for evolving this business model are *strategic moves* that increase the market share or make the product more differentiated.

B. Scenario Model in UCM

In the GRL actor diagram in Figure 1, we have identified several actors which will be shown in UCM models as *agent components* (rectangles with thick lines). They have to be involved in the support of the various functionalities identified in the informal requirements (e.g., those from Section I).

UCM models often start with a single top-level map, or *root map*. A possible root map for our business process is shown in Figure 2. A consumer visiting the retailer Web site expresses her intent to purchase goods by submitting an order. The retailer system replies by fulfilling the order. There are two outcomes: RejectOrder, or ShipmentConfirmed.

In the UCM notation, scenarios are initiated at *start points*, represented as filled circles, and terminate at *end points*, shown

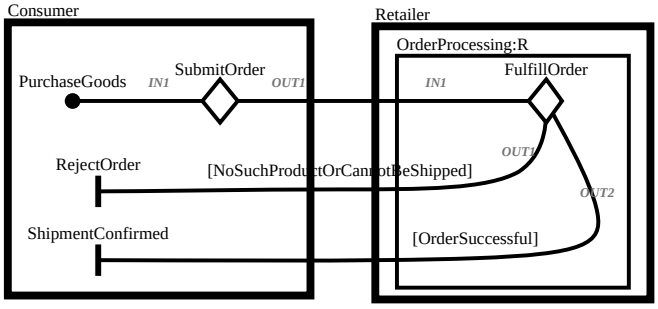

Fig. 2. Sell-to-stock root Use Case Map.

as bars. *Paths* show the causal relationships between start and end points. Components are responsible for the various activities (called *responsibilities* and indicated by X's on a path) allocated to them. As a convention here, we use UCM agents (thick lines) to represent GRL *actors*.

Additionally, team components (thin lines) are used to capture the various *roles* an agent can play. Several roles (e.g., OrderProcessing, InventoryManagement, and Production in our example) can be associated with an agent by nesting components. In the following UCM models, we use :R to indicate that a role is associated with a retailer agent, :M for a manufacturer, and :W for a warehouse. Several UCM diagrams will not be shown here due to space constraints, however the reader is invited to consult [8] for a complete example.

Diamonds are used to represent *stubs*, which are containers for submaps called *plug-ins*. Stubs have named input and output segments (e.g., *IN1*, *OUT1*, and *OUT2* in Figure 2) that are bound to start and end points in a plug-in, hence ensuring the continuation of a scenario from a parent map to a submap, and to the parent map again. The Sell-to-stock root

map contains two stubs: SubmitOrder and FulfillOrder.

In FulfillOrder (not shown) the retailer checks with its warehouses whether they can supply the items in the order (assuming the requested product exists), and asks them to ship the items. A SourceGoods stub handles the interaction with the warehouses. The corresponding plug-in is shown in Figure 3.

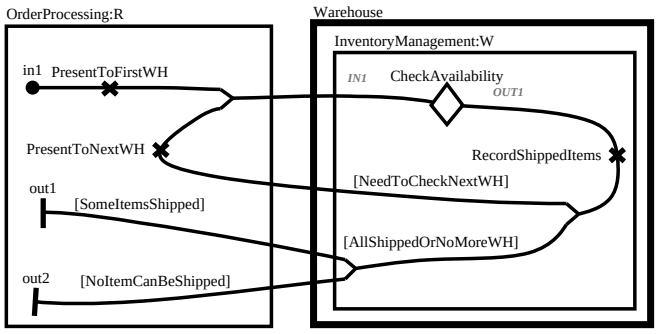

Fig. 3. SourceGoods plug-in for FulfillOrder submap.

The CheckAvailability submap (not shown) describes the iteration through the list of items presented to an individual warehouse. Whenever an item is available, the ordered quantity is decremented from the warehouse inventory. A Replenishment dynamic stub handles the update of the stock while the items are shipped. One plug-in for that stub, selected when no replenishment is required, would simply be a straight, pass-trough connection from *IN1* to *OUT1*. The other plug-in is shown in Figure 4. The warehouse orders goods from manufacturers to replenish its own stock for a given product.

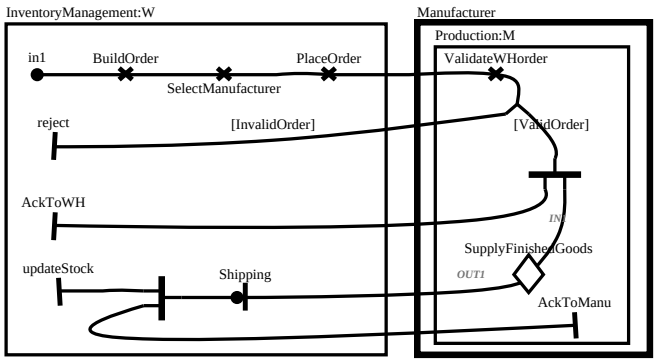

Fig. 4. ReplenishStock plug-in for CheckAvailability submap.

C. Linking Goals and Scenarios

In URN, various traceability links can be created between GRL and UCM models. If a GRL model is fine grained, then detailed elements such as GRL tasks and goals can be linked to specific UCM responsibilities, path segments, scenario definitions or entire plug-in maps. GRL goals and softgoals such as those in Figure 1 can be refined into high-level tasks (not shown here), and those tasks into low-level UCM responsibilities. This provides a traceable rationale for the scenarios and their responsabilities, hence explaining *why* they exist and are structured in this way.

From another perspective, UCM models explain *what* the activities related to a business goal are (responsibilities and scenarios), *who* is involved in these activities (actors and components), *where* they are performed (allocation to components/agents or subcomponents/teams), as well as *when* they should be performed (via constructs for expressing sequence, choice, concurrency, timers, and synchronization).

GRL models also allow analysts to link business or system goals to architectural alternatives, and thus to document the rationale for a particular choice. For instance, in [1], [8], several ways of allocating UCM responsibilities to components are explored and the decision is based on the contribution of each alternative to the satisfaction of higher-level GRL goals such as performance, reuse of current infrastructures, and maintainability. Another use for GRL models consists in considering and evaluating different configurations of actors, or allocation of roles to actors. This aspect will be further explored in an evolution context in Section III.

III. Evolving Business Processes

A. Evolution of Business Goals

Our working hypothesis in this paper is that we can use the *same* scenario to describe *different* business models and to reason about them. The fundamental underlying concept of UCM is the separation of the definition of a scenario from its allocation to components. Allocations can be reasoned about, and compared using GRL models. This concept lays the basis for an incremental *evolution* of the business model.

Consider the options available to a manufacturer that currently sells its products via intermediaries. As indicated in Figure 1, the manufacturer could consider actions that result in either one or both of the preconditions Small market share and Standardize product to change. Exploring those options leads to several possible evolutions of the business model available to the manufacturer, which are summarized in Figure 5.

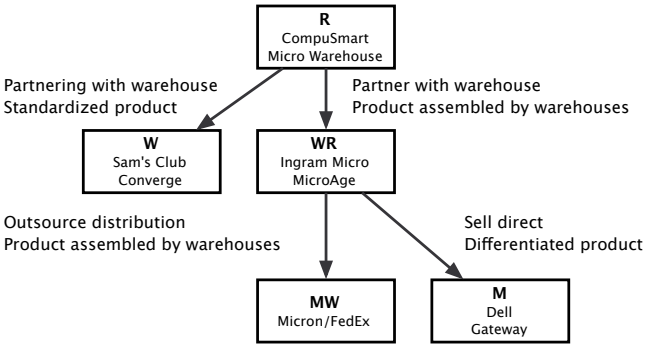

Fig. 5. Different ways for a manufacturer to sell its products.

The initial option (**R**) represents the manufacturer's current model. We name it for the dominant role played by the retailer in controlling access to the customer in this model. The arrows indicate the evolution between these business models,

and the labels on the arrows characterize the nature of the transition between the models. For example, the transitions from **R** to **W**, and **R** to **WR**, are both about increasing market share. However, in the transition from **R** to **W** the manufacturer keeps selling a standardized product, whereas in the other transition, it can offer a differentiated product. It is the warehouse that assembles the customized product. In both options the warehouse keeps control of order processing.

The manufacturer could increase its market share by partnering with a warehouse. This leads to either the **W** (Warehouse), the **WR** (Warehouse-Retailer), or the **MW** (Manufacturer-Warehouse) strategy. In the first option (**W**), the warehouse now owns the relationship with the customer, and its impact on the manufacturer is in many ways similar to that of the **R** strategy. However, a higher revenue can be expected due to the reduced length of the supply chain. In this option, the manufacturer keeps selling a standardized product.

In the second option (**WR**), the warehouse assumes additional value-adding responsibilities such as assembly of all or part of the product. The main difference from the **W** strategy is that the manufacturer can now (via the warehouse) offer a customized product, and can strengthen its market position against competitors that continue to sell standardized goods. In common with the first option, the warehouse still controls the flow of customer information to the manufacturer.

Of greater interest to the manufacturer, however, should be the third option (**MW**). In this strategy the manufacturer is in the driver's seat. It sells its products directly to the customer, but, in part to share revenue risks, and in part to leverage the distribution experience of a warehouse partner, it outsources distribution to a warehouse. Traditional shipping service providers (such as FedEx) have developed additional inventory and merge-in-transit capabilities.

The most evolved of these strategies, however, is to assume all key responsibilities (order processing, inventory management, and production) within the manufacturer. This is labeled as the **M** (Manufacturer) strategy in Figure 5. Note that this option does not necessarily imply that the manufacturer handles the physical product, but refers to the control the manufacturer exerts over the information flow in the supply chain. The fully virtual version of this business model (not discussed here) is also known as Value-Net-Integrator [7].

Figure 6 summarizes the business architectures corresponding to these e-business model alternatives. These are UCM models which only show components, not scenarios: agents for the business models participants, and teams for the roles they play. These models clearly captures how participants are removed from the model or added, and their roles are reassigned between them as the e-business model evolves.

B. Evolving the Business Process

Evolving a UCM model usually involves modifications to the path elements (including responsibilities), the component architecture, and the allocation of path elements to components. However, in order to evolve our business process from a Sell to stock via warehouse and retailer (**R**) strategy to a

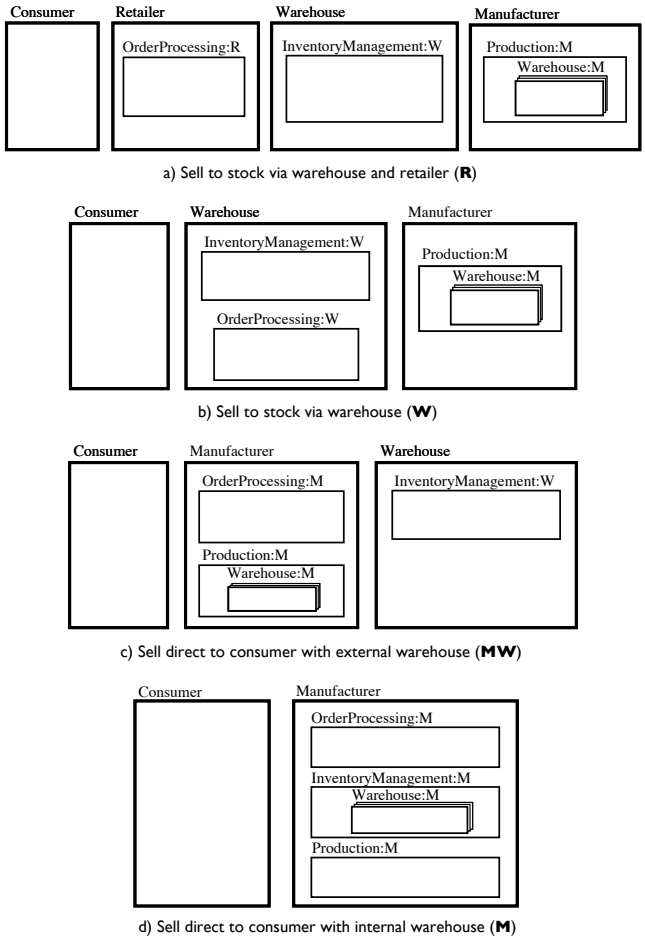

Fig. 6. Alternative architectures for supporting the business process.

Sell direct to consumer with internal warehouse (**M**) strategy, there is little need for modifying the existing UCM paths.

We do not need to modify the allocation of responsibilities to roles, nor do we need to modify the paths themselves. Only the deployment of roles to actors needs to be updated. For example, Figure 7, shows the new root map after eliminating two actors (Retailer and Warehouse), and reallocating their three roles (OrderProcessing and InventoryManagement) to the remaining Manufacturer component. This illustrates a major benefit of the UCM notation: the scenarios are often robust and long-lived, even when the underlying architecture changes. This is usually not the case with message-based scenario notations like MSCs [2] and UML sequence diagrams.

C. Comparing the Business Models

A GRL rationale diagram can be used to compare the business models in terms of their impact on profitability and risk. For the sake of readability, Figure 8 only shows the result of comparing the two extreme strategies, namely **M** and **R**. In a rationale diagram we can represent the preconditions on which each choice is based as soft subgoals of the alternatives.

However, not all companies will be able to evolve their business models as rapidly as they would like to. Figure 8

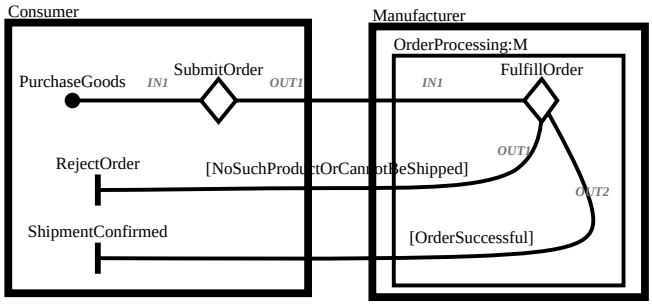

Fig. 7. Strategy M: root Use Case Map.

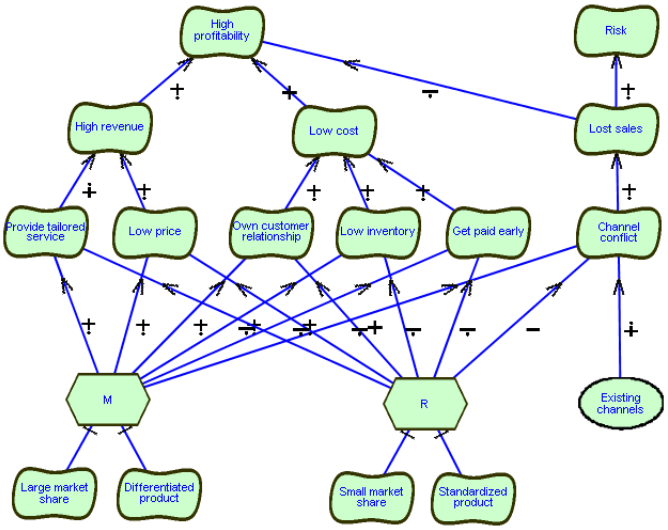

Fig. 8. GRL rationale diagram: Comparison between strategies M and R.

also indicates a key obstacle for evolving quickly from the **R** strategy to the **M** strategy for manufacturers with existing resellers. Trying to remove those resellers from the chain will (initially, at least) lead to a *channel conflict*, and to a loss in sales via those resellers. Thus, the strength of the Existing channels (modeled as a precondition) is a key determinant for how fast the manufacturer can evolve its business model.

IV. Conclusion

This paper introduced how the User Requirements Notation can be used to design and evolve business models. Through a case study, we illustrated the systematic and incremental evolution of (a family of) business model alternatives. GRL models allow the analyst to model the business goals, the specific benefits and liabilities (risks) of each alternative, as well as the dependencies between all participants in the supply chain. We decided to model preconditions as GRL beliefs, which help assess the applicability of business models, and select among different alternative business models.

While GRL models provide rationales, UCM models focus more on the operational aspects of the business process by describing, in abstract terms, who should do what, when, and where. UCMs can integrate multiple scenarios and use cases in a collection of interrelated maps. The notation promotes model evolution by allowing analysts to map responsibilities to components as well as roles to agents/actors in various ways, while minimizing the impact on the rest of the model.

In our case study, we used a standard example (sell through resellers), and developed several alternatives based on the exact same scenario. We discussed the impact of the business models from the perspective of the Manufacturer, who would like to determine when would be a good time to eliminate intermediaries such as Retailers and Warehouses, as well as how this would affect current ways of delivering services.

Due to the space limitations of this paper we could not explore all the issues related to e-business model design and evolution. A detailed description of our approach to business process modeling using URN can be found in [8]. Further information on different aspects of e-business model evolution, and a discussion of related work is available in [9], [10].

In future work we plan to document common e-business models in the form of patterns using URN. We also want to investigate how the evolution of (a family of) business models can be conceptualized as the evolution of a product line. It would also be interesting to explore how more concrete representations of the value flows between business model participants can be integrated into URN models.

Acknowledgment

This work has been supported financially by the Natural Science and Engineering Research Council of Canada, through its Strategic Grants and Discovery Grants programs.

References

[1] D. Amyot, “Introduction to the User Requirements Notation: Learning by Example”, *Computer Networks*, 42(3), 285-301, 21 June 2003. http://www.usecasemaps.org/pub/ComNet03.pdf

[2] ITU-T – International Telecommunications Union, *Recommendation Z.120 (04/04) Message Sequence Chart (MSC)*. Geneva, Switzerland, 2004.

[3] ITU-T – International Telecommunications Union, *Recommendation Z.150 (02/03), User Requirements Notation (URN) – Language Requirements and Framework*. Geneva, Switzerland, 2003.

[4] L. Liu and E. Yu, “Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach”. *Information Systems (Journal)*, Vol.29, No.2, 2003. http://www.cs.toronto.edu/∼liu/publications/

[5] URN Focus Group, *Draft Rec. Z.151 – Goal-oriented Requirement Language (GRL)*. Geneva, Switzerland, Sept. 2003. http://www.UseCaseMaps.org/urn/.

[6] URN Focus Group, *Draft Rec. Z.152 – Use Case Map Notation (UCM)*. Geneva, Switzerland, Sept. 2003. http://www.UseCaseMaps.org/urn/.

[7] P. Weill and M. Vitale, *Place to Space*, Harvard Business School Press, 2001.

[8] M. Weiss and D. Amyot, “Business Process Modeling with URN”. *International Journal on Electronic Business Research*, 63-90, July-September, 2005.

[9] M. Weiss and D. Amyot, “Designing and Evolving Business Models with URN”. *Montreal Conference on eTechnologies (MCeTech)*, 149-162, 2005.

[10] M. Weiss and D. Amyot, “Business Model Design and Evolution”. *International Conference on Management of Technology (IAMOT)*, forthcoming, 2005.

[11] WS-I – Web Services Interoperability Organization, *Supply Chain Management: Use Case Model, Version 1.0*, 2003. http://www.ws-i.org/Documents.aspx

[12] WS-I – Web Services Interoperability Organization, *Supply Chain Management: Sample Application Architecture, Version 1.0.1*, 2003. http://www.ws-i.org/Documents.aspx