From Business Process Modelling to Business Process Automation

Kuldar Taveter[1]

[1] VTT Technical Research Centre of Finland,
VTT Information Technology,
P.O. Box 1201, FIN-02044 VTT, Finland

Abstract. We first describe an agent-oriented business modelling technique that enables to create business process and domain models that are executable by software agents. The technique consists of the steps of analysis by goal-based use cases and design by the extended AOR Modelling Language. We then show how the models created can be represented in XML and executed by agents.

1 Introduction

Because of the increasing need to improve the efficiency of inter-organizational business processes on one hand and the development of Internet on the other, research on Business Process Automation (BPA) is gaining momentum. *Software agents* is an emerging technology which can be utilized for building automated business-to-business solutions. When building such systems, it is important not to view an *agent* as just a technological building block like it is sometimes understood, but also as an important modelling abstraction that can be used at different logical levels in the development of an information system. This enables to create business process models that are executable by software agents. More specifically, we distinguish between *active entities* – ***agents***, which communicate with each other and have beliefs, and *passive entities* – ***objects***, which are intended for representing agents' beliefs, already at the stage of modelling the problem domain that software agents are to be used for.

Conceptually, we add the ***Agent Layer***, comprising inter-agent communication and an agent's features such as capabilities, goals, and commitments, onto the top of the Object Layer. We understand the ***Object Layer*** in a wide sense of the term as either some relational, object-relational, or object-oriented database, Enterprise Resource Planning (ERP) or Enterprise Application Integration (EAI) system, or some object-oriented framework such as COM™ or CORBA™. Agents of the Agent Layer communicate with each other by exchanging high-level typed messages, such as "ASK", "TELL", "REQUEST", and "PROPOSE". Communication between an agent of the Agent Layer and an object of the Object Layer is defined as ***manipulation***. This term has been coined by us to express the fact that objects are submitted to agents.

Taveter K. (2004).
From Business Process Modelling to Business Process Automation.
In *Proceedings of the 1st International Workshop on Computer Supported Activity Coordination*, pages 198-210
DOI: 10.5220/0002679401980210

2 Business Process Modelling

We use for agent-oriented business modelling of the problem domain at hand the Agent Object Relationship Modelling Language (AORML) described in [1] which has been extended by the author of this paper. The extension is described in [2] which also proposes the corresponding methodology for agent-oriented business modelling. The methodology consists of the steps of *analysis* by goal-based use cases and *design* with the extended AORML. By using this methodology, we have modelled the business process of quoting from the perspectives of a buyer and seller as is described below.

2.1 Analysis by Goal-Based Use Cases

Use cases as such were originally introduced by Jacobson in [3]. In [4] and [5], Cockburn proposes an extended version of use cases which he calls "use cases with goals". Goal-based use cases are elaborated in [6].

A goal-based use case consists of a ***primary actor***, the ***system under discussion***, and a ***secondary actor***. Cockburn notices in [4]: "It turns out that the system is itself an actor, and so the communication model needs only work with actors". According to [6], when use cases document an organization's business processes, as in our case, the *system under discussion* is the organization itself or an organization unit. The stakeholders are the company shareholders, customers, vendors, and government regulatory agencies. The primary actors include the company's customers and perhaps their suppliers. Consequently, as 'actor' and 'agent' are synonyms for our purposes, the communication model of goal-based use cases should work in agent-oriented modelling, as well.

A use case may be triggered by an external or internal actor. For example, the use case "Manage quoting" with the buyer in focus is triggered by the buyer's internal actor 'clerk', while the example use case "Process the request for quote" presented in Table 1 is triggered by receiving from a buyer a request for quote. Use case 1 ("Process the request for quote") in Table 1 is modelled from the perspective of a buyer with the seller in focus (*scope*) which means that the goal of the use case is the so-called *user goal*, the goal of the actor (i.e., a buyer) trying to get work (*primary task*) done. The buyer is therefore called the *primary actor* of the use case. The primary task "Process the request for quote" (use case 1) includes as a *subfunction* use case 2 ("Process product item") which is presented in Table 2. A ***subfunction*** is a step in a scenario that is below the main level of interest of the primary actor. The goal of a subfunction, which is a subgoal of some user goal, is therefore assigned to the actor in focus. For example, as Table 2 shows, the goal "to expect bidding of the product item to be decided" of the subfunction "Process product item" has been assigned to the seller. The last use case mentioned includes the main, positive scenario for satisfying its goal and one extension scenario for the case when the product item in the quantity requested is not available.

Table 1. Extended use case for the business process "Process the request for quote".

USE CASE 1	Process the request for quote.	
Goal in Context	The buyer expects to receive from the seller the quote.	
Scope & Level	Seller, primary task.	
Preconditions		
Success End Condition	The buyer has received from the seller the quote.	
Primary Actor Secondary Actors	Buyer.	
Trigger	A request for quote by the buyer.	
DESCRIPTION	**Step**	**Action**
	1	*For each product item requested*: process product item (**Use Case 2**).
	2	*The quote has been approved by the clerk*: send the quote to the buyer.

Table 2. Extended use case for the business process "Process product item".

USE CASE 2	Process product item.	
Goal in Context	The seller expects bidding of the product item to be decided.	
Scope & Level	Seller, subfunction.	
Preconditions		
Success End Condition	The bidding of the product item has been decided.	
Primary Actor Secondary Actors	Buyer.	
Trigger		
DESCRIPTION	**Step**	**Action**
	1	*The product item is available in the quantity requested*: the product item is to be bid which is registered in the quote.
EXTENSIONS	**Step**	**Branching Action**
	1a	*The product item is not available in the quantity requested*: the product item is not to be bid which is registered in the quote.

2.2 Design by Extended AORML

The AOR diagrams were proposed in [1] as an agent-oriented extension of Entity-Relationship (ER)-style or UML-style class diagrams. According to [1], in AORML only *agents* can communicate, perceive, and act. *Objects* do not communicate, cannot perceive anything, and are unable to act. Being entities, agents and objects of the same type share a number of attributes representing their properties or characteristics.

At the design stage of the methodology described in [2], a problem domain at hand is modelled from the informational, organizational, interactional, functional, motivational, and behavioural views. These views are applied in the order shown in Figure 1. The metamodel of the relationships between the views mentioned is represented in [2].

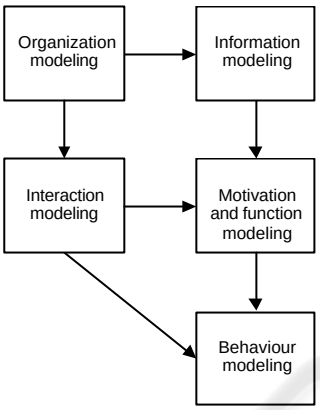

Figure 1. The views of agent-oriented modelling.

2.2.1. Organization and Information Modelling

According to [1], object types, such as sales orders or product items, are visualized in AORML as rectangles essentially in the same way like entity types in ER diagrams, or object classes in UML class diagrams. They may participate in *association*, *generalization*, or *aggregation/composition* relationships with other object types, and in *association* or *aggregation/composition* relationships with agent types. In AOR diagrams, an agent type is visualized as a rectangle with rounded corners. An agent type may be defined as a subclass of another agent type, thus inheriting all of its attributes (and operations). For example, Clerk in Figure 2 is a subclass of Person.

An ***organization*** is viewed under the ***organizational view*** as a complex institutional agent consisting of a number of ***internal agents*** that act on behalf of it and are involved in a number of interactions with ***external agents***. Internal agents may be humans, artificial agents (such as software agents), or institutional agents (such as organization units).

The ***informational view*** of agent-oriented modelling deals with the modelling of *beliefs* of the focus agent about its 'private' objects and shared objects related to it.

The organization and information models depicted in an ***agent diagram*** of Figure 2 represent the agent role types Buyer and Seller with their respective internal agent types. The object types PurchaseOrder, Quote, and Invoice are *shared* between agents of the types Buyer and Seller, while the object instances SellerDatabase and ProductDatabase are represented exclusively within agents of the types Buyer and Seller, respectively. An object of the type QuoteLineItem in Figure 2 satisfies one of the following ***status predicates***: isBid, isNoBid, and isPending, while an object of the type ProductItem represented in Figure 4 is characterized by the ***intensional predicate*** isAvailable(Integer).

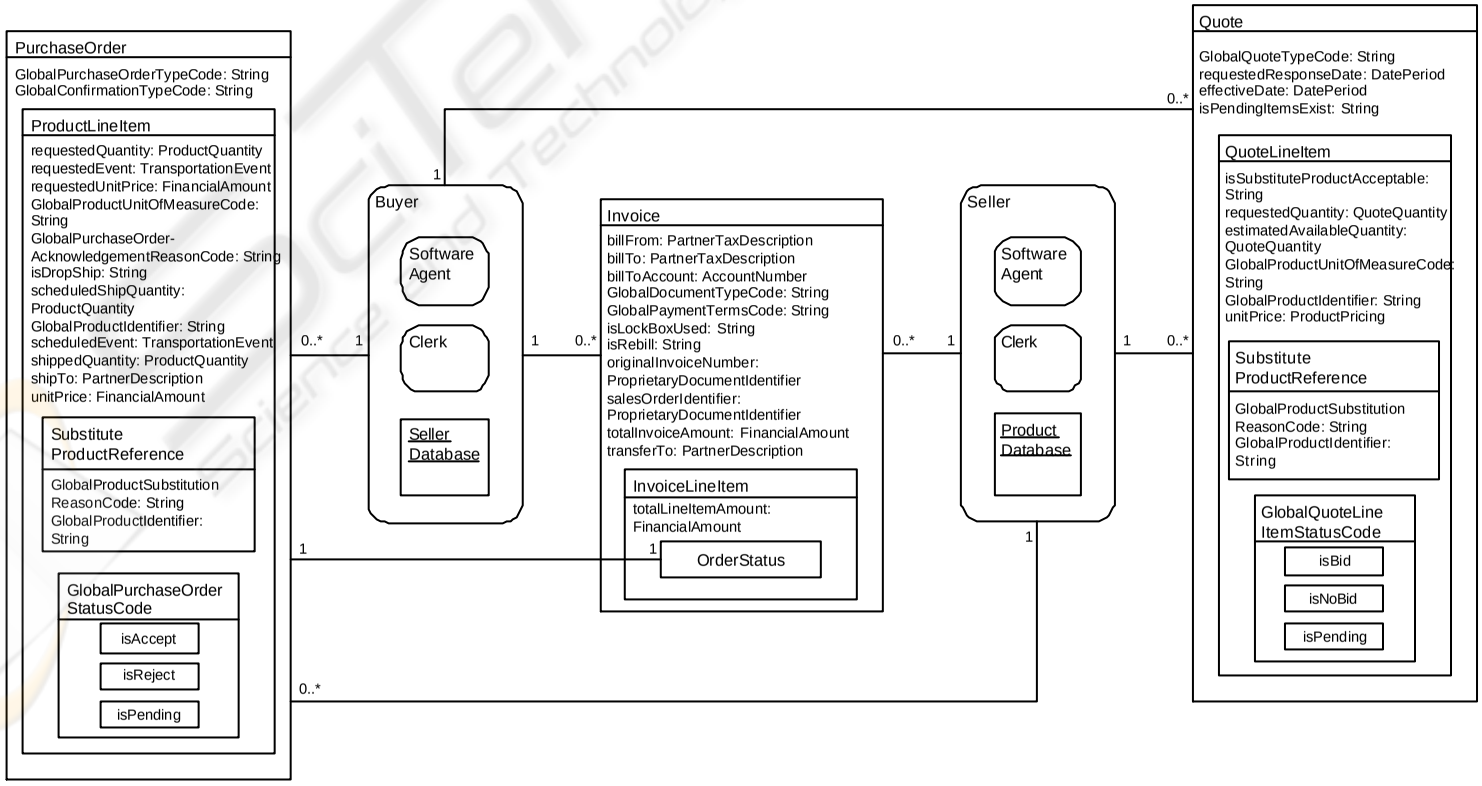

Figure 2. The agent diagram of the domain of business-to-business e-commerce.

2.2.2. Interaction Modelling

According to [1], in a business domain, there are various types of actions performed by agents, and there are various types of state changes, including the progression of time, that occur in the environment of the agents. For an external observer, both actions and environmental state changes constitute events which are addressed by the ***interactional view*** of agent-oriented modelling.

Actions create events, but not all events are created by actions. Those events that are created by actions, such as delivering a product to a customer (e.g., provideProduct in Figure 3), are called ***action events***. Examples of ***non-action events*** are the fall of a particular stock value below a certain threshold, or the arrival of a deadline.

We make a distinction between ***communicative*** and ***non-communicative*** actions and events. The expressions receiving a message and sending a message may be considered to be synonyms of perceiving a communicative event and performing a communicative action.

Business communication may be viewed as ***asynchronous*** point-to-point message passing. As opposed to the low-level (and rather technical) concept of messages in object-oriented programming, AORML assumes the high-level semantics of speech-act-based *Agent Communication Language (ACL)* messages (see [7]).

According to [1], an ***interaction frame diagram*** of AORML provides *a static picture* of the possible interactions between two (types of) agents without modelling any specific process instance. In an interaction frame diagram, an action event type is graphically rendered by a special arrow rectangle where one side is an incoming arrow linked to the agent (or agent type) that performs this type of action, and the other side is an outgoing arrow linked to the agent (or agent type) that perceives this type of event. Communicative action event rectangles have a dot-dashed line. The interaction frame diagram, covering the business process type of quoting between the agent types Buyer and Seller, is depicted in Figure 3.

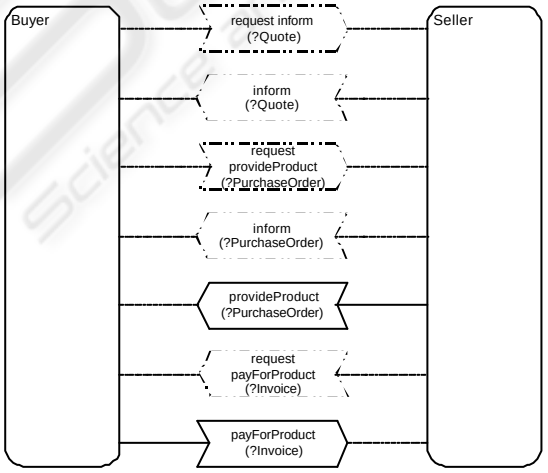

Figure 3. The interaction frame between Buyer and Seller.

2.2.3. Function, Motivation, and Behaviour Modelling

The ***functional view*** of agent-oriented modelling deals with the modelling of *activities* performed by agents. In order to enable function modelling, we have in [2] extended AORML by activities. While an action happens at a time point (i.e., it is immediate), an ***activity*** is being performed during a time interval (i.e., it has duration), and consists of a set of actions, performed by a particular agent. An ***activity type*** (*task* in [10]), like "Process quote request" in Figure 4, is defined as a prototypical job function in an organization which specifies a particular way of doing something [10].

The functional view is closely related to the ***motivational view*** which deals with the modelling of the *goals* agents are trying to achieve, because goals are attached to activities. We express goals by means of the Object Constraint Language (OCL) included by UML [11].

While the functional view of agent-oriented modelling addresses the modelling of business functionality (what has to be done), the ***behavioural view*** addresses the modelling of *business behaviour* (in which order work has to be done). It also deals with the decomposition of activities into actions. Modelling of an agent's behaviour is based on the semantic framework of Knowledge-Perception-Memory-Commitment (KPMC) agents introduced in [8, 9] and extended in [2]. The behaviour of a KPMC agent is encoded by a set of *reaction rules*. A ***reaction rule*** is defined as a quadruple $e, C \rightarrow a, F$ where e denotes the triggering event term, C denotes the precondition formula, a denotes the resulting action term, and F denotes the mental effect formula. An action term specifies the performance of a communicative or non-communicative action, while a mental effect formula specifies a change in the agent's beliefs. Reaction rules are graphically represented as is shown in the legend for Figure 4.

For graphical function and behaviour modelling, an extension to AOR modelling – ***activity diagrams*** – has been introduced in [2]. In activity diagrams, activity types are visualized as rectangles with rounded left and right sides, as is shown in Figure 4. An activity can be started by a reaction rule as is shown in Figure 4 where an activity of the type "Process quote request" is started in reaction to perceiving an action event of the type request inform (?Quote), presenting a request for quote. When an activity of the outermost level is started, to its input parameters are assigned the values of the data items within the triggering event instance. For example, to the input parameter q of the type Quote of an activity of the type "Process quote request" is assigned the value of the data item ?Quote within the triggering event.

Through an ***activity border event*** of the type START(ActivityType), an activity can trigger its subactivity or an internal reaction rule. The triggering event type START(ActivityType) is graphically represented by an empty circle with the outgoing arrow to the symbol for a subactivity type or an internal reaction rule. In Figure 4, upon the start of an activity of the type "Process quote request", its subactivity of the type "Process line items" is started. Additionally, each activity is associated with another implicit activity border event of the type END(ActivityType) that can trigger a subsequent activity or reaction rule. This event type is visualized by drawing a triggering arrow from the activity type symbol to either the symbol of the next activity type or to the symbol of a reaction rule, as is exemplified by Figure 4.

An activity implicitly passes the values of its input parameters to the activity of the next level started by it. For example, in Figure 4, an activity of the type "Process quote

request" passes the value of its input parameter q of the type Quote to the activity of the type "Process line items" started by it.

When a precondition arrow of a reaction rule originates in an object type for which no instance is determined by the triggering event, then the resulting action is performed for all current instances of that object type. A selection condition on the set of all instances may be defined by attaching to the status condition arrow within brackets a suitable expression in OCL For example, the OCL expression {quote = q} attached to reaction rule R2 in Figure 4 means that the action prescribed by the rule (starting an activity of the type "Process line item") is repeated for each QuoteLineItem included by the given instance of Quote which is represented as the input parameter q.

The subactivity "Process line item" in Figure 4 checks the availability of the given product item that is specified by the input parameter item of the type QuoteLineItem. If the product item is available in the quantity requested, the status of the QuoteLineItem is set to isBid. In the opposite case, the status of the QuoteLineItem is changed to isNoBid. In both cases, the attributes of the QuoteLineItem are accordingly updated.

A reaction rule may have more than one triggering event: in such a case the rule is triggered only when all specified triggering events have occurred (without any required order). For example, in the activity state "Confirm quote", the Seller informs a Clerk about the instance of Quote and then waits for the confirmation by the Clerk.

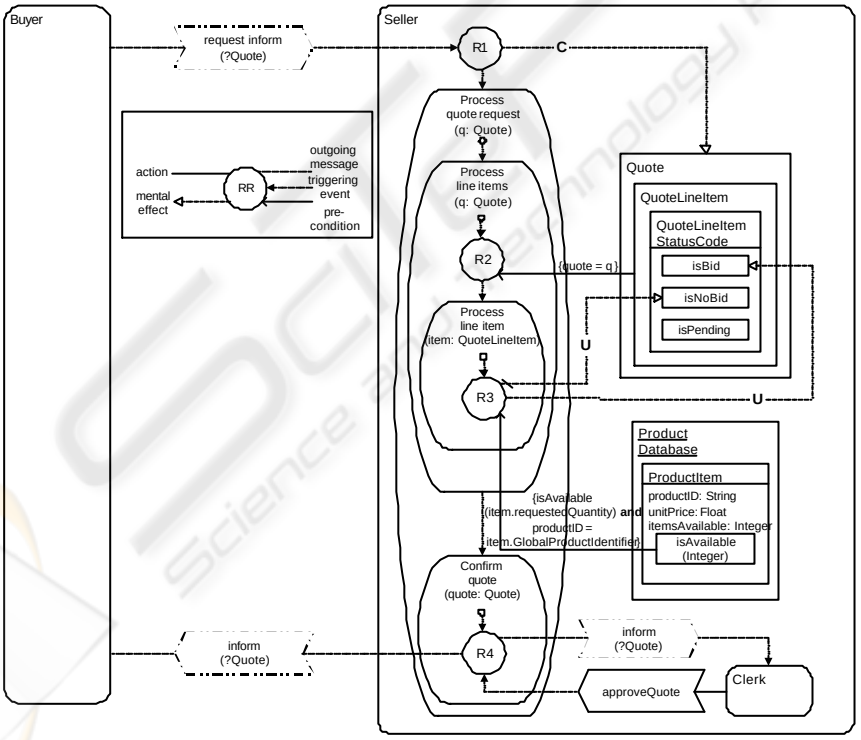

Figure 4. The business process type "Quoting" with the Seller in focus.

3 Business Process Automation by Software Agents

The busines process models developed in section 2 do not constitute a goal in itself but serve as a basis for business process automation by software agents. The principle of automation is very straightforward: *all the functionality described by the behavioural model of a business process type is to be performed by a software agent, with the exception of situations where the intervention by a human agent is absolutely necessary*. For example, in a business process of the type "Quoting" described by Figure 4, a human agent of the type Clerk approves each Quote to be sent to a Buyer, even though in principle software agents of sellers and buyers could handle quoting all by themselves. Intervention by a human agent is needed mainly for administrative or legal reasons, but also in cases where the criteria for selecting e.g. a supplier are not clear-cut. Next, we will briefly describe how agent-based business process automation based on business process modelling can be achieved.

3.1 Generation and Interpretation of Business Process Representations in XML

In our approach, executable models of business process types in the extended AORML are transformed into equivalent XML-based representations in order to enable the execution of the models by software agents, and to grant that all the parties in a business process instance use the descriptions of the same business process type.

To facilitate generation of XML-based representations of business process models, we have developed the corresponding XML Schema [12] whose instances describe business process types in a machine-interpretable way. By using the schema, it is possible to represent business process types from different perspectives. For example, the models of the business process type "Quoting" explained in section 2 are transformed into two XML-based representations that describe the business process type from the perspectives of a Seller and Buyer, respectively. An excerpt from the schema for representing business process types that is shown on the next page reveals that a business process type is represented as a sequence of reaction rules.

Each party in a business process is represented by a software agent. In a preparatory stage, the agent reads the XML-based descriptions of the business process types that the party represented by it is involved in, and creates the corresponding internal executable representations of these business process types. After that, the descriptions of the business process types are ready to be interpreted by the agent. A business process instance is started in response to the corresponding request by a human agent or in reaction to receiving the matching message from a software agent representing some other party. For example, an instance of the quoting business process at the buyer's side is triggered by the buyer's internal human agent of the type Clerk, while the same business process at the seller's side is started by receiving from the buyer's agent a message of the request inform (?Quote) type. A software agent communicates with human agents (e.g., clerks) via a graphical user interface, while the messages exchanged between software agents are represented in the FIPA Agent Communication Language [7].

```
<xs:element name="businessProcess" type="businessProcessType"/>
<xs:complexType name="businessProcessType">
    <xs:sequence>
        <xs:element name="perspective" type="nameType"/>
        <xs:sequence maxOccurs="unbounded">
            <xs:element name="reactionRule" type="reactionRuleType"/>
        </xs:sequence>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="reactionRuleType">
    <xs:sequence minOccurs="2" maxOccurs="7">
        <xs:element name="eventPart" type="eventPartType"/>
        <xs:element name="ruleName" type="nameType" minOccurs="0"/>
        <xs:element name="conditionPart" type="conditionPartType"/>
        <xs:element name="mainActionPart" type="actionPartType" minOccurs="0"/>
        <xs:element name="mainEffectPart" type="mentalEffectPartType" minOccurs="0"/>
        <xs:element name="elseActionPart" type="actionPartType" minOccurs="0"/>
        <xs:element name="elseEffectPart" type="mentalEffectPartType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

3.2 Interfaces to the Internal Information Systems of a Company

An agent representing a company needs to interact with the internal information systems, e.g. with ERP- or EAI-systems, of the company. Because of the heterogenity of such systems, their modelling is not as straightforward as that of business processes. In fact, the corresponding techniques and tools do not yet exist. However, the extended AORML enables to describe interfaces to the internal systems at a high level of abstraction. For example, as we saw in section 2.2.3, the interface to the product database of a Seller is represented as the agent's internal object ProductDatabase shown in Figure 4 which includes the internal object type ProductItem. The instances of the latter represent types of product items sold by the company. Each instance of ProductItem is characterized by a number of attributes, like productID, productName, and itemsAvailable. Additionally, at the implementation level each instance has a method with the signature isAvailable (int quantity) : boolean. When this method is invoked by the agent, firstly the description of the corresponding product item is retrieved from the product database by using the following SQL query where `value` contains the value of the attribute GlobalProductIdentifier of the corresponding instance of QuoteLineItem (v. Figure 4):

```
select * from products where PRODUCT_ID = 'value'
```

After the attribute values of the product item retrieved have been copied into the respective attributes of the instance of ProductItem, the method isAvailable calculates the availability of the product item based on the values of the product item's attribute itemsAvailable and the method's formal parameter quantity, and returns.

In our approach, we wrap the internal information systems of a company into an interface based on Web Services [13]. This means that accessing of either the seller's database of product items or the buyer's database of sellers of product items, as well as of any other internal system of a company, is interpreted as a call of the Web Serv-

ice wrapping the corresponding system. The intermediate layer of Web Services thus provides a clear interface between the Agent Layer and Object Layer which were described in section 1.

3.3 Implementation

For graphical modelling of business process types, Integrated Business Process Editor has been created based on the CONE (Conceptual Network) Software worked out at VTT Information Technology. The Integrated Business Process Editor also enables to transform graphical descriptions of business process types in the extended AORML into their XML-based representations. An agent-based prototype system consisting of the software agents representing the Seller and Buyer has been implemented by using the FIPA-based JADE [14] agent platform. Both agents consist of the Business Process Interpreter (BPI), which is able to read and interpret XML-based representations of business processes, JADE agent, and graphical user interface (GUI). The BPI first reads the descriptions of business process types as requested by a human user. After that, the BPI is invoked to act accordingly by either the JADE agent, when it has received a message, or by the GUI. When the BPI acts, it, in turn, invokes the JADE agent and the GUI.

4 Related Work

Lately, a variety of XML-based techniques and notations for creating executable business process specifications based on Web Services (WS) [13], such as BPEL4WS [15] and BPML [16], have emerged. With respect to these, the difference between an approach based on WS and an agent-oriented approach like ours should first be clarified. Even though Web Services Description Language (WSDL) [13] allows to represent message sequences consisting of e.g. receiving a message and replying to it, this is far from the dynamics of agent communication protocols like FIPA Contract Net [7]. Secondly, in principle an interface to a software agent can be embedded in a WS-based interface, event though its feasibility still needs to be studied.

According to [15], BPEL4WS allows specifying business processes and how they relate to WS which are described by e.g. WSDL. This includes specifying how a business process makes use of WS to achieve its goal, as well as specifying WS that are provided by a business process. A BPEL business process interoperates with the WS of its partners. Finally, BPEL supports the specification of business protocols between partners and views on complex internal business processes.

As it is described in [16], BPML provides an abstracted execution model for collaborative and transactional business processes, and considers e-Business processes as made of a common public interface and as many private implementations as process participants. BPML represents business processes as the interleaving of control flow, data flow, and event flow, while adding orthogonal design capabilities for business rules, security roles, and transaction contexts. BPML is accompanied by a BPMN – Business Process Modeling Notation.

In [17], the types of workflow patterns supported is suggested as an objective measure for comparing WS composition languages. The comparison between a number of standard proposals for busines process modelling on one hand and the extended AORML on the other hand that we have carried out in [2], based on the set of workflow patterns defined in [18], reveals that the extended AORML supports all workflow patterns except "Synchronizing Merge", "Interleaved Parallel Routing", and "Milestone". The extended AORML thus provides a better support for workflow patterns than any other busines process modelling standard compared, including BPEL4WS, BPML, and UML Activity Diagrams [11].

Topic maps described in [20] consist of addressable information objects around topics ('occurrences') and relationships between topics ('associations'). Topic maps enable multiple, concurrent views of sets of information objects. As such, topic maps seem to be suitable for XML-based representation of the informational view (i.e., *ontology*) which has so far not been addressed by the research initiative described.

5 Conclusions and Future Work

We have proposed a modelling technique and notation that enable to generate from business domain and process models described at a high level of abstraction their representations in XML that are executable by software agents. This raises the perspective of business process automation to a new level where, after installing the agent-based system, the user has to perform just the following steps in order to automate his/her business processes:

1. Find and retrieve from some authorized body, like e.g. RosettaNet [19], the user perspective models of the types of business processes the user is willing to be involved in.
2. Request the software agent to read the models of the business process types.
3. Start a business process instance by using a graphical user interface of the software agent, or have the software agent to wait for the message triggering a business process of the type "understood" by it.

The methodology presented in this paper is just the first step towards a more flexible approach/system with a more loose integration between models of business process types and actual business process instances carried out between agents. This would enable a software agent, which is possibly assisted by a human, to select at each step of a business process from many alternatives the most appropriate actions to be performed. Some of the other new questions and research challenges that arise from our approach are:

1. How can we incorporate into our approach the modelling of goal-oriented *proactive behaviour* based on planning and plan execution?
2. How can we handle more systematically *exceptions* (now they are handled mainly by the ELSE-constructs of reaction rules) to standard processes (for instance, when a response to a quote has not arrived in due time)?
3. How can we more precisely model *interfaces* to human agents (now they are modelled by using non-communicative action event types)?

Acknowledgements

The author would like to thank the Technology Agency of Finland, the companies participating in the Plug-and-Trade project, and VTT Information Technology for funding the project. The author also owes thanks to the present and former members of the Plug-and-Trade project team. The research work that has led to this project has been partly funded by the Estonian Science Foundation's grant number 4721.

References

1. Wagner, G. The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. Technical report, Eindhoven Univ. of Technology, Faculty of Technology Management, May 2001. *Information Systems* 28:5 (2003).
2. Taveter, K. A *Multi-Perspective Methodology for Agent-Oriented Business Modelling and Simulation*. Ph.D. thesis, Tallinn Technical University, 2004 (in press).
3. Jacobson, I., et al. *Object-Oriented Software Engineering: A Use-Case Driven Approach*. Addison-Wesley, Reading, MA, 1992.
4. Cockburn, A. Goals and Use Cases. *Journal of Object-Oriented Programming*, September 1997.
5. Cockburn, A. Using Goal-based Use Cases. *Journal of Object-Oriented Programming*, November/December 1997.
6. Cockburn, A. *Writing Effective Use Cases*. Addison-Wesley, 2001.
7. *Foundation for Intelligent Physical Agents* (FIPA). http://www.fipa.org
8. Wagner, G. Vivid Agents - How they Deliberate, How They React, How They Are Verified. Extended version of: Wagner, G. A Logical And Operational Model of Scalable Knowledge- and Perception-Based Agents. Van de Velde, W., Perram. J. W. (Eds.), *Agents Breaking Away, Proceedings of MAAMAW'96*, Springer Lecture Notes in Artificial Intelligence 1038, 1996.
9. Wagner, G. *Foundations of Knowledge Systems with Applications to Databases and Agents*, volume 13 of Advances in Database Systems. Kluwer Academic Publishers, 1998.
10. Yu, E. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, Department of Computer Science, University of Toronto, 1995.
11. *OMG Unified Modeling Language Specification*, Version 1.5, March 2003. Available at http://www.omg.org/uml/
12. *XML Schema 1.0*. http://www.w3.org/XML/Schema.
13. *Web Services Activity of W3C*. http://www.w3.org/2002/ws/
14. *JAVA Agent Development Environment (JADE)*. http://jade.cselt.it/
15. *Business Process Execution Language for Web Services version 1.1*. Specification available at http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
16. *Business Process Modeling Language 1.0 and Business Process Modeling Notation 0.9*. Specifications available at http://www.bpmi.org/
17. van der Aalst, W.M.P. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, Jan/Feb 2003.
18. *Workflow Patterns*. http://www.tm.tue.nl/it/research/patterns
19. *Rosetta Net*. http://www.rosettanet.org
20. XML Topic Maps (XTM) 1.0. http://www.topicmaps.org/xtm/1.0/