MODELING OF E-BUSINESS BROKERAGE SYSTEMS USING UML AND PETRI NET

Boris Shishkov and Joseph Barjis
Delft University of Technology, Faculty of Information Technology & Systems; The Netherlands;
e-mails: Shishkov@IS.TWI.TUDelft.nl, J.Barjis@its.tudelft.nl

Abstract: The modeling of software applications that realize brokerage functionality is of particular importance for e-business. UML appears to be a suitable tool for conducting such a modeling. The UML is widely accepted now as a standard modeling environment for modern application software development. It is popular among researchers and well proven in practice. UML offers also an appropriate "interface" between requirements engineering and software design, namely the concept of Use case. However, it is still a question how to properly derive use cases in such a way that the software design model stems from a model of the processes to be supported by the software. The idea that is proposed and elaborated is to derive Use cases from a Petri net business process model. Petri net is chosen because of its ability to completely capture a system (including its dynamics) and because a Petri net model is straightforwardly mappable into a Use case model. A development process is suggested about the combination of UML and Petri net, in realizing design (which stems from business process modeling) of e-business brokerage system. The development process is illustrated using a case example.

Key words: Modeling; UML; Petri net; e-Business

1. INTRODUCTION

In the past few years there have been major developments in the field of Information and Communication Technology (ICT). The most significant examples of this are the digital multimedia and the great advances in global telecommunications. Business processes (BP) also underwent development –

e.g. re-engineering and distribution. By creating new possibilities for the business area, this great progress increases dramatically the demands towards business. These increasing demands could be met if business activities are put on advanced technological foundation (BETADE, 2002). Although this is still not the case in many of the contemporary business domains, there are already some promising examples in this direction, e.g. e-business (EB), which have inspired us for the current research activities.

According to Shishkov & Dietz (BETADE, 2002), EB is defined as: *business conducted using to a large extent the possibilities of ICT, including Internet. EB encompasses such diverse activities as: identifying relevant partners, negotiating with them, and conducting business transactions.* This definition is adapted from contemporary definitions concerning EB, e.g. the definitions in (Krcmar et al, 1995; MALL, http://www.we-trade.org; MEMO, http://abnamro.com/memo). Actually, the content of the definition of EB includes elements of some definitions of e-commerce, e.g. in (Linkvest, http://www.linkvest.com). The concept of EB is much broader than the concept of e-commerce and is a good example of basing completely a business cycle on ICT. However, a number of problems appear which need to be solved in order to successfully develop EB. Such a problem, valid not only for EB but also for most of the contemporary businesses which rely on Internet and thus face the consequences of globalization, is the following: these businesses do not succeed to perform effective partner and/or goods searching, which could make them much more successful. Evidence of the existence of this problem is the fact that currently a number of projects, e.g. (MALL, http://www.we-trade.org; MEMO, http://abnamro.com/memo) concentrate on exploration of brokerage systems, e.g. for EB.

The goal of our paper is to address the outlined problem by considering the design of EB brokerage system. In particular, we deepen towards relating the design to prior modeling of the BP to be supported by the system.

The Unified Modeling Language (UML) appears to be a suitable tool for conducting such a design. The UML is widely accepted as a standard modeling environment for modern application software development. It is popular among researchers and well proven in practice. UML offers also an appropriate "interface" between requirements engineering and software design, namely the concept of Use case. However, it is still a question how to properly derive Use cases in such a way that the software design model stems from a model of the processes to be supported. The idea that is proposed and elaborated is to derive Use cases from a Petri net BP model. Petri net is chosen because of its ability to completely capture a system (including its dynamics) and because a Petri net model is straightforwardly mappable into a Use case model. A development process is suggested about the combination of UML and Petri net, in realizing design (which stems

from BP modeling) of EB brokerage system. The suggested process is illustrated using a case example.

The outline of the paper is as follows. Section 2 analyzes some basic concepts regarding EB in general and EB brokerage systems in particular. Sections 3 and 4 summarize the up-to-date knowledge on UML (with stress on the Use case and Activity diagrams) and Petri net, respectively, and are intended for readers who are not familiar with these tools. The core of the paper consists of sub-sections 5.1 and 5.2 of Section 5. In sub-section 5.1 we introduce a development process (based on UML and Petri nets) for EB brokerage systems design. In sub-section 5.2 we illustrate the process with a case example. Section 6 contains the conclusions.

2. BASIC CONCEPTS ABOUT EB AND BROKERAGE SYSTEMS SUPPORTING EB

Considering the definition of EB, provided in the introduction, it might be concluded that the EB process includes, as shown on Fig. 1: a) provision of information (partners, relevant goods and/or services should be found); b) negotiation and contracting (these activities should take place independently of time and space); c) conducting the negotiated transaction(s). In (MEMO, http://abnamro.com/memo), a), b), and c) are called Information, Negotiation and Fulfilment phases, respectively.

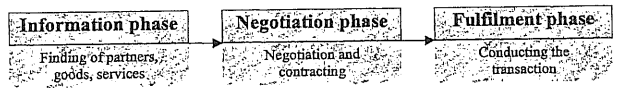

Figure 1: The EB Process

The addressed problem, which was outlined, implies that the focus should be put on the Information phase. The problem formulation is close to the broader formulation in (Linkvest, http://www.linkvest.com): "many companies (including EB companies) are still poorly positioned to take advantage of the proliferation of globally disseminated information sources". Brokerage systems, knowledge-based digital libraries, and portals are among the promising directions of contemporary research activities, which could contribute to the creation of better searching and match-making mechanisms.

As already stated, we consider worthwhile investigating the modeling of brokerage systems because this could improve (particularly in EB) the way in which the actors involved (in EB) find effectively proper partners and/or goods. Since two UML diagrams are considered, vis. the Use case diagram

and Activity diagram, as well as Petri net (reasons for these choices are explained later on), the particular focus of the paper is: modeling of a brokerage system supporting EB, based on UML and Petri net.

With respect to these issues, it is necessary to discuss briefly such brokerage systems. The general requirements towards the functionality of a brokerage system supporting EB are well known from many literature and other sources treating EB, e.g. (Linkvest, http://www.linkvest.com; MALL, http://www.we-trade.org; MEMO, http://abnamro.com/memo; O'Sullivan et al, 1998). Analogous functionality is investigated in (Shishkov & Barjis, 2000). It is required for such a system to support users, spread across multiple locations. Thus, it should represent a distributed ICT application which effectively supports its users across distributed computing environments. It is required also that such a brokerage system performs match-making, as shown on Fig. 2.

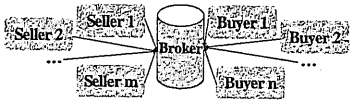

Figure 2: General view of the functionality of a brokerage system

As seen from the figure, there are: 1) different sellers aiming at succeeding to sell their goods as quickly as possible; 2) different buyers aiming at purchasing specific goods they are interested in. The system is supposed 1) to let seller *i* find the buyer being interested in the goods offered by him; 2) to let buyer *j* find the seller offering the goods he is interested in. Sellers and Buyers could, for example, pay on a subscriptional basis for the realized services. Anyway, behind this not so complex general functionality, there are many issues that should be taken into account when developing such a distributed application: how to store, operate and maintain the data; how the application should provide its services to users, how some non-standard situations should be approached, etc.

Thus, we should pay special attention to the modeling of such an application. It is important to find a consistent combination of tools for the modeling process, which to allow both grasping the system (and its functionality) as a whole and elaborating precisely any parts of it, parts having importance for its operation. It is also essential that the system design stem from the modeling of the BP to be supported. According to Shishkov & Dietz (BETADE, 2002), bridging these two tasks (software design and BP modeling) is crucial for developing effective software. In order to do this, it is needed to combine tools for application modeling with BP modeling tools.

These tools should be: suitable for applying them together, within an integrated task; adequate for the peculiarities of the particular (EB) domain.

The following two sections discuss the modeling tools suggested to be used, elaborating on their advantages (which motivate the choice).

3. REALIZING SYSTEM DESIGN USING USE CASES AND UML

Rumbaugh, Booch and Jacobson state that the UML is meant for "visualizing, specifying, constructing and documenting the artefacts of software intensive systems" (Booch et al, 1999). It can be seen as a selection of models that allow developers to specify a system, to actually have the system built and to document it. The UML offers a wide and rich variety of widely accepted modeling concepts. This makes it actually the standard language for modeling automated information systems. Thus, we suggest to base the software design concerning an EB brokerage system on UML.

3.1 The application modeling task

When considering application modeling, it is necessary to take into account some specificities regarding the particular modeling task, e.g. peculiarities of the domain, required insights, etc. As stated in the introduction, the paper is focusing on EB in general and EB brokerage systems in particular. Since grasping the dynamics in some of the system's activities is important in the modeling of a brokerage system, it might be expected that the modeling of such an ICT application should require not only a consistent view over the entire system, taken as a whole, but also a dynamic perspective with respect to the realized activities with importance for the system's functionality.

With regard to this, some particular modeling tools, related to UML, are considered and discussed further on in this section.

3.2 Use cases, Use case diagram and Activity diagram as suggested modeling tools

Use cases (UC) are considered crucial for the system design addressed in this paper. The UML UC and Activity diagrams play also an important role in the considered modeling of EB brokerage system. These modeling tools are briefly outlined and discussed below.

3.2.1 UC diagram and UC

The UC diagram is an important UML diagram. This diagram has a fundamental role in the UML development process, showing actors and UCs together with their relationships (OMG, 2000). The UCs represent functionality of a system. The diagram itself is a graph of actors, (a set of) UCs, and the relationships between these elements (associations, generalizations, etc.). It might include also some interfaces. By representing the potential UCs for the system to be built and relevant actors, the diagram provides the starting point in modeling systems.

As for UC (Sparks, 2000), exploring them should not be limited to UML, since they have appeared independently of UML. Anyway, Jacobson, who introduced UC (Jacobson, 1992), developed further his ideas about UC in the direction of UML (Booch, 1999). But Cockburn developed his own UC perspective (Cockburn, 2001) which, although complementing the concepts of Jacobson in some respects, offers slight contrasts to it in other issues. Therefore, when considering the modern UC theory, one should take into account not only the fundamental concepts of Jacobson, but also e.g. the UC perspective of Cockburn (which adds elicitation value).

Regarding **Jacobson's conceptual views**, shared also by Fowler (Fowler & Scott, 2000), in a UC "*a user performs a behaviorally related sequence of transactions in a dialogue with the system*". This perspective is illustrated in Fig. 3 A and according to it, a UC is a typical user/computer system interaction. It captures some user-visible function. This view suggests that developers of good UC identify the users' goals, not the system functions. However, the UC concepts of Jacobson are not discussed in more detail since they are well known to the public.

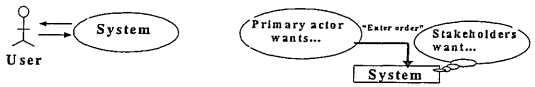

Figure 3: UC perspectives of Jacobson and Cockburn

According to the **concepts of Cockburn** (Cockburn, 2001), a UC describes a system's behavior. In his model (Fig. 3 B), Cockburn defines at the most generic level a *system* and *actors* (both having responsibilities and behavior). He is interested just in these actors, situated outside the system, and from them chooses those whose interests should be protected by the system. Cockburn calls these actors *stakeholders*. The *primary actor* (or *user*) is one of the stakeholders, the one who has the goal and who initiates the system's activity. The system should satisfy this goal and protect at the

same time the interests of the stakeholders. The primary actor's *goal* drives the UC. The primary actor should be determined as well as the goal level - a goal may contain sub- and sub-sub-goals. Besides determining the goals, it is essential to determine exactly what is the system under discussion. This is called *scope*. Functional scope refers to the services that the system offers. Design scope is the set of systems, hardware and software, that the developer is charged with designing and discussing. It is important that the writer and reader are in agreement about the design scope for a UC. Cockburn suggests as fundamental: "*Enterprise*" scope – the UC describes a person's interaction with an organization, "*System*" scope – the UC describes a person's interaction with hardware/software, "*Subsystem*" scope – refers to situations where we describe how a piece of a system works. A UC may include also other elements – action steps, scenarios, preconditions, etc.

Below, the outlined perspectives are **compared** considering the purposes behind them.

Aiming at looking inside UC, Cockburn has built a concept, allowing developers to keep interest not only in the user of the system (the primary actor) but also in the other stakeholders. For this reason, the model of Cockburn seems more complete than the model of Jacobson. At the same time, the graphical representation of this model is unsuitable for presenting a multitude of UC.

The latest developments in Jacobson's UC concept are put in the perspective of UML. For this reason the concept emphasizes not on the complete representation of a UC, but on features allowing developers to show relationships which cover many UCs and actors. This is the main function of the UC diagram. Thus, Jacobson extracts the gist – actors, pieces of functionality and their relationships.

Jacobson and Cockburn form their UC perspectives from different angles. This shows that UC needs further exploration in order to provide options for both complete insight and flexible multi-UC representation. The ideas in the current paper are also in this direction, suggesting application of UC in combination with Activity diagram, for an extended insight.

3.2.2 Activity diagram (AD)

An AD is a special case of a state diagram in which all (or at least most) of the states are action or sub activity states and in which all (or at least most) of the transitions are triggered by completion of the actions or sub activities in the source states. The entire AD is attached (through the model) to a class or to the implementation of an operation, or a UC. The purpose of this diagram is to focus on flows driven by internal processing, as opposed to external events (OMG, 2000). AD is based on ideas from different

techniques, e.g. state modeling techniques. Unlike the other UML diagrams, AD is not founded on object-oriented ideas. It is particularly useful in describing behavior, providing a possibility to capture some dynamical aspects of behavior.

One of the strengths of AD lies in the fact that it supports not only sequential but also parallel behavior. This makes it a great tool for workflow modeling (Fowler & Scott, 2000). Another strength of AD is that it could be used as a starting point for conducting computer simulation. Models represented with the means of AD could be easily simulated, using different tools (Barjis & Shishkov, 2001).

Taking into account the advantages of the modeling tools discussed in this section, the belonging of the UC diagram and AD to the standard modeling language – UML, as well as the suitability for applying these tools on the basis of a Petri net BP model (this will be discussed in the following sections), the choice is made to base the modeling of EB brokerage system on the UC diagram, UC and AD.

4. BUSINESS SYSTEM MODELING USING PN

Any business can be viewed as a collection of processes, where a process can be described as "a set of identifiable, repeatable actions, which are some way ordered and contribute to the fulfilment of an objective". These processes change as organizations evolve over time in response to their business environments. The focus on the BP is important in order to design, maintain and improve the way businesses work effectively and efficiently.

So, as it becomes obvious, in the modeling of software systems (e.g. supporting EB), there are two aspects to be studied. First, it is essential to model consistently the system itself, eliciting precisely all the necessary static and dynamic issues as well as to reflect the requirements in the designed functionality. Thus, all this needs to be based on a sound BP model. Second, it is necessary to have such a BP model, to have a sound graphical language or tool to build models of the business system, to capture the system dynamics, to represent processes in time sequence. As for the system elicitation and modeling tool, we consider UML suitable for this purpose. As for the graphical representation and capturing the dynamics of the BP to be supported by the software system, we use Petri net (PN). PNs have well supported tools to allow modeling, analysis, and, if necessary, simulation (execution) of systems (including EB systems). PN is a well known and widely used modeling technique (Aalst, 1998; Agostini & Michelis, 1998; Barjis & Dietz, 2001). The PNs are formalism and a graphical language for the design, specification, and verification of systems.

In order to better understand PNs, there are some typical examples of PN depicted in Fig. 4. These examples are especially chosen to demonstrate PNs application and way of modeling, while dealing with processes in series, parallel, and conditional or alternative processes. It should be noted that rectangles coloured in grey indicate that these transitions are enabled.

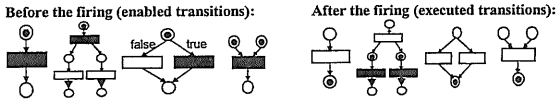

Figure 4: Typical PN examples

In Fig. 4, there are the following situations represented before and after the firing of transitions from the left to the right. The first example shows an ordinary process having one transition. The second example represents parallel processes. When the enabled transition fires, it enables two other transitions in parallel. The third example represents alternative processes. This example shows that only one of the two processes, for which the condition is true, will be undertaken. The fourth and last example represents synchronization or AND-join processes. In this situation it is necessary to finish both processes completely before starting the following one.

In order to relate a PN model to UML, a development process is suggested. In order to illustrate the suggested process and its practical implementation, an example of real life system is considered. All this is focused in Section 5.

5. COMBINED APPLICATION OF UML AND PN

Aiming at contributing to the knowledge on software design based on BP study, and in particular – in the perspective of EB brokerage activities, we introduce below a development process, suitable for the modeling of EB brokerage system. Further on, we illustrate the process using a case example.

5.1 Development process based on UML and PN

As stated in the introduction, we claim that the combination of PN and UML is useful for the modeling of software systems supporting BP (in general) and of EB brokerage systems (in particular). The suggested development process allows developers to conduct software design based on a PN process model.

The development process consists of the following steps:

- Delimitation of the domain under study.
- Applying PN to explore the BP to be supported.
- System modeling, including the development of a UC diagram, concerning the system being designed, based on the BP PN model.
- Elaboration, including:

  * Partial representation of the system as a subsystem (based on the UC concept of Cockburn (Cockburn, 2001)) allowing developers to look inside any particular UC(s), taking into account not only the (primary) actor, but also the stakeholders involved. The scenario development, suggested in the mentioned concept, could be valuable for getting an extended insight.

  * Further granularity of the subsystem(s) with respect to the realized activities, using AD.

- Proceeding with computer simulation, if necessary, to validate the developed models.

The HBS case example illustrates the suggested development process.

5.2 The HBS case

The considered case study concerns one particular example of an EB related brokerage system - Holiday Brokerage System (HBS). It matches the data about clients' required holidays and offers of tour-operators. In this case, if we consider the general view over the functionality of a brokerage system (Fig. 2), clients should take the role of "buyers" and tour-operators should take the role of "sellers". Both the tour-operators and clients need to register in order to use the service for a selected period of time. The subscription fees for the tour-operators are fixed depending on the chosen period and the content of their offers; the fees are fixed for clients also, depending on the chosen period. Besides these subscription fees, both clients and tour-operators pay fixed fees when a match-making is realized. HBS accepts holiday requirements from the clients (e.g. check in/out dates, place, type of accommodation, price, etc) and information from the tour-operators (e.g. number and type of accommodation, facilities, transport included, etc.). Once HBS has received requirements from a client, if requested, it performs match-making on a real time basis. HBS provides the client with a list of available holiday offers (which satisfy client's requirements) to select from. Once the client has accepted one of the offers, he as well as the corresponding tour-operator pay a fixed fee, reservation is conducted and the service is considered finished. Below, we demonstrate the modeling of such a system following the suggested development process.

First. After delimitation of the domain, PN should be applied to explore the BP to be supported by HBS.

In order to design a proper PN model of the HBS, one needs to firstly identify processes. From the process modeling point of view, there are four processes involved in the entire circle. These four processes are: 1) Subscription/renovation – this process also includes payment for the HBS service; 2) Requirements collection – this process can be either accomplished electronically, by phone, or other communication means; 3) Searching – this process is accomplished by HBS and the result is provided to the client; 4) Offer acceptance – a final process that leads to the offer of (a package of) services by the HBS and its acceptance by the client.

To represent these processes in terms of PN, the following model is designed (Fig. 5). The model consists of transitions and places. Transitions are represented as rectangles while places are represented as circles. Rectangles coloured in grey represent main processes in the HBS business.

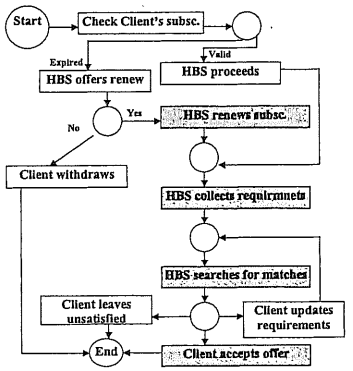

Figure 5: PN model of the HBS system

Second. Based on the PN business model, it is continued with the development of a UC diagram. The diagram (Fig. 6) shows UCs and actors typical for such a brokerage system. Since the purpose is just illustrative, only some of the UCs and actors typical for such a system are considered.

Regarding the diagram, there is one actor represented on it: *Client*. Concerning Client – he takes the decision, he has the responsibility, he has the goal to initiate or renew a subscription, have his information matched up with relevant data from a tour-operator, to accept any desired offer(s) and make reservation for it.

The diagram contains six UCs: "Initiate/renew Subscription", "Request Additional Information", etc. The UC "Initiate/renew Subscription" is highlighted since it will undergo the further investigation steps.

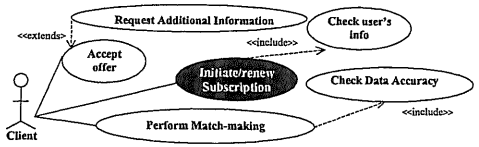

Figure 6: UC diagram of HBS

It is seen that the basic UCs are straightforwardly derived from the PN model. For example, the UC "Accept offer" (Fig. 6) directly reflects the process/transition "Client accepts offer" (Fig. 5). Hence the UC model which founds the software design activities, stems from a BP investigation model.

Regarding the considered UC diagram, besides the outlined UCs and actor, there are two <<include>> relationships ("Perform Match-making" requires "Check Data Accuracy"; "Initiate/renew Subscription" requires "Check user's information") and one <<extends>> relationship (in some cases, before accepting an offer, the client might request some additional information, so the basic UC is "Accept offer" and it is extended with "Request Additional Information").

Third. Based on the UC diagram, it is continued with further investigation of any particular UC(s) of interest, based on the concept of Cockburn. We have selected, for illustrative purpose, the highlighted UC "Initiate/renew Subscription"; the mentioned investigation is applied to it.

UC: Initiate/renew Subscription (for a better illustration, we have simplified the example by considering only subscription initiation). **Primary Actor**: Client. **Goal in Context**: Client's subscription is activated. **Scope**: System (because the UC describes a person's interaction with a computer system). **Stakeholders and Interests**: Client – wants to be correctly subscribed for the service of HBS; the owner of HBS – wants to be compensated for running HBS; the public – wants to be sure that the data provided by HBS is correct. **Precondition**: none. **Minimal guarantee**: Client is in a position to provide correct data and pay for the service. **Trigger**: Client decides to subscribe for the service of HBS. **Main success scenario**: 1. **Client**: decides to subscribe (and initiates contact with HBS). 2. **HBS**: provides initial information and requires identification (ID) data and credit card number (nr). 3. **Client**: provides ID data and credit card nr. 4. **HBS**: initiates credit card authorization procedure and lets Client log on. 5. **Client**: enters HBS and submits a subscription form. 6. **HBS**: checks the data provided and asks for Client's confirmation. 7. **Client**: confirms his will to subscribe. 8. **HBS**: inserts Client's data into the database and charges his credit card with the fee for the selected period. 9. **Client**: logs out. Scenario's END reached. **Extensions** (only those related to activity 6 are

depicted): **6a**. The data from the subscription form submitted by Client is not complete. => HBS asks Client to submit again the form and provide complete data, indicating what is incomplete in the submitted form. Go: 5. **6b**. The data from the form submitted by Client is irrelevant with respect to HBS's scope. => HBS informs Client that the provided data is inadequate to its service and cancels the authorization procedure. Go: END.

Fourth. Based on the UC investigation, it is continued with the construction of an AD model for the chosen UC. As seen from the main success scenario, there are nine core activities (plus extensions) in the UC "Initiate Subscription". Some of them are shown on Fig. 7 as an overall AD.

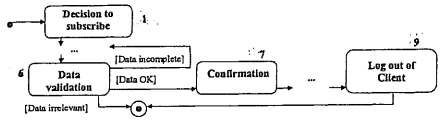

Figure 7: AD model for the UC: "Initiate Subscription"

Fifth. And finally, from the AD model it is straightforward to proceed with computer simulation. However, this is left beyond the scope of this paper. The usage of an AD model as a pre-simulation model, prior to realizing simulation (e.g. using ARENA), is studied and illustrated with an example in (Barjis & Shishkov, 2001).

6. CONCLUSION

The paper's goal, as stated in the introduction, is to contribute to the knowledge on software design in general and e-business brokerage systems design in particular, by aligning software design and BP modeling. UML and PN are chosen as proper tools for this purpose because:

1) UML is a well proven and probably the most popular tool for software design, which is already de facto the standard software modeling language.

2) PN represents (as shown in the paper) a complete and powerful tool for BP modeling, and also – it is straightforwardly relatable to UML.

A fundamental problem regarding UML-based information systems (or ICT-applications) design is the identification problem: how to find all relevant Use cases (Mallens et al, 2001)? This problem appears to be fully solved (regarding the modeling of e-business brokerage systems) by the means of the suggested development process which allows developers to derive Use cased from a PN BP model. This was illustrated using a case example (the HBS case).

It was shown also that the Use case diagram is a helpful starting point for system modeling, providing elicitation in relation to identification of processes and requirements specification. It is of particular benefit that the diagram consists of a number of Use cases, which allows analysts to choose any desired Use case(s) for further study. After carrying out a complete analysis of a chosen Use case (according to the model of Cockburn, where action steps are described within a scenario, supported by a set of extensions), it is straightforward to build an Activity diagram model, which helps to represent and visualize the action steps within a Use case in sound graphical notations.

Concerning PN, it was demonstrated that this tool is easily readable, simple and adapted to capture the dynamics of systems and processes. Another advantage of this type of PN is its place transition structure. There are many available tools supporting PN models of place transition type, e.g. simulation tools (models based on PN can be easily simulated, studied, and analyzed using any discrete event simulation tools).

The suggested development process is expected to be a helpful contribution to the development of e-business.

7. REFERENCES

Aalst, W.M.P. van der. Finding Errors in the Design of a Workflow Process: A Petri-net-based Approach. Proc: 19th Int. Conf. on Applications and Theory of Petri Nets – WFM, 1998, Lisbon, Portugal.

Agostini, A.; Michelis, G. De. (1998). Simple Workflow Models. Proc: 19th Int. Conf. on Applications and Theory of Petri Nets – WFM, 1998, Lisbon, Portugal..

Barjis, J.; Dietz, J.L.G. A Type of Petri Net Based on Speech Act Theory for Modeling Social Systems. Proc: 4th Int. EUROSIM Congress, 2001, Delft, The Netherlands.

Barjis, J., Shishkov, B. UML based Business Systems Modeling and Simulation. Proc: 4th Int. EUROSIM Congress, 2001, Delft, The Netherlands.

BETADE project, *Building blocks for Effective Telematics Application Development and Evaluation*. TU Delft Press, 2002.

Booch, G., Rumbaugh, J., Jacobson, I., *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley Longman, 1999.

Cockburn, A., *Writing Effective Use Cases*. Addison-Wesley, 2001.

Fowler, M., Scott, K., *UML Distilled, Second Edition – a Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, 2000.

Jacobson, I., Christenson, M., Jonsson, P., Overgaard, G., *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.

Kremar, H., Bjorn-Andersen, N., O'Callaghan, R., *EDI in Europe*. John Wiley & Sons Ltd., 1995.

Linkvest, corporation web site: http://www.linkvest.com.

MALL 2000, project web site: http://www.we-trade.org.

Mallens, P., J.L.G. Dietz, B-J Hommes. The value of Business Process Modeling with DEMO prior to Information Systems Modeling with UML. Proc: EMMSAD, 2001, Interlaken, Switzerland.

MEMO, project web site: http://abnamro.com/memo.

OMG, 2000. Unified Modeling Language (UML), Version 1.3, www.omg.org.

O'Sullivan, D., Richmond, C., Power, T.; e-Business in the Supply Chain: creating value in a networked marketplace, Financial Times Business Limited, London, 1998.

Shishkov, B.; Barjis, J. Analysis of a System of Telecenters as a Distributed Network. Proc: 14th Int. SAER Conference, 2000, Varna - St. Konstantin Resort, Bulgaria.

Sparks, G., 2000. 'An Introduction to UML: The Use Case Model': http://www.sparxsystems.com.au.