DERIVING USE CASES FROM BUSINESS PROCESSES
The advantages of DEMO

Boris Shishkov, Jan L.G. Dietz
Faculty ITS, Delft University of Technology, 4 Mekelweg, Delft, The Netherlands
Email: Shishkov@IS.TWI.TUDelft.nl, j.l.g.dietz@its.tudelft.nl

Keywords: Use Cases; Business Process Modeling; DEMO; Norm Analysis; Petri Net

Abstract: The mismatch between the business requirements and the actual functionality of the delivered software application is considered to be a crucial problem in current software development. Solving this problem means to find out how to consistently place the software specification model on a previously developed business process model. If considering in particular the UML-based software design, we need to answer in this regard a fundamental question, namely – how to find all relevant use cases, based on sound business process modeling? Adopting the business process modeling as a basis for identification of use cases has been studied from three perspectives – Language/Action Perspective, Organizational Semiotics and Petri Net. The goal of the current paper is to study and analyze the strengths of DEMO concerning the derivation of use cases. This could be helpful not only for the investigation of DEMO but also for the further activities directed towards finding out the most appropriate way(s) of identifying use cases from business processes.

1 INTRODUCTION

The mismatch between the business requirements and the actual functionality of the delivered (software) application is considered to be an actual research problem in current software development (Shishkov & Dietz, 2002). In order to solve this problem, it is necessary to find out how to consistently place the specification of software on a business process model. It is worthwhile addressing these issues from the perspective of the Unified Modeling Language – UML (OMG, 2000) not only because of its completeness and wide applicability but also because UML turns out to be *de facto* the standard language for designing software, widely accepted by both researchers and practitioners.

Considering the mentioned problem from the perspective of UML leads directly to the notion of use case because, as it is well known, use cases are modeling constructs that serve to link the application domain (the business world) to the software domain, regarding the UML-based software development.

Ivar Jacobson introduced use cases in 1986, to be applied to requirements analysis (Jacobson et al, 1992). This was an essential contribution to UML where the use case concept plays a fundamental role. According to the concept, in a use case, a user performs a behaviorally related sequence of transactions in a dialogue with the (software) system (Fowler & Scott, 2000). Thus, a use case is a typical user / computer system interaction. A use case captures some user-visible function. This view suggests that developers of good use cases identify the users' goals, not the system functions. Based on these UML-related concepts, Alistar Cockburn further studies them, discussing how use cases should be developed and documented. Cockburn discusses the way in which use cases can be represented with varying levels of formality (Cockburn, 2001). The concepts of Jacobson and Cockburn were thoroughly investigated by Shishkov and Dietz (2001) from the point of view of their actuality for the development of UML. In UML, use cases (representing text documents) are implemented through the use case diagram which shows actors and use cases together with their relationships (OMG, 2000). The diagram itself is a graph of actors, a set of use cases, and the relationships between these elements (associations, generalizations, etc.). It might include also some interfaces. By representing the potential use cases for the system to be built and relevant actors, the diagram provides the starting point in system modeling. Therefore, the proper derivation of use cases and the construction of use case diagram are crucial concerning the task to place consistently the (UML-based) specification of software on prior

O. Camp et al. (eds.), Enterprise Information Systems V, 249-257.

business process modeling. Hence, it is essential to know how to derive use cases based on a sound business process model. We take into consideration that the software community still misses consistent guidance for the use case identification. Sound and complete methods for construction of UML use case diagram (Jacobson et al, 1992; Fowler & Scott, 2000; Shishkov & Dietz, 2001) on the basis of business process modeling are still needed.

This paper reports further results of a study directed towards derivation of use cases from business processes. Adopting the business process modeling as a basis for the identification of use cases has been studied (Shishkov & Dietz, 2002; Shishkov et al, 2002; Shishkov & Barjis, 2002) from three perspectives, namely: Language/Action Perspective and the DEMO theory in particular (Dietz, 1999), Organizational Semiotics and Norm Analysis in particular (Stamper, 1997), and Petri Net (Aalst, 1998). The goal of the current paper is, by considering the mentioned achieved results, to study and analyze the strengths of DEMO concerning the derivation of use cases. This could be helpful not only for the investigation of DEMO but also for the further activities directed towards finding out the most appropriate way(s) of identifying use cases from business processes.

Further on in this paper: The basic concepts regarding DEMO, Norm Analysis and Petri Net as well as the theoretical background for relating them to use cases are considered in Sections 2, 3, and 4, respectively. Section 5 illustrates through a case example how use cases could be derived, based on each of these tools. On this foundation, Section 6 studies which are the particular advantages of DEMO in deriving use cases, comparing this way of derivation to the Norm Analysis and Petri Net – based ones. Section 7 contains the conclusions.

2 DEMO

Dynamic Essential Modeling of Organizations - DEMO is a methodology for understanding, analyzing, (re)designing and (re)engineering business processes. Its underlying theory about organizations is rooted in the Language/Action Perspective (Flores & Ludlow, 1980), Organizational Semiotics (Liu, 2000) and Philosophical Ontology (Bunge, 1979). DEMO reveals the "construction" and "operation" of an organization, contrary to the current function and behavior-oriented approaches. It is characterized by three major features: 1) a white-box architecture of actors, production and coordination, 2) the extraction of the essence of business processes from their realization, 3) the transaction pattern.

Actors, production, coordination. Like every other system (e.g. an alarm clock or a racing car), the functional behavior of an organization is brought about by the collective working of the constructional components. The construction and the working of a system are most near to what a system really is, to its ontological description (Bunge, 1979). An organization is defined as a (discrete dynamic) system in the category of social systems. This means that the elements are social individuals or actors, each of them having a particular authority to perform production acts (P-acts) and a corresponding responsibility to do that in an appropriate and accountable way. The structure of an organization consists of coordination acts (C-acts), i.e. the actors enter into and comply with commitments regarding the performance of P-acts. The generic white-box organizational model (Figure 1) consists of: the actors, the P-world, and the C-world (Dietz, 1999).

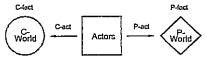

Figure 1: The white-box model of an organization

By performing P-acts, the organization does what it is supposed to do according to its function. C-acts serve to coordinate and control the performance of P-acts.

Essence, realization. Three perspectives on an organization are distinguished in DEMO: essential (the organization viewed as a system of authorized and responsible actors that create new original facts), informational (the organization viewed as a system of information processors that remember facts and derive new facts from existing ones), documental (the organization viewed as a system of formal operators that collect, transport, store, copy, destroy representations of facts) (Dietz, 1994).

Take for example the process of withdrawing money from a bank account using an ATM machine. Think of observing this process through essential, informational or documental "glasses" as a metaphor. Looking through documental "glasses" we see someone inserting a card into a machine, pushing buttons on a keyboard and finally getting out the card and other pieces of paper. Nothing with respect to the information on it or the purpose for which they are used, is seen. Looking at the same process through informational "glasses", we see someone providing information to an ATM system: a PIN code and specification of an amount of money. Also, the machine provides information if withdrawal is possible to the customer. We see that the machine outputs money and receipts. Looking through essential "glasses" shows responsible actors, their actions and interactions. A customer requests a bank to withdraw money from an account. The bank decides to do this and states that the money is withdrawn. Further on, the customer accepts it.

The transaction pattern: P-acts and C-acts appear to be performed in particular sequences that can be viewed as paths through a generic pattern called the (business) *transaction* (Dietz, 1999). A transaction is a finite sequence of C-acts between two actor roles, the customer and the producer. It takes place in three phases: the order phase (O-phase), the execution phase (E-phase), and the result phase (R-phase). O-phase is a conversation that starts with a request by the customer and that, if successful, ends with a promise by the producer. E-phase basically consists of the performance of the P-act by the producer. R-phase starts with the statement by the producer that the requested act is performed and ends, if successful, with the accept by the customer. The whole pattern of a transaction is represented by one symbol in the so-called Coordination Structure Diagram (CSD). Fig. 2 exhibits CSD for the money withdrawal example. The two boxes represent the two actor roles involved: A0(A1) is the customer(producer). The small black box indicates that A1 is the producer of T1 (and consequently A0 is the customer). The successful result of a transaction T1 is the P-fact "withdrawal W is performed" where W is constituted by the account, the amount and the time.

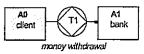

Figure 2: CSD of the money withdrawal example

Deriving Use Cases

A use case model can be consistently derived based on a DEMO business process model, as it has been studied in (Shishkov & Dietz, 2002). Applying DEMO, developers could provide a sound business process model for the software design, a clear model that captures the features which remain unchanged from realization. Such a model could be a proper basis for improving the delimitation, identification and the specification of the modeled software system (Dietz, 1994). Hence, DEMO possesses completeness and capability of capturing the essence of business processes. Therefore, if the developed software model stems from a DEMO business process model, the software designer would have the right (re)design freedom (Dietz, 1999). All this makes a DEMO model to be a sound basis for further software specification activities. For this reason, we consider it worthwhile exploring the use case derivation based on DEMO business process modeling. In this regard, it has been studied that DEMO transactions are straightforwardly relatable to the pieces of functionality concerning the software (further specified). Thus, deriving use cases based on DEMO is well founded theoretically. Next to that, the actors associated with the identified use cases would be a reflection of the DEMO actors because they concern these same transactions. All this makes the reflection of a DEMO business process model in a use case model complete, consistent and well founded theoretically.

3 NORM ANALYSIS (NA)

When studying organizations from the perspective of agents' behavior it is necessary to specify the norms based on which this behavior is realized. Norms (Stamper et al, 1997) are the rules and behavior patterns - formal or informal, explicit or implicit, existing within a society, an organzation, or even a small group of people working together to achieve a common goal (Liu et al, 2001).

Norms are determined by Society or collective groups, and serve as a standard for coordination of actions. If the norms can be identified, individuals' behaviors, hence their collective behaviors, are mostly predictable. From this perspective, to specify an organization can be done by specifying the norms (Stamper, 1992).

Four types of norms exist: evaluative, perceptual, cognitive and behavioral norms. Each type of norms governs human behavior from different aspects. In business process modeling, most rules and regulations fall into the category of behavioral norms. They prescribe what people must, may, and must not do, which are equivalent to three deontic operators "is obliged", "is permitted", "is prohibited". Hence, the following format is considered suitable for specifying behavioral norms:

whenever <condition>
if <state>
then <agent>
is <deontic operator>
to <action>

It is essential to recognize that norms are not as rigid as logical conditions. If a person does not drink water for certain duration of time he cannot survive. But an individual who breaks the working pattern of a group does not have to be punished in any way. For those actions that are "permitted", whether the agent will take an action or not is seldom deterministic. This elasticity characterizes the

business processes, therefore is of particular value to understand the organizations.

A NA is normally carried out on the basis of the results of the Semantic Analysis (for information on Semantic Analysis interested readers are referred to (Liu, 2000)). The semantic model delineates the area of concern of an organization. The patterns of behavior specified in the semantic model are part of the fundamental norms that retain the ontologically determined relationships between agents and actions without imposing any further constraints. However, NA could be successfully related also to other modeling tools, e.g. Activity diagram, Petri net.

In general, a complete NA can be performed in four steps: 1) **Responsibility analysis** – it enables one to identify and assign responsible agents to each action, focusing on the types of agents and types of actions. 2) **Proto-norm analysis** – it helps one to identify relevant types of information for making decisions concerning a certain type of behavior; the aim of this analysis is to facilitate the human decisions without overlooking any necessary factors or types of information. 3) **Trigger analysis** – it is to consider the actions to be taken in relation to the absolute and relative time, the absolute time means the calendar time, the relative time makes use of references to other events; the results of trigger analysis are specifications of the schedule of the actions. 4) **Detailed norm specification** - it concerns the specification of norms in a natural and a formal language versions; the goals of this are to capture the norms as references for human decision and to perform actions in the automated system by executing the norms in the formal language.

For those norms identified in the business processes, some refer to the major authorities and responsibilities of the major figures in the organizations. These norms govern some trivial, relatively less important norms or those of lower priorities, from the perspective of organizational functionalities. This strongly suggests the possible hierarchies exist not only in the organization structure, but also in the norms. Liu et al (2001) use the terms framing norms, contractual norms, etc. to express the hierarchies.

Deriving Use Cases

As studied in (Shishkov et. al, 2002), NA can be useful in creating a business process model to be reflected into a use case one. NA has widely been used as an effective and proven tool for investigation (in combination with Semantic Analysis) of business processes. Regarding a business system under study, NA specifies the rules according to which agents' behavior is realized. This is a potential link to use cases which represent functionality of a system, by defining its behavior. Hence, deriving use cases from a NA model would be useful and is put on sound theoretical foundation, since both modeling tools reflect behavior within business/software systems (norms describe rules of behavior; use cases represent pieces of functionality). Thus these could be methodologically related.

4 PETRI NET (PN)

Petri Net (PN) is a well known and widely used modeling technique (Aalst, 1998) that allows for consistent investigation of business systems by consideration of processes. 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 ordered in some way 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 business processes is important in order to design, maintain and improve the way businesses work, effectively and efficiently.

PN could be very useful in supporting software specification activities. As it is well known, in the modeling of software systems, 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. As already stated, all this needs to be based on business process modeling, represented through a sound graphical tool. This is in order to capture the system dynamics, to represent processes in time sequence, etc. It is claimed that PN could be successfully used for this purpose, including in the cases in which we consider UML as a system elicitation environment. Combining UML and PN has been studied by Shishkov & Barjis (2002).

PN have well supported tools to allow modeling, analysis, and, if necessary, simulation (execution) of systems. PN are formalism and a graphical language for the design, specification, and verification of systems. In order to better understand PN, there are some typical examples of PN depicted in Figure 3. These examples are especially chosen to demonstrate PN 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.

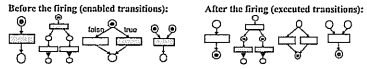

Figure 3: Typical PN examples

In Figure 3, 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 example shows synchronization or AND-join processes. In this situation it is necessary to finish both processes completely before starting the following one.

Deriving Use Cases

It was studied and demonstrated (Shishkov & Barjis, 2002) that a PN process/transition could be used as a consistent basis for use case derivation. We consider useful in this regard that PN is easily readable, simple and adapted to capture the dynamics of systems and processes; and next to that – there are many available tools supporting PN models of place transition type, e.g. simulation tools. All this is a guarantee that the use case model would stem from a consistently developed and verified business process model. Regarding the PN - use case mapping, it is well founded theoretically because PN models the dynamics of a business system, elaborating on the concrete business processes that reflect a particular behavior. As for use cases, they reflect the pieces of functionality concerning the designed system that is to support these same business processes.

5 THE HRB CASE

In this section, as said in the introduction, we will illustrate, through a case example, the three studied ways of deriving use cases from business processes, namely: based on DEMO, NA and PN. A system, representing a Hotel Reservation Broker (HRB) is modeled in order to illustrate this.

HRB matches the data about clients' required accommodation and hotel offers. Both the hotels and clients need to register in order to use the service for a selected period of time. The subscription fees for hotels are fixed depending on the chosen period and the hotel size; the fees are fixed for clients also, depending on the chosen period. Besides these subscription fees, both clients and hotels pay fixed fees when a match-making is realized. Further on, we refer to these fees as: a "reservation fee" (paid by a client) and a "hotel fee" (paid by a hotel). HRB accepts accommodation requirements from clients (e.g. check in/out dates, place, type of accommodation, price, etc) and accommodation information from hotels (e.g. number and type of rooms/beds available, etc.). Once HRB has received requirements from a client, if requested, it performs match-making on a real time basis. HRB provides the client with a list of available accommodation (all of them meeting the client's requirements) to select from. Once the client has accepted one of the offers, he pays the reservation fee. He has to pay also the cost of the selected accommodation. Then, HRB is obliged to guarantee the accommodation. HRB should contact the selected hotel and realize the actual booking of a particular room/bed. Then, the hotel must pay to HRB the hotel fee, and will be paid (by HRB) the cost of the reserved accommodation. Once this is done, the reservation is actually completed. The service is considered finished.

Further on in this section, we will illustrate how the use case model for HRB could be built based on DEMO, and what alternative ways for use case derivation are offered by NA and PN.

5.1 Deriving the use case model based on DEMO

We will first consider issues concerning the building of the DEMO model itself and afterwards – its reflection in the derivation of the use case model.

Table 1 - Business Transactions List

transaction type	result fact type
T1 *match-making*	F1 *match <M> is made*
T2 *subscription*	F2 *subscription <S> is arranged*
T3 *subscr. payment*	F3 *the fee for period <P> by <Cl./Hotel> is paid*
T4 *reserv. payment*	F4 *the fee for reserv. <R> by <Cl./Hotel> is paid*
T5 *accom. payment*	F5 *the cost for accom. <A> by <Cl.> is paid*
T6 *accom. compens.*	F6 *<Hotel> is compensated for ac. <A>*
T7 *refund*	F7 *refund <RE> for violation <V> is arranged*
T8 *reservation*	F8 *reservation <R> by <Hotel> is arranged*

After delimitation of the domain, the business processes to be supported by HRB are explored with DEMO. The eight identified business transactions (transaction types) are listed in Table 1 together with their corresponding resulting fact types.

The focus is only on transactions on the essential level, in order to keep the business model abstract enough so that it should remain unchanged during (eventual) re-design of its realization.

Based on the transactions and result facts, the system(s) to be investigated should be selected, relevant DEMO actor(s) – identified, and their roles – determined (as customer and producer). Once this is done, all interaction relationships are determined. All this is depicted in Fig. 4, representing the Coordination Structure Model – CSM (it is

incomplete, since the goal is only to illustrate the usage of DEMO for use case derivation).

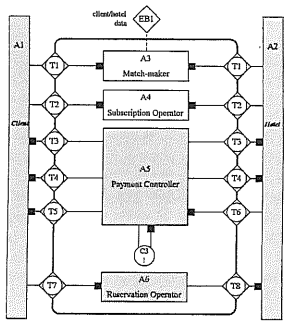

Figure 4: HRB – DEMO Coordination Structure Model

The system under study (HRB) is considered as well as the Client and Hotel (as actors). Regarding the system under study, it is represented on the figure in more detail: actors A3 and A4 (they are represented in white boxes, because they are elementary actors – involved in just one transaction each) are depicted as well as actors A5 and A6, whereas Client and Hotel are actors outside the system. The transaction types are represented by a symbol combining a disk and a diamond symbol. The small disk C3 represents a so-called conversation for initiation. It models the periodic activation of A5 to issue payment requests. The system boundary is represented by a grey round angle. There is a so-called external bank (EB1) which contains the accommodation data provided by hotels. The dotted line between EB1 and A3 means that actor A3 is allowed to inspect the contents of EB1. In other words, actor A3 is allowed to know the information provided by hotels. The reason for this allowance is that A3 needs to know the provided information. How A3 gets access and also how hotels add and remove data is not shown. These matters are considered to belong to the information and documental perspective and thus are not represented in the CSM.

As already stated, the DEMO transactions are straightforwardly mappable into use cases. The derived use case model is depicted on Fig. 5. The use cases are given numbers in order to trace directly from which DEMO transactions they are derived. This is illustrated in Table 2.

Table 2 - DEMO transactions and derived use cases

Source transaction (s)	Derived use case(s)
1	0;1
2	2
3	3
4	4.1; 4.2
5	5
6	6
7	7
8	8

As seen from Figure 5 and Table 2, it is possible that more than one use case is derived from one DEMO transaction. Transaction 4, for example, has the same essence no matter who the actor is. As for the use case model, it should clearly distinguish the cases in which the Client pays reservation fee from the cases in which the Hotel does this. The reason is that the realization issues are different in these two situations. Thus, two use cases are needed. As for Transaction 1, it is reflected in the use case "Perform Match-making", but in order for this functionality to be realized it is necessary the data accuracy to be checked first. However, this is not an essential business transaction, it is just information checking. For this reason, the use case "Check Data Accuracy", being related to the use case "Perform Match-making", is also derived from Transaction 1.

Most of the derived use cases reflect essential behavior, as opposed to some use cases that reflect, for example, information update, like the use cases – "Check Data Accuracy" and "Add data in DBC" (the abbreviation "DBC" will be explained below). As already discussed, although these use cases are not directly derivable from DEMO transactions (as the use cases that reflect essential behavior), the DEMO theory helps developers clearly understand their role and thus – precisely identify them. As far as actors are concerned, the DEMO CSM offers methodological guidance for determining (DEMO) actors that are further reflected in the use case model.

The diagram on Fig. 5 shows use cases and actors typical for such a HRB. Since the purpose is just illustrative, only some of the use cases and actors typical for such a system are considered. Regarding the diagram, the abbreviation "DB" stands for the database, used by HRB. For convenience, DB is virtually divided into DBC and DBH (containing data of offered and searched accommodation, respectively). The diagram shows 2 actors: *Client & Hotel*. Concerning Client (Hotel) – he takes the decision, has the responsibility, has the goal to add request (offer) in DBC (DBH), and/or remove it from DBC (DBH), and have his data matched up with relevant data from DBH (DBC). There are 16 use cases: "Add Data in DBC", "Check User's Inf.", etc. There are 3 <<include>> relationships ("Perform Match-making" requires "Check Data Accuracy"; "Add Data in DBC" & "Add Data in DBH" require "Check User's Inf.") and two <<extends>> relationships (in some cases, before adding their data to DBC/DBH, the system might request from Client/Hotel additional inf., so the basic use cases are "Add Data in DBC" and "Add Data in DBH", and they are extended with "Request Additional Inf.").

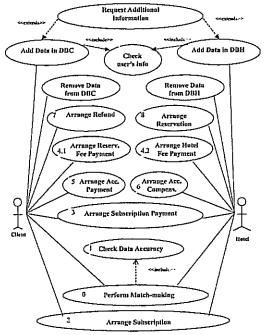

Figure 5: Use case diagram of HRB

As seen from this example, a DEMO business process model can be used as a consistent basis for derivation of a use case model. The DEMO transactions are straightforwardly mappable into use cases that reflect essential behavior. The DEMO theory offers methodological knowledge and guidance for clearly understanding and distinguishing between essential business transactions and transactions representing information update, for example. This could be useful as sound guidance in identifying further on the use cases that reflect information-related operations. As for actors, they directly reflect the DEMO actors which are consistently modelled based on the DEMO theory. Regarding the graphical representation, as seen from the example, the graphical notations of the DEMO CSM are very suitable from the point of view of use case derivation activities – the derivation process is easily visible and understandable. Next to these considerations it should be noted that a major advantage of this way of use case derivation is the consistency and completeness of DEMO as a business process modeling tool as well as the wide consideration and applicability of this tool.

5.2 Deriving use cases based on NA

Starting from the textual description concerning HRB, it is necessary to build a NA model on which to base use case derivation. In order to realize this, it is necessary, after delimitation of the domain, to draw an Ontology chart, as discussed in (Shishkov et. al, 2002). The norms are to be identified based on the Ontology chart. Because of the limited scope of this paper we are not going to depict the Ontology chart. Conducting Semantic Analysis and producing an Ontology chart, on the basis of textual description, is well studied and demonstrated in (Liu et al., 2001). It is also well studied how norms could be composed based on an Ontology chart. We will stress upon the issues connected with the derivation of use cases based on already constructed norms.

The following norms are chosen (depicted in Table 3) to illustrate the use case derivation.

Table 3 - Four norms concerning HRB

NORM: *Matchmaking* Whenever <The subscr. fee is paid by the Client/Hotel> If <The Client initiates the match-making> Then <HRB> Is <Obliged to> To <Perform the match-making>	NORM: *Subscr. Payment* Whenever <Client/Hotel has decided to use HRB> If <The Client/Hotel initiates subscription> Then <the Client/Hotel> Is <Obliged to> To <Pay the subscr. fee>
NORM: *Ac. Cost Payment* Whenever <The match-making is completed successfully> If <An accommodation is selected by the Client> Then <the Client> Is <Obliged to> To <Pay the ac.cost to HRB>	NORM: *Hotel Fee Payment* Whenever <An accommodation is selected by the Client> If <HRB has triggered booking procedure with a hotel> Then <the Hotel> Is <Obliged to> To <Pay hotel fee to HRB >

Based on these norms as well as on the theoretical foundation from Section 3, we derive the use case model, as shown on Figure 6 (only some of the use cases are depicted, for illustrative purpose).

It is easily seen from Figure 6 which are the norms each of the use cases is derived from (dotted line). The set of high-level norms (like the ones mentioned above) identified based on an Ontology chart is not enough to derive a complete use case model. Some use cases are to be derived, based on lower level norms (as "behavior" norms, for example). These lower level norms should be first identified based on relevant higher-level norms. Due to the limited scope of this paper we are not going in more detail concerning these issues which are well studied and demonstrated in (Shishkov et. al, 2002). The use case "Arrange Refund" (Fig. 5) is an example of a use case that comes from a lower level norm. It should be about the refund to be paid back to Client if a guaranteed accommodation is refused later for any reason. The norm should be identified

based on a higher-level one concerning HRB's obligation to guarantee a reserved accommodation.

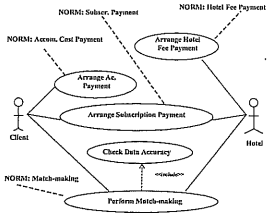

Figure 6: HRB – from NA to use cases

A great advantage of this way of use case derivation is that the high level norms complemented by lower level ones are a consistent basis for constructing a complete use case model. Concerning the use case model actors – they are not directly and formally identifiable from the NA model, although the contents of the norms make it clear which the actors are. A disadvantage of this way of use case derivations is that producing norms requires Semantic Analysis (or other business process investigation) to be conducted beforehand. This makes the modeling process more complicated.

5.3 Deriving use cases based on PN

Since it is considered well known how to derive a PN model based on textual description, we will start directly from the derivation of a use case model based on PN. Our constructed PN model is depicted on Fig. 7, the left side. To the right are several derived use cases (those same ones, considered in the previous sub-section). The PN and use case models are incomplete, because the purpose is just illustrative. It has only been demonstrated how the use cases considered in the previous subsection have been derived, now – based on PN.

It is seen that the use cases are straightforwardly derived from the PN model. For example, the use case "Perform Match-making" directly reflects the process/transition "HRB searches for matches" (the dotted line in Fig. 7 shows the connection between processes and corresponding use cases). As for the actors, they are derived indirectly, not as straightforwardly as e.g. from the DEMO model.

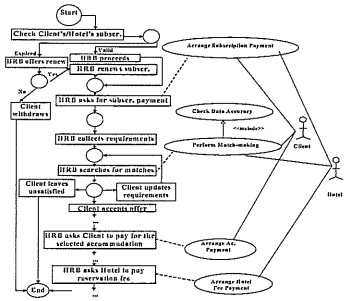

Figure 7: HRB – from PN to use cases

A significant advantage of this kind of derivation is the great popularity of PN models and the ease of building such models based on textual description. PN models are also well supported by relevant simulation tools that could be useful in some cases.

6 THE ADVANTAGES OF DEMO

Based on the realized theoretical and case study, some strengths of DEMO will be outlined, regarding the derivation of use cases from business processes.

A definite strength regarding DEMO is that its models are essential, fully abstracted from all realization issues. The completeness of the models is guaranteed by the DEMO theory that provides methodological knowledge on understanding and identifying essential business transactions as well as on distinguishing them from the realization issues. All this allows us to develop a sound and complete essential business process model. Placing the use case derivation on such a model, developers could be sure that the software specification activities are put on sound theoretical foundation.

Another strength of DEMO is the possibility (illustrated in Section 5) to straightforwardly derive: **1) use cases** that reflect essential behavior (from the DEMO transactions) **2) actors** in the use case model (from the DEMO actors). As for the rest of the use cases, the DEMO theory offers methodological knowledge (as stated), helpful for their derivation.

Regarding the ease and understandability of the derivation process, the used graphical notations are of significant importance. As seen from the example, the graphical notation of DEMO CSM is suitable with respect to the derivation of a use case model. The DEMO actors and transactions are depicted graphically in a very convenient way for their reflection into actors and use cases, in a use case model. This adds value to the derivation mechanism, making it easily understandable and illustratable.

Thus, use case derivation based on DEMO is well theoretically founded and easily implementable.

7 CONCLUSION

Our goal in this paper, as stated in the introduction, was to contribute to the knowledge on use case derivation based on business process modeling. We further continue the investigations in this direction that approach the use case derivation from three perspectives, namely – Language/Action Perspective (considering DEMO in particular), Organizational Semiotics, and Petri Net. Considering these alternative ways of use case derivation, we have studied and analyzed the advantages of DEMO in this regard. The study was supported by a case example.

It has been concluded, based on the conducted study, that the DEMO transactions and actors can be used as a consistent basis for deriving a use case model: in particular, use cases and actors, respectively. It is of particular importance that DEMO models consistently and completely the business reality, and guarantees in this way the soundness of the derived use case model. Regarding the derivation itself – the DEMO transactions are straightforwardly mapped into use cases that reflect essential behavior. The DEMO theory provides methodological guidance (by clearly distinguishing between essential and non-essential transactions) also concerning the identification of the rest of the use cases. All this contributes to the possibility to build a complete use case model. Next to that, the graphical notation of DEMO is suitable with respect to the derivation of a use case model. This makes the derivation process easy to understand and illustrate. For all these reasons, the DEMO-based use case derivation is claimed to be useful. There are issues however that are considered a subject of further study, in this regard, e.g. suggesting a more concrete guidance for identification of those use cases which do not reflect essential behavior.

The other two studied ways of use case derivation have also their advantages, as demonstrated in Section 5. However, this paper considers only the strengths of DEMO in this respect, identifying them from the perspective of the conducted study. It would be of benefit to further explore how the three considered tools could complement each other in deriving use cases.

The conducted study is expected to be helpful for current software development.

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.

Bunge, M.A. *Treatise on basic philosophy* (vol.4, D. Reidel Publishing Company, Dordrecht, NL, 1979).

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

Dietz, J.L.G. Understanding and Modelling Business Processes with DEMO, Proc. *ER*, Paris, FR, 1999.

Dietz, J.L.G. Business Modelling for Business Redesign, Proc. *27th IEEE Hawaii Int. Conference on System Sciences*, Los Alamitos, CA, 1994.

Flores, F. and Ludlow, J.J. *Doing and speaking in the office* (G. Fick and H. Sprague, Decision Support Systems: Issues & Challenges. NY, Perg. Press, 1980).

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

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

Liu, K., Sun, L., Dix, A., Narasipuram, M. *Norm-based Agency for Designing Collaborative Information Systems*. Info Systems Journal, 11, 2001.

Liu, K. 2000. *Semiotics in information systems engineering*. Cambridge University Press.

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

Shishkov, B. and Barjis, J. Modeling of e-Business Brokerage Systems Using UML and Petri Net. Proc: *17th edition of the IFIP World Computer Congress*, August 25-30, 2002, Montreal, Quebec, Canada.

Shishkov , B. and Dietz, J.L.G. Integrated Methodology Allowing Design of ICT Applications Based on Bus. Process Investigation. Proc: *IASTED Int. Conf. Applied Simulation & Modelling*, June 25-28, 2002, Crete, GR.

Shishkov, B. and Dietz, J.L.G. Analysis of Suitability, Appropriateness and Adequacy of Use Cases Combined with Activity Diagram for Business Systems Modeling. Proc: *3rd Int. Conference on Enterprise Inf. Systems (ICEIS)*, July 7-10, 2001, Setubal, Portugal.

Shishkov, B., Xie, Z., Liu, K., Dietz, J.L.G. Using Norm Analysis to Derive Use Cases from Business Processes. Proc: *5th Workshop On Organizational Semiotics*, June 14-15, 2002, Delft, The Netherlands.

Stamper, R., Liu, K., Hafkamp, M., Ades, Y. 1997. Signs Plus Norms – One Paradigm for Organizational Semiotics. Proc: *1st Int. Workshop on Computational Semiotics*, Paris, France.

Stamper, R., 1992. *Language and computer in organised behaviour*. In Riet, R.P.v.d. and Meersman, R.A., (eds.), Linguistic Instruments in Knowledge Engineering. Elsevier Science, The Netherlands.