

USING NORM ANALYSIS TO DERIVE USE CASES FROM BUSINESS PROCESSES*

Boris Shishkov¹, Zhiwu Xie², Kecheng Liu², Jan L.G. Dietz¹

¹*Faculty of Information Technology and Systems, Delft University of Technology, Delft, 2600 AJ, The Netherlands
E-mail: {b.b.shishkov, j.l.g.dietz}@its.tudelft.nl*

²*Department of Computer Science, The University of Reading, Whiteknights, Reading, RG6 6AY, UK
E-mail: {z.xie, k.liu}@reading.ac.uk*

ABSTRACT

Bridging software design and business process investigation appears to be a crucial research problem in modern software development. With respect to the UML-based software design, a fundamental question to be answered in solving the mentioned problem, is 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 by different researchers – it has been studied how use cases could be derived based on a DEMO and Petri Net business process models. The goal of the current paper is to tackle the problem from a new perspective in order to study the appropriateness of placing a use case model on a semiotic analysis. This could be helpful for identifying strengths of semiotic models, which could be useful for deriving use cases. We consider in particular Norm Analysis to be a proper semiotic tool for this purpose. We demonstrate derivation of use cases based on Norm Analysis through a case study.

Key words: Use cases; Norm Analysis; Business modeling; Software design; Unified Modeling Language (UML)

1 INTRODUCTION

UML (OMG, 2000) appears to be the standard language for conducting software design (Mallens et al, 2001). UML reflects the ideas of some of the most outstanding researchers and practitioners within the software community. Probably for this reason UML is widely accepted within the community and most of the current research on software design relates to UML. Rumbaugh, Booch and Jacobson state that the UML is meant for “visualizing, specifying, constructing and documenting the artefacts of a software intensive system” (Booch et al, 1999).

With regard to the design of software (including UML-based) that is intended to support contemporary business processes, a fundamental goal should be its alignment to prior investigation of these processes. The lack (in many cases) of such alignment, on a consistent basis, is considered by Shishkov & Dietz (2002) as a major reason for the great percentage of software project failure (Liu, 2000). Actually, we observe two opposite phenomena (Shishkov, 2002).

On one hand, we observe software being developed without prior (consistent) investigation of the (business) processes to be supported by it. This means that the business requirements are poorly determined and the software design model does not have its roots in a business process model. Therefore, the developed software would support the business processes inadequately to their needs; and although its quality might be high from a software point of view, the effectiveness of the support it realizes to the target business processes would remain low.

On the other hand, although (in many cases) sound business process modeling is conducted prior to the design of software, the business process model is only partially used, since it is not straightforwardly transformable into a relevant input for the software design. This does not allow for full employment of the software and ICT possibilities in solving the particular business problem(s).

Therefore, the two outlined tasks need to be aligned in a better way, as claimed by Shishkov & Dietz (2002). The business process investigation and the development of ICT (software) applications for the support of the business processes should be considered as one integrated task.

Different researchers address issues related to the outlined problem. Dehnert & Rittgen present a formal representation for describing business processes (Dehnert & Rittgen, 2001). These results could be of particular significance, especially if the work is further developed towards relating this representation to software design. Olivera, Filho and Lucena introduce an approach for conducting software design, based on business requirements analysis (Olivera et al, 2001). The approach appears

* The work reported in this paper was carried out during the first author’s visit at Staffordshire University, School of Computing, October – December 2001

to be consistent regarding the software related issues. However, it does not provide a straightforward mapping of a business process model into a software design model. Hikita & Matsumoto suggest a framework for development of adaptive systems (Hikita & Matsumoto, 2001). It is investigated how the appearance of additional requirements could be easily reflected in system's construction. But again, the framework does not provide a consistent mechanism for building a design model on the basis of business process investigation. Therefore, it might be concluded that further knowledge is still required in the direction of consistently basing application design on business process investigation.

Narrowing the outlined problem by considering in particular the UML-based design of software (that supports business processes), our goal should be to find out how to derive use cases based on a consistent business process model (because it is well known that use cases are modeling constructs that serve to link the application domain (the business world) to the software domain, regarding UML-based software development). We take into consideration that the software community still misses consistent guidance for 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 investigation are still needed.

Adopting the business process modeling as a basis for the identification of use cases has been studied by different researchers – Shishkov & Dietz (2002) have been investigating use case derivation based on DEMO; Shishkov & Barjis (2002) have been investigating use case derivation based on a Petri Net business process model. The goal of the current paper is to tackle the problem from the perspective of Semiotics in order to study the appropriateness of basing a use case model on a semiotic one. This could be helpful for identifying some advantageous features of semiotic models, which could be useful for deriving use cases. We consider Norm Analysis (NA) to be a proper semiotic tool for this purpose. We demonstrate derivation of use cases based on NA through a case study.

Further on in this paper: The basic concepts regarding NA and use cases are discussed in Sections 2 and 3, respectively. Section 4 studies how use cases could be derived based on NA. In Section 5, we explore a case example related to a hotel reservation system, in order to demonstrate the suggested NA-based use case derivation. Section 6 contains the conclusions.

2 BUSINESS PROCESS MODELING BASED ON NORM ANALYSIS

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 patterns of behaviour, either formal or informal, explicit or implicit, existing within a society, an organization, 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 the members to coordinate their actions. An individual member uses the knowledge of norms to guide his or her actions. If the norms can be identified, the behaviours of the individuals, hence their collective behaviours, are mostly predictable. From this perspective, to specify an organization can be done by specifying the norms (Stamper, 1992).

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

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

It is essential to recognise that norms are not as rigid as logical conditions. If a person does not drink water for certain duration of time he can not 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 characterises the business processes, therefore is of particularly 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 UML activity diagram, Petri net, etc.

In general, a complete NA can be performed in four steps: 1) Responsibility analysis; 2) Proto-norm analysis; 3) Trigger analysis; 4) Detailed norm specification. Responsibility Analysis enables one to identify and assign responsible agents to each action. The analysis focuses on the types of agents and types of actions. In an organization, responsibilities may be determined by the organizational constitution or by common agreements in the organization. Proto-norm Analysis helps one to identify relevant types of information for making decisions concerning a certain type of behavior. After the relevant types of information are identified, they can be used as a checklist by the responsible agent to take necessary factors into account when a decision is to be made. The objective of this analysis is to facilitate the human decisions without overlooking any necessary factors or types of information. Trigger Analysis is to consider the actions to be taken in relation to the absolute and relative time. The absolute time means the calendar time, while the relative time makes use of references to other events. The results of trigger analysis are specifications of the schedule of the actions. The detailed Norm Specification concerns the specification of norms in two versions, a natural language and a formal language. The purposes of this are (1) to capture the norms as references for human decision, and (2) to perform actions in the automated system by executing the norms in the formal language.

For those norms identified in the business processes, some refers 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.

3 THE IMPORTANCE OF USE CASES IN THE UML-BASED DESIGN OF SOFTWARE

With regard to the UML-based design of software, use cases have a crucial role in capturing the functional requirements that have to be fulfilled by the application software. These modeling constructs represent the system functionality, by defining its behavior without revealing its internal structure. They are incorporated in UML through the UC diagram – an essential UML diagram. It has a fundamental role in the UML development process, showing actors and UCs together with their relationships (OMG, 2000). 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 system modeling.

However, it is still a question how to properly identify UCs reflecting correctly the model of the processes to be supported by the developed software. Solving this problem would bring a solution also to the problem outlined in the introduction, regarding the alignment between software design and business process investigation, because if the UC model stems from a business process model, then the software development would be consistently placed on prior business process investigation. For this reason, the current section stresses upon UCs, outlining the most up-to-date concepts about these

modeling constructs. In the following sections, the relation between the UC model and business process modeling is explored, particularly from the perspective of Semiotics.

As for UCs, 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 et al, 1999). It should be noted also that Cockburn has developed his own consistent UC perspective (Cockburn, 2000) 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). The concepts of Jacobson and Cockburn are briefly outlined below.

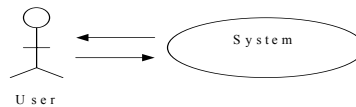


Figure 1: UC concept of Jacobson

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 Figure 1 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 UCs identify the users’ goals, not the system functions.

Jacobson provides description for several UC formalisms: 1) The basic UC consists of structured text description, including alternative and exceptional behavior. Scenarios may also be used to explain different perspectives of use. 2) There are two UC stereotypes - <<include>> and <<extends>>, and actor inheritance hierarchies. 3) There is a contract, specifying an object’s interface in detail. Actors may provide a contract that involves multiple UC. Conversely, a UC may provide a contract that multiple actors use. 4) A stimulus from an actor will cause the system to leave its current state and carry out some tangible amount of work, which is associated with pre- and post-conditions.

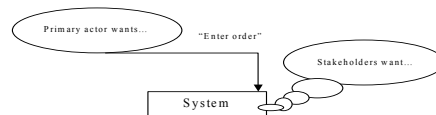


Figure 2: A UC as seen by Cockburn

Cockburn’s concept adds elicitation value to the fundamental UC theory. His work (Cockburn, 2000) is among the most ambitious material in the domain since 1992, when Jacobson introduced UC. According to Cockburn, a UC describes a system’s behavior. His view is illustrated in Figure 2.

In his model, 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 but at the same time it should protect 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. Three most widely used goal levels are suggested – user goal (the goal the primary actor has in trying to get work done), summary goals (involving multiple user goals), sub-function goals (those required to carry out user 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.

The outlined perspectives need to be **compared** considering the purposes behind them.

Aiming at looking inside UC, Cockburn has built a concept that allows 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, instead of focusing on the complete representation of a UC, the concept emphasizes on features that allow developers to show relationships which cover many UC 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.

However, in the current paper, basing our study on the concept of Jacobson (reflected in the UML), we take also into consideration the ideas developed by Cockburn.

4 DERIVING USE CASES FROM NORM ANALYSIS

As stated in the introduction, we claim that NA can be useful in bridging UML-based software design and business process investigation. Our arguments are as follows:

- (a) NA is an effective and proven tool for investigating (in combination with Semantic Analysis) business processes;
- (b) UCs mark the starting point in UML-based software design;
- (c) Regarding a business system under study, NA specifies the rules according to which agents’ behavior is realized;
- (d) UCs represent functionality of a system, by defining its behavior.

Hence, in trying to align software design and business process investigation, we claim that deriving UCs 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; UCs represent pieces of functionality). Thus these could be straightforwardly related.

In this section, we suggest how to base UC derivation on NA (Figure 3).

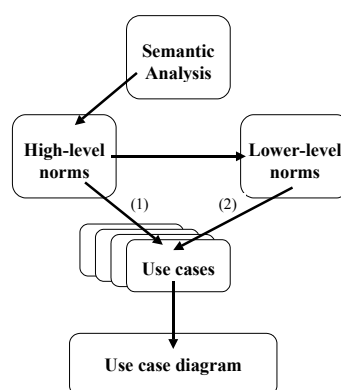


Figure 3: Deriving Use cases from Norm Analysis

It is clear that before deriving UCs out of norms, we need to produce a consistent and complete set of norms, reflecting the considered business system. The first step in doing this, as shown on the figure, is to conduct Semantic Analysis and construct Ontological chart. On this basis, it is possible to derive straightforwardly the High-level (HL) norms which include: Framing norms (these fundamental norms (concerning the model) which, as explained in Section 2, govern the general responsibilities underneath) and the rest of the higher priority norms. The HL norms can easily be derived based on Semantic Analysis and this is well-known from the Semiotic theory.

Once the HL norms are identified, it could be proceeded with the identification (based on them) of UCs that reflect essential behavior (Dietz, 1994). These UCs should be straightforwardly identifiable since both they and HL norms concern essential business system behavior.

Taking into consideration that in order to produce a complete UC model, we need not only the UCs reflecting essential behavior but also those UCs reflecting behavior elaboration, our development process suggests as well to further elaborate from the HL norms towards lower-level norms (e.g. Behavior norms). These norms are valuable for deriving from them UCs that reflect elaboration of essential behavior.

In summary, our suggested development process is claimed to be a contribution to the knowledge on aligning software design and business process investigation (in general) and on deriving UCs from business processes (in particular). According to the process, some UCs are straightforwardly derived from HL norms, others – from lower-level norms. This is illustrated in Section 5 using a case example.

5 THE HRB CASE

A system, representing a Hotel Reservation Broker (HRB) is modeled in order to illustrate the NA-based UC derivation.

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 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. The hotel will be paid (by HRB) the cost of the reserved accommodation. Once this is done, the reservation is actually completed and the service is considered finished.

Further on, we design (using the UML UC diagram) the functionality of such a software broker, based on a consistent investigation of the business processes to be supported by the developed software application. This investigation is based on Semiotics and is performed in the first, second and third phases of our study. The fourth phase deals with the actual UC derivation.

First. Based on the textual description and after delimitation of the domain, an Ontology chart (or another suitable business process modeling tool) needs to be applied. Based on it, we should further identify the HL norms corresponding to the system outlined above. 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 based on a textual description is well studied and demonstrated in (Liu, 2000).

Second. Based on an Ontology chart, the set of HL norms (including the Framing norms (called for short "f-Norms" in this paper) as well as the other higher-level norms) are identified. These are:

f-NORM 1

Whenever

<Client/Hotel has decided to use HRB>

If <The Client/Hotel initiates subscription>

Then <the Client/Hotel>

Is <Obligated to>

To <Pay the subscription fee>

f-NORM 2

Whenever

<The subscription fee is paid by the Client/Hotel>

If <The Client initiates the match-making>

Then <HRB>

Is <Obligated to>

To <Perform the match-making>

NORM 3

Whenever

<The match-making is completed successfully>

If <An accommodation is selected by the Client >

Then <the Client>

Is <Obligated to>

To <Pay the reservation fee>

NORM 4

Whenever

<The match-making is completed successfully>

If <An accommodation is selected by the Client >

Then <the Client>

Is <Obligated to>

To <Pay the accommodation cost to HRB>

NORM 5

Whenever

<An accommodation is selected by the Client >

If <The reservation fee and the accommodation cost are paid by the Client>

Then <HRB>

Is <Obligated to>

To <Guarantee the accommodation >

NORM 6

Whenever

<An accommodation is selected by the Client >

If <HRB has triggered the booking procedure with the corresponding hotel>

Then <the Hotel>

Is <Obligated to>

To <Pay the hotel fee to HRB >

NORM 7

Whenever

<An accommodation is selected by the Client >

If <HRB has triggered the booking procedure with the corresponding hotel>

Then <HRB>

Is <Obligated to>

To <Pay the accommodation cost to the Hotel >

NORM 8

Whenever

<The hotel fee is paid by the Hotel >

If <The accommodation cost is paid by HRB>

Then <the Hotel>

Is <Obligated to>

To <Reserve the accommodation >

Third. Based on the identified norms it is possible to derive lower-level norms, as “behavior” norms, for instance (we call this “norm elaboration”). As already stated, these norms are an important supplement to the basic identified norms, for the purpose of UC derivation, because a complete UC model needs to reflect the behavior elaboration as well. We have illustrated the norm elaboration with one norm derived from Norm 5:

NORM 5

Whenever

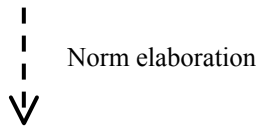
<An accommodation is selected by the Client >

If <The reservation fee and the accommodation cost are paid by the Client>

Then <HRB>

Is <Obligated to>

To <Guarantee the accommodation >



NORM A

Whenever

<An accommodation is guaranteed by HRB >

If <The accommodation is refused by the Hotel>

Then <HRB>

Is <Obligated to>

To <Return the accommodation cost, reservation fee, and penalty to the Client >

Fourth - development of a UML UC diagram, based on the built models. The diagram (Figure 4) shows UCs and actors typical for such a HRB. Since the purpose of this section is just illustrative, only some of the UCs and actors typical for such a system are considered.

Regarding the diagram, there are two actors represented on it: *Client* and *Hotel*. Concerning Client (Hotel) – he takes the decision, he has the responsibility, he has the goal to subscribe for using HRB and have his information matched up with relevant data about accommodation (potential clients).

The diagram contains eight UCs: “Perform Match-making”, “Arrange Refund”, “Arrange Reserv. Fee Payment”, etc. There is one <<include>> relationship (“Perform Match-making” requires “Check Data Accuracy”).

It is easily seen from the figure, which are the norms each of the UCs is derived from (dotted line). It is seen also that UCs are derived not only from HL norms but also from lower-level norms, which comes to indicate that in designing a software system it is essential to reflect not only the essential behavior (reflected by the HL norms) but also the behavioral elaboration regarding the considered business processes.

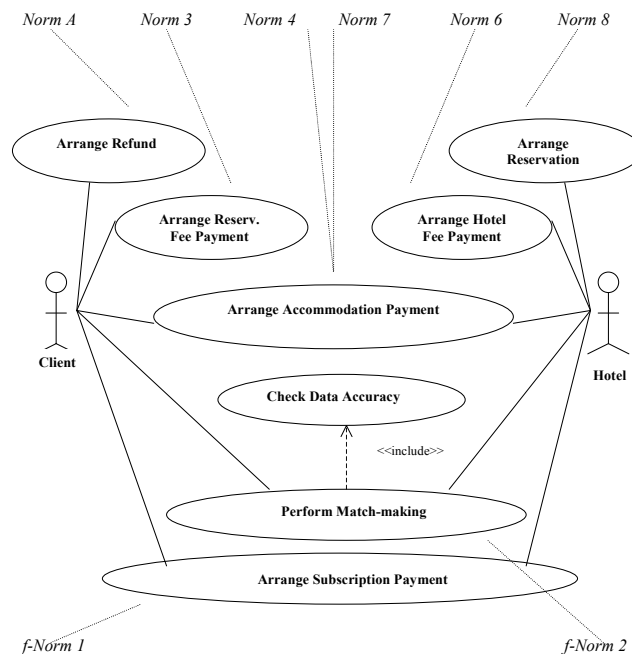


Figure 4: Use Case Diagram of HRB derived from NA

6 CONCLUSION

The paper's goal, as stated in the introduction, is to contribute to the knowledge on use case derivation based on business process modeling. We further continue the investigation of Shishkov & Dietz in this direction, by studying how a use case model could be placed on a semiotic analysis. In particular, it was shown that use case derivation could be consistently realized based on a business process model that is developed using Norm Analysis. This was demonstrated through a case study.

It was studied that norms (both HL and lower-level norms) are a suitable source for basing on them use case identification. However, it should be noted as well that producing norms requires Semantic Analysis (or other business process investigation) to be conducted beforehand – this is a drawback since it makes the modeling process more complicated.

The suggested use case derivation, based on Norm Analysis, is claimed to be useful and needs to be further studied and compared with the other existing ways of use case derivation. This could contribute to the efforts directed towards finding the most effective way of deriving use cases based on business process investigation. Succeeding in this would be a good contribution to software design, regarding the cases when software applications are built for the support of business processes.

7 REFERENCES

1. Booch, G., J. Rumbaugh, I. Jacobson, 1999. *The Unified Modeling Language User Guide*. Addison-Wesley, USA.
2. Cockburn, A., 2000. *Writing Effective Use Cases*. Addison-Wesley, USA.
3. Dehnert, J. and P. Rittgen, 2001. Relaxed Soundness of Business Processes. In the proceedings of the 13th Int. Conference on Advanced Information Systems Engineering, Interlaken, Switzerland.
4. Dietz J.L.G., 1994. Business Modelling for Business Redesign. In the proceeding of the 27th Hawaii International Conference on System Sciences, Los Alamitos, USA.
5. Dietz, J.L.G., 1999. Understanding and Modelling Business Processes with DEMO. In the proceedings of the 18th International Conference on Conceptual Modeling (ER), Paris, France.
6. Fowler, M. and K. Scott, 2000. *UML Distilled, Second Edition – a Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, USA.
7. Hikita, T. and M.J. Matsumoto, 2001. Business Process Modeling Based on the Ontology and First-order Logic. In the proceedings of the 3rd Int. Conference on Enterprise Information Systems, Setubal, Portugal. ISBN: 972-98050-2-4.
8. Jacobson, I., M. Christenson, P. Jonsson, G. Overgaard, 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, USA.
9. Liu, K., 2000. *Semiotics in Information Systems Engineering*. Cambridge University Press, London, United Kingdom.
10. Liu, K., L. Sun, A. Dix, M. Narasipuram. Norm-based Agency for Designing Collaborative Information Systems. *Info Systems Journal*, 11, 2001.
11. Mallens, P., J.L.G. Dietz, B.J. Hommes, 2001. The Value of Business Process Modeling with DEMO Prior to Information Systems Modeling with UML. In the proceedings of the 6th CAiSE/IFIP Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, Interlaken, Switzerland.
12. OMG, 2000. *UML, Version 1.3*. Object Management Group – www.omg.org.
13. Olivera, T.C., I.M. Filho, C.J.P. Lucena, 2001. Using XML and Frameworks to Develop Information Systems. In the proceedings of the 3rd Int. Conference on Enterprise Information Systems, Setubal, Portugal. ISBN: 972-98050-2-4.
14. Shishkov, B., 2002. Business Engineering Building Blocks. In the proceedings of the 9th CaiSE Doctoral Consortium, Toronto, Canada.
15. Shishkov, B. and J. Barjis, 2002. Modeling of e-Business Brokerage Systems Using UML and Petri Net. In the proceedings of the 17th IFIP World Computer Congress, Montreal, Canada.
16. Shishkov, B. and J.L.G. Dietz, 2002. Modeling of e-Business Brokerage Systems Using DEMO and UML. Chapter 11, *Building blocks for Effective Telematics Application Development and Evaluation*. TU Delft Edition, Delft, The Netherlands. ISBN: 90 5638 092 3.
17. Shishkov, B. and J.L.G. Dietz, 2001. Analysis of Suitability, Appropriateness and Adequacy of Use Cases Combined with Activity Diagram for Business Systems Modeling. In the proceedings of the 3rd Int. Conference on Enterprise Information Systems, Setubal, Portugal. ISBN: 972-98050-2-4.
18. 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, Amsterdam, The Netherlands.
19. Stamper, R., K. Liu, M. Hafkamp, Y. Ades, 1997. Signs Plus Norms – One Paradigm for Organizational Semiotics. In the proceedings of the 1st Int. Workshop on Computational Semiotics, Paris, France.