Design of Software Applications Using Generic Business Components

Boris Shishkov and Jan L.G. Dietz
Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4, 2628 CD, Delft, The Netherlands
E-mails: b.b.shishkov@ewi.tudelft.nl, j.l.g.dietz@ewi.tudelft.nl

Abstract

One frequent cause of software project failure is the mismatch between the (business) requirements and the actual functionality of the delivered (software) application. An approach is proposed in this paper, for design of software, basing consistently this design on prior business process modeling. The alignment between these two tasks is realized in a component-based way, by deriving the software model from identified (generic) business components, thus - taking advantage of the benefits of object-orientation. The paper introduces not only the concepts of the approach but also elaborated views on how it could be implemented using particular software design and business process modeling techniques. A way to implement the approach is through UML - the standard language for designing software. The suggested approach is expected to be a useful contribution to the knowledge on aligning business process modeling and software design.

1. Introduction

Software applications are supposed to have the crucial role of an environment between the technology (in particular – Information and Communication Technology – ICT) and the business processes supported by it. Hence, an essential issue for current business development is the effective application support.

Considering the development of software applications that should support (business) processes, one frequent cause of software project failure is the mismatch between the (business) requirements and the actual functionality of the delivered application. Actually, we observe two opposite phenomena [13].

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. Thus, the developed software would support the business processes inadequately. Although its quality might be high from a software point of view, the effectiveness of the support it offers 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: the business process modeling and the development of ICT applications for the support of the business processes. They both should be considered as one integrated task.

Many researchers address issues related to these problems. Most of the existing formal representations for describing business processes are not further related to software design. Olivera, Filho and Lucena have contributed in this direction, by investigating the design of software on the basis of business requirements analysis [11]. Their suggested approach is a step ahead even though it does not yet offer a straightforward mapping of a business process model into a software design model. Hikita and Matsumoto have studied how the appearance of additional requirements could be reflected in the system's construction [5], which is also a promising result achieved so far (although not completely solving the problem). Krutchen suggests (based on the existing use case concepts [6]) a "Business use case" – considered useful in bridging business process modeling and software design [8]. But it is still a question how to consistently identify such use cases. Therefore, it might be concluded that further knowledge is still required in

the direction of consistently basing application design on business process modeling.

With respect to the outlined research problem, a promising contemporary approach for application development is the component-based development [6], founded on the principles of object-orientation (OO). As it is well known, OO (characterized by the fundamental concepts of encapsulation, classification, inheritance and polymorphism) is widely considered as a special approach to the construction of models of complex systems, in which a system consists of a large number of objects. This applies not only to software systems but also to business systems [7]. Thus, it seems feasible to expect that software design and business process modeling could be bridged by basing the design on software components which are derived from some business components. Such components should fill the gap between the two mentioned tasks. If generic components are identified, they could be re-used for designing different applications. Next to that, component-based development seems beneficial for the application design itself. By basing application development on encapsulated, individually definable, reusable, replaceable, interoperable and testable (software) components, developers could build applications which possess durable configuration and a high degree of flexibility and maintainability. The process of application development would also be improved because building new applications would include using already developed components. This reduces development time and improves reliability. The performance and maintenance of developed applications would be enhanced because changes could occur in the implementation of any component without affecting the entire application. All this makes the component-based application development much more effective than the traditional way of application development.

For all these reasons, in considering the problem of alignment between business process modeling and software design, we focus in particular on realizing this on the basis of (generic) business components identified from target business processes. By basing the design of applications on such components, it is expected that the application support to business processes can be improved considerably. In this paper, the SDBC (SDBC stands for Software Derived from Business Components) approach is introduced. The approach allows for specification of software based on identified (generic) business components.

The outline of the paper is as follows: Section 2 outlines the essential issues behind SDBC. Section 3 briefly introduces SDBC. Section 4 elaborates on some aspects of the implementation of the approach. Section 5 contains the conclusion.

2. SDBC – essential foundations

Being an approach for specifying software on the basis of identified (generic) business components, SDBC is based on four essential fundaments: 1) integrated view over business process modeling and software specification; 2) the DEMO [4] transaction theory; 3) the principles of component-based system development; 4) re-use requirements. These four fundaments are elicited further on in this section.

Integrated view over business process modeling and software specification. SDBC integrates business process modeling activities and software specification activities (as shown on Fig. 1), contrary to the current software design approaches which consider business issues from software design point of view.

br: business reality
bpm: business process modeling
ss: software specification

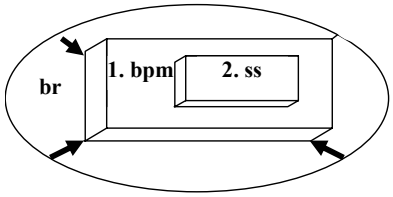


Figure 1: Integrated view over business modeling and software specification

As seen from the figure, before conducting the software specification, it is necessary to realize consistent business process investigation that thoroughly reflects the considered business reality. This investigation should result in a business process model that grasps all essential business issues. Based on such a model, the software specification model should be derived. This would allow for precise reflection of the business requirements in the functionality of the specified software.

DEMO transaction theory. The Business Process concept is essential for SDBC. This concept is fundamentally based on the DEMO (Dynamic Essential Modeling of Organizations) transaction theory (information on the theory is to be found in [4]), as depicted in Figure 2.

LAP: Language/Action Perspective
OS: Organizational Semiotics
PO: Philosophical Ontology

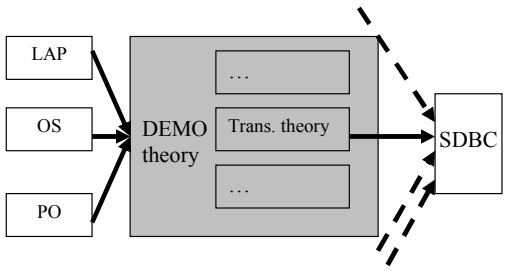


Figure 2: The importance of DEMO Transaction theory for SDBC

In complementing the DEMO Transaction theory, the concept *starting transaction* is suggested. A starting transaction is defined as a transaction that is not triggered by another transaction but may trigger other transactions.

A definition of a Business Process (considered crucial for SDBC) is proposed, which is based on the DEMO Transaction theory:

> ***Definition 1**: A Business Process is a collection of connected transactions that are realized in order to fulfil a starting transaction.*

Further developing Definition 1 would lead to the Business Component concept:

> ***Definition 2**: A Business Component is a model in which a Business Process is modeled. The model should be characterized by a good representation of the Business Process (according to Definition 1), providing elicitation on the considered transactions (including elicitation on the links that some of them realize to the outside environment) as well as on the actors involved.*

Hence, a Business Component is seen as a part of a system, that has a clearly defined function and clearly defined interface to the other parts.

As for defining a Software Component, some existing concepts [17] are considered. Based on them, the following definition is suggested:

> ***Definition 3**: A Software Component is a self-contained part of a software system, possessing its particular functionality.*

The Business Component concept and the Software component concept are essential for SDBC. They relate to the third fundament essential for SDBC, namely the component-based system development.

Principles of component-based system development. Conducting the business process modeling in a component-based way and specifying the software in a component-based way is of significant importance for SDBC. As depicted in Fig. 3, the studied business reality is to be reflected in a set of identified business components. Based on these components, a (component-based) software model is to be specified. It should be noted that the business and software components are not to be mapped always one to one – the bottom line in developing the business process model should be a business-oriented study, while the software specification (and integration), though derived based on the business components, is to be realized from the perspective of the functionality of the software system under development.

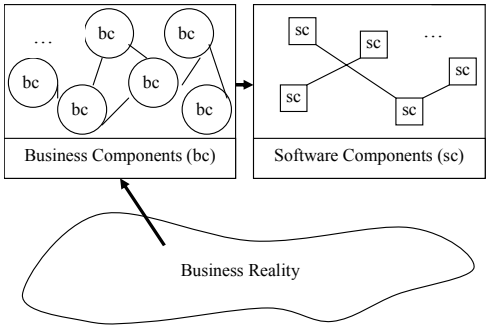


Figure 3: From business components to software specification

Hence, following the principles of component-based system development brings all the advantages associated with this type of system development to both the business process modeling phase and the software design phase. If some business requirements change, it would be possible to replace a business component with another one, without affecting the entire business process model. Next to that, in some cases business components may be re-used (this issue is considered further on). As for the software components, as already said, they are to be individually definable, testable, maintainable and so on. As for the alignment between business process modeling and software specification, the fact that this alignment concerns mapping of one component-based model to

another component-based one, contributes to consistency of the alignment.

Re-use requirements. As stated already, re-use is an essential issue for SDBC. It benefits from the advantages of re-use both in the business process modeling phase and in the software specification one. This is depicted in Fig. 4.

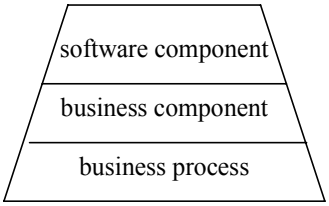


Figure 4: Levels of re-use

As seen from the figure, three re-use levels are to be distinguished: business processes, business components, and software components can be considered for the purpose of re-use.

Regarding the specification of a business process as a set of transactions, if this is generalized, then the specified business process could be re-used for the development of different business components.

As for business components, if general or generic ones (discussed below) are identified, they could be re-used in the specification of different software artefacts.

As long as software components are concerned, their reusability is left beyond the scope of this reported study, since the existing knowledge on integrating software, based on re-usable software components, is considered to be sufficient. However, in general, the logic of re-use is considered to be analogous with business components and software components.

Regarding the derivation of a business component, the most trivial way to do this is by developing a model on the basis of a business process. However, it is also possible to derive a business component using re-usable patterns: considering SDBC, we distinguish between *general* business components and *generic* ones (Fig. 5). General business components are models which reflect core issues and can be applied from different perspectives. For example, a general brokerage model could be further developed – in one way for building an e-trade system and in another, for building a hotel reservation system. Hence, a general business component needs to be extended depending on the purpose of use. On the contrary, a generic business component should contain in itself several optional extensions. Through parameterization, such a component can be adjusted depending on the purpose of use.

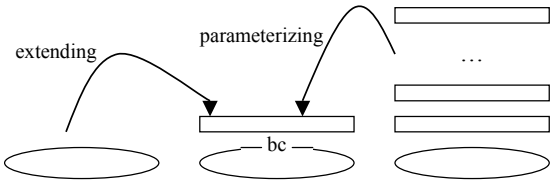
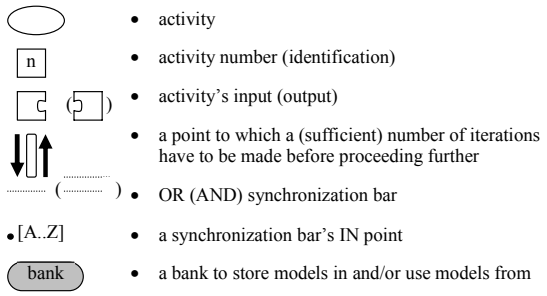


Figure 5: Extending a general component or parameterizing a generic one

3. SDBC - outline

Based on the essential SDBC fundaments, (already introduced) this section outlines the approach. Two graphical tools are developed for this purpose: Activity Model and Input/Output Model. The development of such tools was considered necessary because neither of the popular existing activity techniques (e.g. Activity Diagram, Flow charts and so on) proved to be sufficiently effective to thoroughly represent SDBC's steps, providing information on both the dynamics of the activities to be realized and their inputs and outputs.

The Activity Model (Fig. 6) represents the dynamics of the steps to be realized in implementing SDBC; the Input/Output Model (Fig. 7) represents the inputs and outputs of each activity. The legend regarding the graphical representation of these tools is as follows:



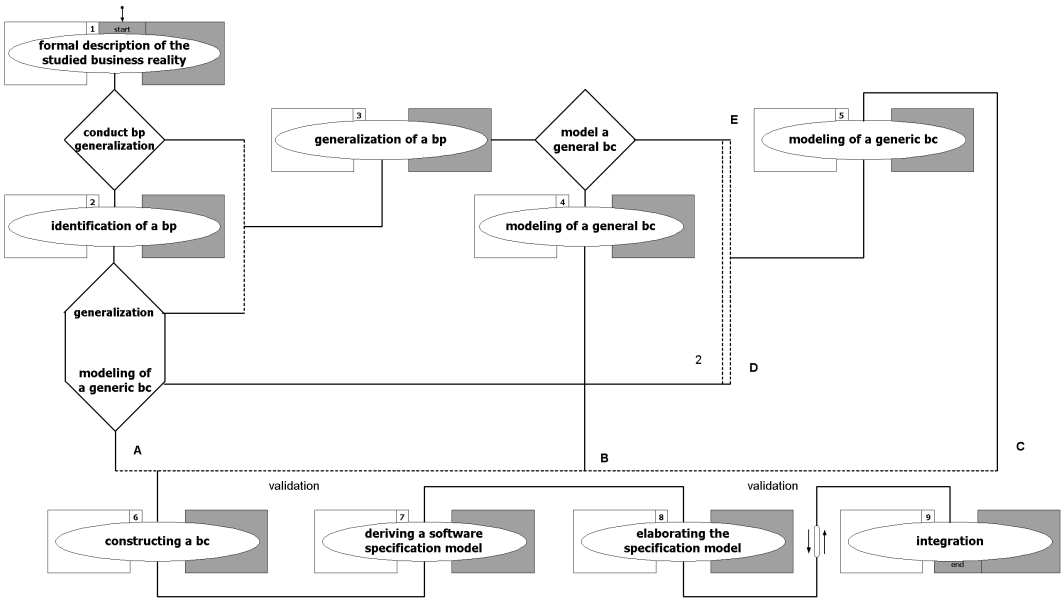


Figure 6: SDBC – Activity Model

As seen from Fig. 7, the starting input for implementing SDBC is any (informal, unstructured) description of the business system to be considered. This might be a textual description, a graphical model, a conversation or any other form. The first activity's output should be a formal and well-structured description of the studied system. This description should thoroughly reflect the considered business reality.

As seen from Fig. 6, the first decision to be made is whether the developed formal description should be used for the specification of a particular business process (e.g. hotel reservation match-making) or for achieving a generalized view (e.g. match-making). This decision should be based on consistent criteria developed by studying the particular domain. For example, it might be known that an issue is unique for a company and thus, there is no sense to develop a generalized model of it. As seen from Fig. 6, such a generalized model can be developed not only from a formal description of the studied business system but also based on the specification of a particular business process (this should be done if such a specification will be further needed by the modeler). Such a specification might be used also for building a generic business component. As seen from Figures 6 and 7, for modeling such a component, the required input is a specification of at least two (seen from the "2" at point D, Figure 6) particular business processes AND a specification of one general business process. The reason is that the generic model would require not only core specification (derived from a general business process) but also at least two realizations to be offered as selection options (options selected through parameterization).

A general business process is sufficient for building a general business component (Activity 4). It is possible that the developed general and/or generic business components are stored in banks and be used further on. It is possible also, besides storing them for future purposes, to use them as a basis for the development of business components – by extending a general business component or by parameterizing a generic business component. It is possible of course, to develop a business component using directly a particular business process specification (if coming through point **A** – Fig. 6). A developed business component is to be reflected in a software specification model (as seen from Fig. 7, there is an option also to use a prefabricated software component from an external banc; however, this is left beyond the scope of the current study since SDBC considers the issues related to the alignment between business process modeling and software specification). It is essential that the model be consistently derived from the source business component. Further on, the specification

model is to be elaborated regarding its structure and dynamics in order to bring sufficient elicitation for the further software design activities. Based on a number of such elaborated models, system integration is to be conducted. This would put together the developed software subsystems. All these steps are to be validated.

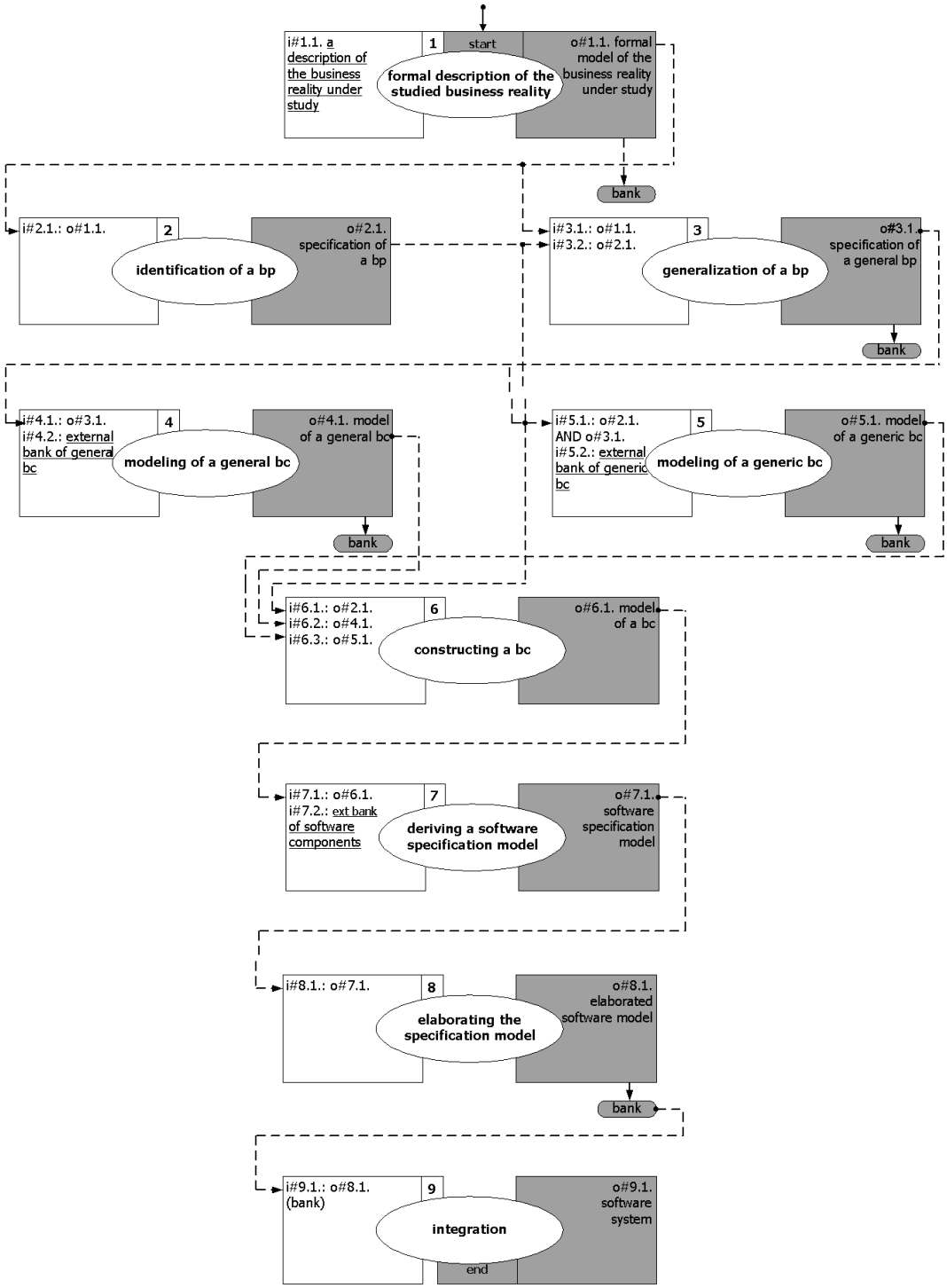


Figure 7: SDBC – Input/Output Model

4. SDBC – implementing the approach

Although the implementation of SDBC is not to be limited to particular tools, to consistently validate it, we have studied modeling techniques and tools (and also their consistent combination) through which SDBC could be applied.

4.1. From business components to software specification

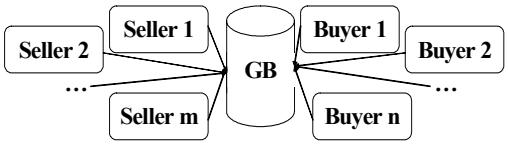
The suggested implementation of SDBC is fundamentally based on UML [12] – the *de facto* standards language for designing software. This means to reflect an identified business component (activity nr. 6, Fig. 6,7) in the specification of a use case model since, as it is well known, use cases serve to link the application domain (the business world) to the software domain, in the UML-based software design. Further on, the use case model needs to be structurally and dynamically elaborated (activity nr. 8, Fig. 6,7). And finally, such different models (sufficiently elicited) are to be integrated (activity nr. 9). Thus, this modeling phase includes three major tasks: use case derivation; use case elaboration; integration.

Use case derivation. Although there are a number of studies related to use cases [6,3] the problem of identifying use cases is however not satisfactory resolved yet [15]. The problem of deriving use cases from business processes has been studied from three essential business process investigation perspectives: Language/Action Perspective, Organizational Semiotics, and Petri Net. The achieved results were analyzed and conclusions were drawn that a sound solution of the use case derivation problem would be basing the derivation on DEMO (and also extending DEMO with semiotic Norm Analysis [9] in some particular cases), achieving in this way a derived use case model consistently rooted in an essential and complete business process model [15].

Use case elaboration. As for the elaboration of a particular use case, inspired by the ideas of Cockburn [3], a formal way of elaborating on a use case was suggested [14].

Integration. Since the research on integration of a number of specification models is still under development, this issue is left beyond the scope of this paper.

Example. An example is used to illustrate the use case derivation and elaboration. A general business component (representing a DEMO business process model) is considered, namely a "General Broker" (GB). GB is supposed to be extensible for use in a particular domain, Tele-Work (TW) [14]. Similar core brokerage functionality is required by TW, e-trade, and hotel reservation brokerage systems. Hence, it seems feasible to expect that identifying a GB (general for these domains) would allow us easily re-use this component for building different brokerage systems, for example a TW brokerage system (TWBS). In this case, what should be the functionality of such a GB? It should match the data of those looking for something (e.g. TW positions, goods, accommodation), we will call them "Buyers", and those offering such issues ("Sellers"). The general view of the required functionality of a GB is depicted in Fig. 8:



There are:
a) different Sellers (**S**) aiming at succeeding to sell their goods as quickly as possible;
b) different Buyers (**B**) aiming at purchasing specific goods they are interested in.

GB is supposed 1) to let S *i* find the B interested in the goods offered by him; 2) to let B *j* find the S offering the goods he is interested in.

S and **B** could, for example, pay on a subscriptional basis for the realized service.

Figure 8: GB - functionality

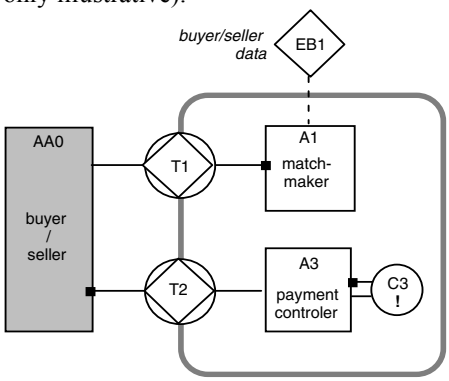
Anyway, behind this not so complex general functionality, there are issues to be considered when further extending the model and developing a software application: how to store, operate and maintain the data; how the application should provide its services to users, how some non-standard situations should be approached, and so on. These issues are partially addressed in the following modeling steps.

First. Based on the description of the required functionality, DEMO should be applied to explore the business processes to be supported by the software under development. For more information on DEMO readers are referred to [4]. From the description, two essential business transactions (transaction types) are identified (how we arrive here starting from a textual information (Activity 1, Fig. 6,7) is considered in the second half of this section). They are listed in Table 1, together with their corresponding resulting fact types. The focus is only on transactions on the essential level. That is in order to keep the business model abstract enough so that it should remain unchanged during (eventual future) re-design of its realization.

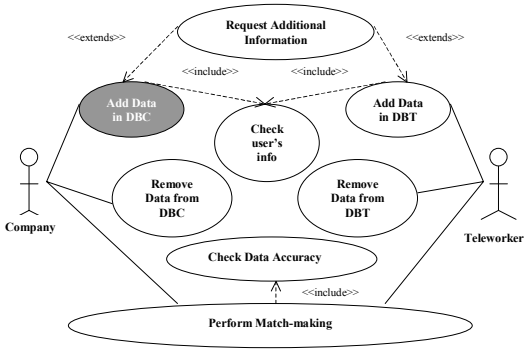
Table 1: Business transactions List

transaction type	result fact type
T1 match-making T2 payment	F1 *match <M> is made* F2 *the fee for period <P>* by <S/B> is paid

On the basis of the transactions and result facts, the system(s) to be investigated should be selected, relevant DEMO actor(s) - identified, and their roles (customer/producer) – determined, as well as all interaction relationships. All this is depicted in Fig. 9 A, representing the Coordination Structure Model or CSM (the model is incomplete, because the purpose is only illustrative).



A. DEMO CSD



B. Use case diagram

Figure 9: Coord. Str. Diagram of GB (A) and Use case diagram of TWBS (B)

The explored system (GB) is considered as well as the Seller and Buyer (as actors). Regarding the system, it is represented on the figure in more detail: actors A1 and A3 (white boxes) whereas the Seller and Buyer are taken together in the aggregate actor AA0 (grey box) since they play the same role towards A1 and A3. The transaction types are represented by a symbol combining a disk and a diamond. The small disk C3 represents a so-called conversation for initiation. It models the periodic activation of A3 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 data provided by sellers/buyers. The dotted line between EB1 and A1 means that A1 is allowed to inspect the contents of EB1. The reason for this allowance is that A1 needs to know the provided information. How A1 gets access and also how sellers/buyers add/ remove data is not shown. These matters to belong to the informational and documental perspective and thus are not represented in the (essential) CSM.

Second: extending the general business component and deriving a use case (UC) model. Due to the limited scope of this paper, the extension of the DEMO model and the UC derivation procedure are omitted. The diagram (Fig. 9 B), to be derived from an extended version of the model depicted in Fig. 9 A, shows UC and actors in the context of TWBS. Only some of the UC and actors typical for such a system are considered. The actions (important for the software design) which represent information providing but are not essential business transactions are additionally identified in building the UC model.

Regarding the diagram, "DB" stands for the database, used by TWBS. For convenience, DB is virtually divided into DBC/DBT (containing data of offered/searched TW positions, respectively). There are two actors: Company and Teleworker. Concerning Company (Teleworker) – it(he) takes the decision, has the responsibility, has the goal to add/remove data in/from DBC (DBT), and have its(his) data matched up with relevant data from DBT (DBC). The diagram contains eight UC: "Add Data in DBC", "Request Additional Inf.", etc. The UC "Add Data in DBC" is highlighted since it will undergo the further investigation steps. There are three <<include>> relationships ("Perform Match-making" requires "Check Data Accuracy"; "Add Data in DBC" and "Add Data in DBT" require "Check user's inf.") and two <<extends>> relationships (in some cases, before adding their data to DBC/DBT, the system might request from Company/Teleworker additional data, so the basic UC are "Add Data in DBC" and "Add Data in DBT", and they are extended with "Request Additional Inf.").

Third – further investigation of any particular UC of interest, based on the concept of Cockburn [3]. We have selected, for illustrative purpose, the UC "Add Data in DBC", and the mentioned investigation is applied to it – Fig 10 (only those extensions related to activity six, from the Main success scenario, are depicted.

The UC's scope is "system" (as opposed to "enterprise") since an interaction with a computer system is described. The indicated "summary" level means that the UC is long running (executed over months or years), showing the context in which the user goals operate.

Fourth: a dynamic elaboration realized through the construction of an activity diagram (AD) model for the chosen UC. As seen from the main success scenario, there are nine core activities (plus extensions) in the UC "Add Data in DBC". Some of them are shown on Fig. 11 as an overall AD. And finally, from the AD model it is straightforward to proceed with computer simulation, in order to validate the model [1].

4.2. Deriving and modeling business components

Two essential issues for the initial phase of SDBC (Activities 1 to 6 – Fig. 6, 7) are: deriving a formal model from any unstructured information about the studied business reality and specifying a business process; developing a business component (it might be also general or generic) based on this. Due to the limited scope of this paper, these issues are only partially illustrated. It is demonstrated below how, based on textual information, a formal model is developed and based on it, a DEMO Business Transactions Table is derived. Semantic Analysis (SA) reality, as well as to correctly distinguish between these issues.

The considered example is about the Uniccord Ltd (UCD) company [16]:

UCD deals with consultancy, sound recording and accommodation booking. A software system is to be developed; it should facilitate the organization of some core business activities of UCD, and operate between the company (Co) and its clients (Cl). To use the services of Co, Cl needs to subscribe. Three types of subscriptions are offered – subscription for one/three of the types of services (consultancy, recording, or booking), and group subscription – if more than one Cl subscribe together, they pay a special price under the condition that afterwards each of the Cl that belong to the group uses at least once a service by Co within the subscription period. Besides the subscription fees, Cl has to pay for the particular service, realized by Co. In calculating the cost of a particular service, there are issues which are common for all of the service types (for instance: costs of an order handling) and others that differ depending on the type of service (for instance: accommodation costs or costs for recording). The payments that Cl makes to Co should be specified and handled. Once a Cl has decided to order a service to UCD, he/she should contact the receptionist who fills in a standardized form.

Starting from the textual description and after delimitation of the domain, SA is conducted. Resulting from this follows the building of an Ontology chart – Fig 12-1. The Ontology Model is incomplete because the purpose is just illustrative. We will not elaborate on

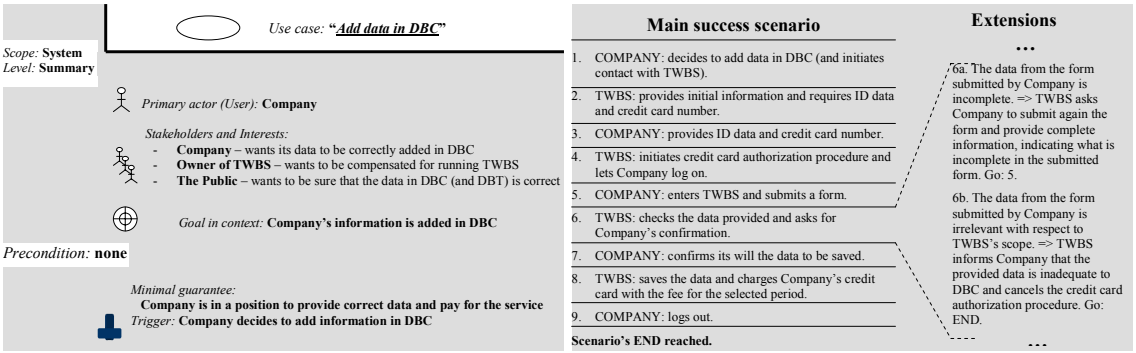


Figure 10: UC elaboration

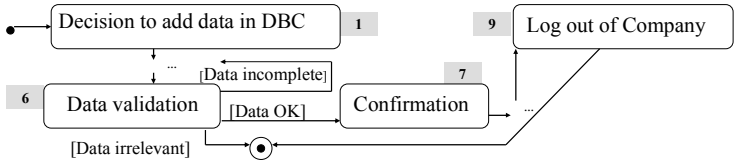
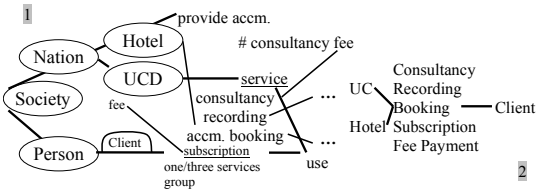


Figure 11: AD model for the UC: "Add data in DBC"

[9] is to be used for building the formal model, because, as studied in [2,16] this semiotic method possesses the capability of structuring the information concerning requirements in such a way that it is well understandable for both developers and potential users. Next to that, SA, possessing sound theoretical foundation rooted in Semiotics, allows developers to straightforwardly and precisely discover both specific and generic issues characterizing the investigated SA activities as well as on the construction of the Ontology chart. Conducting SA and producing Ontology Model based on textual description is well studied and demonstrated in [9]. As seen from the chart, the actors (or agents) will be: UCD, Hotel and Client – UCD provides services to Client; Hotel is involved in the accommodation services; Client is the consumer of the services. The actions related to these actors are: consulting, recording and so on. This is

depicted on Fig. 12-2. Based on this, it is proceeded with precise identification and specification of the transactions – Fig.12-3.



transaction type	result fact type
T1 subscription	F1 subscription <S> is made
T2 consultancy	F2 consultancy <C> by <UC> is realized
T3 recording	F3 recording <R> by <UC> is realized
T4 booking	F4 accm. booking <A> by <Hotel> is realized
T5 payment	F5 the fee for service <S> by <Client> is paid

3

Figure 12: From an Ontology Model to derivation of DEMO transactions

5. Conclusion

The paper's goal, as stated in the introduction, was to reveal the authors' ideas on aligning business process modeling and software specification, by introducing the SDBC approach. SDBC is supposed to complement the existing knowledge in this area and open a discussion on issues that need further study.

By outlining the theoretical foundation behind the approach, the authors show their research perspective in treating problems related to business process modeling and software design. By considering issues connected with the implementation of the approach, authors show their ideas about its practical application.

In fact, the fundamental goal behind the suggested approach is related in one way or another to the goals behind Tropos [10] and other consistent approaches addressing software system development. However, among the distinctive beneficial features of SDBC, to be considered, are the following, concerning the goal of sound software specification:

- it stems from a complete and consistent business process study;
- it is aligned to prior business modeling in a component-based way, benefiting from the advantages of object-orientation;
- it is implementable through the standard language for modeling software systems;
- it is based on re-usable patterns.

The realized research is expected to be a helpful contribution to the knowledge on specifying software, consistently basing this on business process modeling.

6. References

[1] Barjis, J. & B. Shishkov. UML Based Business Systems Modeling and Simulation. Proc: 4th Int. Eurosim Congress, 2001, Delft, NL.

[2] Chong, S. & K. Liu. A Semiotic Approach to the Design of Agent-mediated E-commerce Systems. Proc: 4th Int. Conference ISCO'99, Leiden, NL.

[3] Cockburn, A. Writing Effective Use Cases. Addison-Wesley, USA, 2001.

[4] Dietz, J.L.G. Generic Recurrent Patterns in Business Processes, in: Aalst, W. van der, Hofstede, A. ter, Weske, M. (eds.), Business Process Management, LNCS 2678, Springer-Verlag, 2003.

[5] Hikita, T. and M.J. Matsumoto. Business Process Modeling Based on the Ontology and First-order Logic. Proc: 3rd Int. Conference on Enterprise Information Systems (ICEIS) 2001, Setubal, PT.

[6] Jacobson, I.; M. Christenson; P. Jonsson; G. Overgaard. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Reading, MA, 1992.

[7] Jacobson, I., M. Ericsson, A. Jacobson.. The Object Advantage, Business Process Reengineering with Object Technology. Addison-Wesley, US, 1995.

[8] Kruchten, P. The Rational Unified Process: An Introduction, Second Edition. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2000.

[9] Liu, K. Semiotics in Inf. Systems Engineering. UK: Cambridge University Press, 2000.

[10] Mylopoulos, J.; M. Kolp; J. Castro. UML for Agent-Oriented Software Development: the Tropos Proposal. Proc: 4th Int. Conf. on UML, 2001,Toronto, Ontario, CA.

[11] Olivera, T.C., I.M. Filho, C.J.P. Lucena. Using XML and Frameworks to Develop Information Systems. Proc: 3rd Int. Conference on Enterprise Information Systems (ICEIS) 2001, Setubal, PT.

[12] Object Management Group (OMG). UML, Version 1.3. http://www.omg.org, 2000.

[13] Shishkov, B. Business Engineering Building Blocks. Proc: 9th Doctoral Consortium on Advanced Inf. Systems Eng. (CAiSE), 2002, Toronto, ON, CA.

[14] Shishkov, B. and J.L.G. Dietz. Design of Tele-Work Brokerage Systems Using DEMO-UML Based Generic Components. Proc: IADIS Int. Conference WWW/Internet 2002, Lisbon, PT.

[15] Shishkov, B. and J.L.G. Dietz. Deriving Use Cases from Business Processes, the Advantages of DEMO. Proc: 5th Int. Conference on Enterprise Information Systems (ICEIS) 2003, Angers, FR.

[16] Shishkov, B.; Z. Xie; K. Liu; J.L.G. Dietz. Identifying Generic Business Processes Using Semantic Analysis. Proc. of the 6th Workshop On Organizational Semiotics, 2003, Reading, UK.

[17] Szyperski, C. Component Software: beyond Object-Oriented Programming. ACM, 1998.