ALIGNING BUSINESS PROCESS MODELING AND SOFTWARE SPECIFICATION IN A COMPONENT-BASED WAY

The advantages of SDBC

Boris Shishkov, Jan L.G. Dietz
Faculty of El. Engineering, Mathematics and Comp. Science, Delft University of Technology,The Netherlands

Keywords: SDBC; Business modeling; Software specification; Component-based alignment

Abstract: This paper introduces the SDBC approach. It aligns business process modeling and software specification. They are aligned in a component-based way. In particular, business components are identified and reflected in the software specification. The business components are identified considering the rich semantic and language/communicative expressiveness of current business systems. This is claimed to have a definite value in building a complete and realistic business model. The component-based alignment on the other hand is claimed to have sound theoretical background and promising practical perspective.

1 INTRODUCTION

The current software development practices need improvement. This is shown by a number of examples of software project failures (Liu, 2000), most of which clearly indicate mismatch between stated business requirements and the actual functionality of a delivered software application. Therefore, it can be claimed that the software development methods, widely used today, do not adequately bridge the consideration of the target business system and the specification of the corresponding software artefact(s).

Besides the alignment itself, it is considered useful also improving the business modeling by addressing not only the business system's structure and dynamics but also semantic/language aspects. This is motivated by the fact that great deal of the activities within any organization are about coordination/communication (as opposed to production). For example, considering shoe repairing, the repair of a shoe brought by a client, is just one issue within the set of activities actually taking place. First the client explains his/her request – what exactly (s)he needs to be done. The shoemaker might accept the request or might not accept it. It is possible that further negotiations are needed about the request. Also, after the shoemaker has repaired client's shoe(s), it is possible that the client does not accept the work, claiming that the repair is of low quality, and so on. Hence, overlooking the communicative/coordination aspects would lead to building an incomplete business model.

These mentioned issues are approached through two promising research perspectives: the Component-Based (system) Development (CBD) as a background for aligning business modeling and software specification, and the Language-Action Perspective (LAP) as a theory that contributes to properly grasping the communicative aspects characterizing a business system.

CBD and LAP are discussed in (Shishkov & Dietz, 2004). They are of essential importance for the proposed SDBC approach (SDBC stands for Software Derived from Business Components). It aligns business process modeling and software specification in a component based way and considers not only the structural and dynamic aspects but also the communicative aspects of the target business system. Some issues which are foundational for SDBC have already been introduced (Shishkov & Dietz, 2004).

This paper elaborates on the SDBC, positioning it within the software engineering task (Section 2), and concludes about some strengths of SDBC compared to other existing methods (Section 3).

Shishkov B. and L.G. Dietz J. (2004).
ALIGNING BUSINESS PROCESS MODELING AND SOFTWARE SPECIFICATION IN A COMPONENT-BASED WAY - The advantages of SDBC.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 103-108
DOI: 10.5220/0002600001030108

2 THE SDBC APPROACH

The development of SDBC has been motivated by the following definite conclusions (Shishkov & Dietz, 2004): 1>. It is necessary to consider the semantic/communicative aspects in order to build a sound and complete business model. 2>. Further studies are required towards aligning business modeling and software specification. 3>. Considering business and software systems as well as their alignment, it is worthwhile applying the principles of CBD, benefiting in this way from the undisputable advantages of object-orientation, among which: re-usability, modifiability, design flexibility.

On this basis, it has been considered crucial that:

SDBC allow for grasping the semantic and language business systems expressiveness, and also aligning business process modeling and software specification in a component-based way.

SDBC is positioned as a software specification approach within the software engineering task (IEEE-Std.'610' 1990). This is illustrated on Figure 1. Going as deep as the specification is claimed to be sufficient for SDBC to properly place the software model under development on a sound business modeling background.

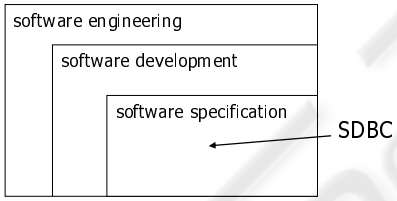

Figure 1: The positioning of SDBC within the software engineering task

Being an approach for specifying software on the basis of business modeling, SDBC is based on: 1) Integrated view over business modeling and software specification; 2) LAP – founded Transaction concept; 3) Alignment based on components; 4) Re-use requirements. These four ***fundaments*** are elicited further on in this section.

INTEGRATED VIEW OVER BUSINESS MODELING AND SOFTWARE SPECIFICATION

The current software development practices are characterized by lack of sound alignment between business modeling and software specification. Small software companies usually rely on an arsenal of "know-how". They try to adapt it to the user's case. Bigger companies, however, spend more time and energy for getting insight about the target business system. Anyway, the process of getting such an insight plays just a supportive role for the system specification, without being integrated with it. A value of SDBC is that it integrates business modeling activities and software specification ones (as shown on Figure 2).

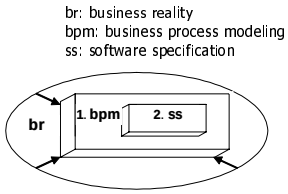

Figure 2: Integrated view over business process modeling and software specification

A business process model is to be built up from the studied business reality. This model is to be reflected in the derivation of a software specification model. This would be the viable link that should

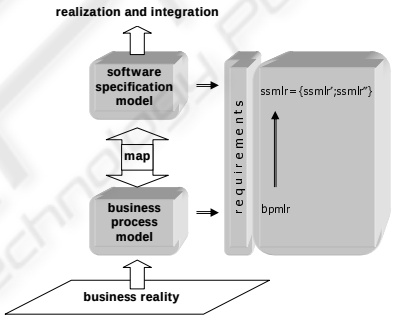

Figure 3: Specifying requirements

guarantee the proper consideration of the original business requirements in the specification of the software's functionality. The derived software specification model should further undergo realization and integration, in building the software system. All this is illustrated in Figure 3. Hence, a crucial issue in this regard is the proper mapping between the two models. Next to that, both the business process model and the software specification model serve as sources for **extraction of requirements**.

The built business model should represent a source for completely discovering the so called bpmlr (bpmlr stands for business process modeling level requirements). The bpmlr characterize the system reflected in the business process model and do not directly refer to the functionality of the software application to be developed. Extracting a particular sub-set of bpmlr, in particular those requirements which concern the functionality of the

software application to be developed and adding to them the requirements additionally discovered throughout the software specification phase, we would come to the ssmlr (ssmlr stands for software specification model level requirements). The ssmlr′ (ssmlr″) stands for the first (second) group of requirements mentioned in the current paragraph. Therefore, in general, the following statement should be true:

$$ssmlr' \subseteq bpmlr,$$

although it is considered possible that a requirement belonging to the ssmlr′ set might appear to be a refined version (not an exact copy) of a requirement belonging to the bpmlr set.

LAP-FOUNDED TRANSACTION CONCEPT

As mentioned before, the essential SDBC goal of grasping the rich semantic and language expressiveness of a considered business system could be accomplished by founding the business system study on LAP. DEMO (www.demo.nl) offers a useful interpretation of some essential aspects of this theoretical orientation (Shishkov & Dietz, 2004). Valuable in this regard is the relation of DEMO also to two other sound and relevant to the mentioned goal theories, namely Organizational Semiotics (Liu, 2000) and Philosophical Ontology (Bunge, 1979). The LAP Transaction and, in particular, the DEMO interpretation of it, is adapted and adopted within SDBC as an elementary business modeling unit. Considering any structural of Transactions (a starting transaction and a tree of transactions triggered by it), a Business process is defined as: *the set of transactions realized in order to fulfill a starting transaction*. A Business component is a complete business process model, as specified in (Shishkov & Dietz, 2004). In this way (through the Business component concept) SDBC allows for placing a specification of software on a business model that is elaborated also in terms

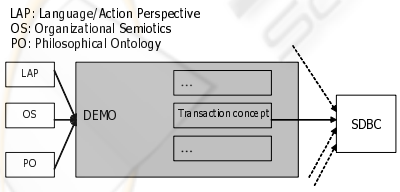

Figure 4: The importance of the transaction concept for SDBC

of communicative action issues, not only in terms of structural and dynamic issues what is the case with the current popular business modeling methods. The relation of SDBC (realized through the Business component concept) to the mentioned semantic and language related theories is illustrated in Figure 4.

SDBC interprets the Transaction concept as centered around a particular Production fact (following the DEMO Production fact definition). The reason is that the actual output of any business system represents a set of Production facts related to each other. They actually bring about the useful value of the business operation to the outside world and the issues connected with their creation are to be properly modeled in terms of structure and dynamics, as in the popular current business modeling methods. However, the already justified necessity of considering also the corresponding semantic and language aspects is important. Although they are not directly related to the Production facts, they are to be positioned around them. As stated already, SDBC realizes this through its interpretation of the Transaction concept, as depicted in Figure 5.

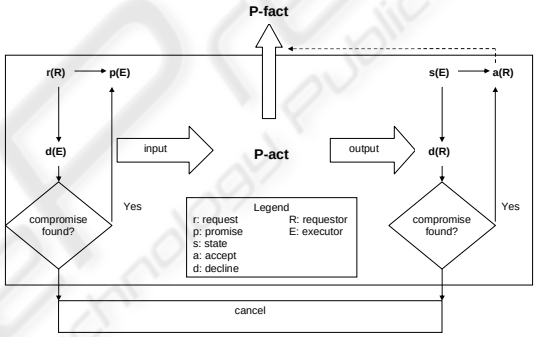

Figure 5: The SDBC interpretation of the transaction concept

As seen from the figure, the classical LAP-DEMO Transaction concept has been adopted, with a particular stress on the transaction's output – the Production fact. There is a proposition (for example, a shoe to be repaired by a particular date and at a particular price, and so on). First, anybody should express a request towards the particular proposition (we call him a Requestor). Such a request might trigger either acceptance or rejection – the other party (we call him an Executor) might either promise to produce the requested product (or service) or decline it. A decline actually triggers a discussion (negotiation), for example: "I cannot repair the shoe today, is tomorrow fine? … and so on". The discussion might result (or might not result) in a compromise. A compromise means that the Executor promises to produce an updated version of the discussed proposition. After a promise, a Production act takes place and afterwards – a statement from the Executor that the requested product or service is produced. Then, analogously, the Requestor might accept (or might not accept) the

production result; a discussion (negotiation) might take place. The transaction is considered to be successful and the corresponding Production fact could be considered successfully created only if the Requestor has accepted the production result (this is indicated by the dashed line on the figure).

Based on the LAP-DEMO theory and its SDBC interpretation, it might be concluded that applying the Transaction concept, SDBC achieves: a realistic position to the issues belonging to a business system; a granularity level (considered) which is the right one as long as the atomic business (process) issues are of interest; a sound theoretical justification.

The Transaction concept directly relates to the Business component one which is fundamental for the SDBC approach and is considered in the following paragraphs.

ALIGNMENT BASED ON COMPONENTS

The perspective of realizing the alignment between the two significant SDBC tasks (business modeling and software specification) on the basis of components is a crucial issue within the SDBC approach. The proposed component-based alignment has been justified (Shishkov & Dietz, 2004) by the undisputable and well proven in practice advantages of the object-orientation theory – a sound theory that allows for representing any system (a software/business one, for instance) in terms of objects/components. Thus, identifying business components and reflecting them in (sets of) software components would be well founded theoretically. Next to that, components could be re-used. As it is well-known, re-use is an essential advantage for any system development method. Re-use will be discussed further on, as a foundational issue within the SDBC approach. The component-based alignment between business modeling and software specification is illustrated in Figure 6.

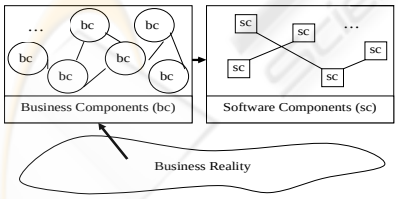

Figure 6: From business components to software specification

As depicted on the figure, the target business reality is to be reflected in a set of identified business components (these are business process models as already mentioned). Based on them, a (component-based) software model is to be specified. The business and software components are not to be necessarily mapped one to one. The bottom line in building up the business process model (through the identification of business components) should be a purely business-oriented study that has nothing to do with the specification of software and related issues. On the other hand, the software specification (and integration), though based on the business components, is to be realized from the perspective of the functionality of the software system under development. Thus, it is possible that more than one software components are derived based on a business component, for instance.

Hence, following the principles of component-based system development brings all the advantages associated with this type of system development to both the business modeling phase and the software specification one. The component-based perspective makes it doable to easily trace a relation between a designed (or even implemented) software component and its originating business process model. Adding corrections and/or modifications further on would be easy as well. Even entire models/components could be inherited and developed in a new context, for example. Therefore, the mapping itself (between business modeling and software specification) could be significantly facilitated. Next to that, re-use would be possible and easily realizable at different levels. A (specified) software component could be used for the development of different software artefacts as this is successfully done currently. Also the business components could be re-used – if some (business) requirements change, it would be possible to replace a business component with another one without affecting the entire model of the business system.

RE-USE REQUIREMENTS

As mentioned before, the re-use options are essential for SDBC. The approach benefits from them both in the business modeling phase and in the software specification one. SDBC allows for reuse of business processes – if they are abstractly described and also for reuse of components (business and software components). This is depicted in Figure 7.

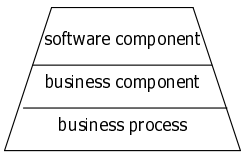

Figure 7 : Levels of re-use

Re-using a Business process within SDBC includes describing a business process at a general level, making such a description abstract enough so that it could be applied in a number of cases. An

example: ARRANGE A SERVICE (this is the transaction which is the starting one); PAYMENT; REDUCTION APPROVAL… This general description could easily be made more specific, for example: ARRANGE A HOTEL RESERVATION SERVICE; DEPOSIT PAYMENT; EARLY BOOKING REDUCTION OFFER…

Re-using a Software component is considered to be also an option within SDBC, especially in tune with the latest software re-use practices associated with distributed computing environments like EJB, CORBA, and .NET (Atkinson & Muthig, 2002).

As long as software components and their re-usability are concerned, the existing knowledge (Atkinson & Muthig, 2002), is considered to be sufficient. It is reflected in developing SDBC.

The component-based software development differs from traditional approaches by splitting the development process into two distinct activities:

- Development for reuse – creating high-quality, specialized components which concentrate on doing a specific job well; they should be of use in multiple applications.
- Development with reuse (called also "Integration") – creating new applications (or possible larger components) by assembling prefabricated components.

Re-using a Business component is of significant importance for SDBC and the approach introduces ideas in this regard.

If we consider the building up of a system (in general) out of some building blocks and want to re-use some of them, we have two solution directions – we could have either a core unit that we should build further on, in order to specify a building block, or we could have a "multiple function" unit that we should adjust in order to create a particular building block. An analogy for the first case is a wagon platform – it could be further developed either into a passenger wagon or into a cargo one. An analogy for the second case is a universal plug adaptor – it has (in it) a number of functionalities that might be adjusted in one way or another depending on the particular purpose of use (for example: use in Europe, USA, or Japan).

Following the classical Object-orientation terminology, it is suggested that the first (second) of the mentioned types of re-usable units is called a **general** (**generic**) unit. Returning to the system that is made up of building blocks, we could, therefore, distinguish between two types of re-usable units: General building blocks and Generic building blocks. This is illustrated on Figure 8.

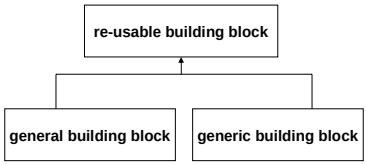

Figure 8: Re –usable units.

These basic principles could be applied to the Business component concept within the SDBC framework, bringing about possibilities of re-using Business components. If general or generic business components are identified, they could be re-used in the specification of different software artefacts; this could be realized either by extending a general business component or by parameterizing a generic one, as depicted in Figure 9.

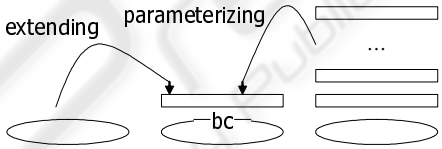

Figure 9: Extending a general business component or parameterizing a generic one.

General business components (bc) are models which reflect core issues and can be extended in a number of directions. 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 reservations system. Hence, the particular extension of a general business component is motivated by the purpose of use. On the contrary, a generic business component should contain in itself several optional functionalities. Through parameterization, such a component could be adjusted depending on the aimed purpose of use.

In summary, it is possible within SDBC, to derive a business component by developing a model of a business process (the trivial way) or by re-using general/generic business components (Figure 10).

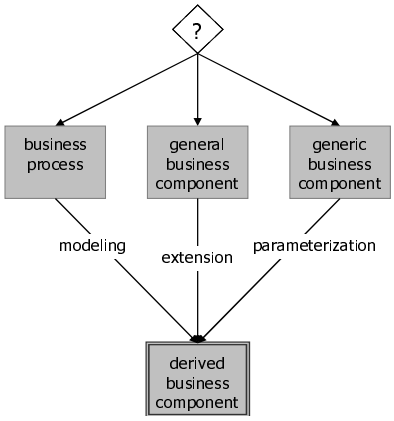

Figure 10: Deriving a business component

OUTLINE OF SDBC

Since the essential goal of the current paper is to elaborate on the foundational issues behind the SDBC approach and due to the limited scope of the paper, the information on the outline of the approach is omitted. However, this information is to be found in (Shishkov & Dietz, 2004).

3 ADVANTAGES OF SDBC

SDBC's innovative features relate primarily to its allowing for a more complete business modeling (in comparison to the widely used business modeling tools) and especially for alignment between this modeling and the specification of software. The allowance for a component-based alignment between the mentioned two aspects is a definite value of the SDBC approach.

In order to justify the advantageous features of SDBC, it is essential to approach through it real-life problems and also compare SDBC to other relevant modeling tools. The fundamental SDBC aspects have been validated through case studies and some of this information has been reported to (Shishkov & Dietz, 2004). As for the comparison with other tools, it is of no use considering either pure business modeling tools or pure software design ones because the essential goal behind SDBC is the alignment between business modeling and software specification. Hence, only tools that relate to these both aspects have been considered. Among them, KobrA (Atkinson & Muthig, 2002), Catalysis (www.catalysis.org), and Tropos (www.troposproject.org) are much popular and successfully applied in practice.

Among the key characteristics of **KobrA** are: architecture-centricity; systematic COTS component re-use; integrated quality assurance. The major strengths of KobrA are its overall consistency, the embracement of the component concept in all phases of the software life-cycle, and the UML-based graphical specification of components. The main limitation is that there are no clear guidelines how to relate the specification of software to prior business studies. **Catalysis** provides well-defined consistency rules across models and powerful mechanisms for composing different views to describe complex systems. However, the method does not offer a solid mechanism for reflecting the (business) requirements in the specification of the software's functionality. **Tropos** approaches the requirements elicitation and specification in a sound way, offering mechanisms for transformation of this output into an input for the further software design phases. However, this transformation is neither sufficiently formal nor completely founded on object-orientation.

Therefore, the current popular methods (including the most significant ones among which are KobrA, Catalysis, and Tropos) related to the considered research problem, have particular limitations either in the pre-specification phase or in the specification one. These limitations are an obstacle for soundly aligning business modeling and software specification.

4 CONCLUSION

The SDBC approach is based on the innovative idea of aligning business modeling and software specification in a component-based way. Next to that, SDBC offers particular improvements in the business modeling task, associated mainly with the consideration of the communicative aspect in approaching a business system.

REFERENCES

Atkinson, C. and D.Muthig, 2002. Enhancing component reusability through product line technology. In *ICSR'02, 7th Int. Conf. on Software Reuse*.

Bunge, M.A., 1979. *Treatise on Basic Philosophy*, Vol. 4, Reidel Publishing Company, Dordrecht.

Liu, K., 2000. *Semiotics in Information Systems Engineering*, Cambridge University Press.

Shishkov, B. and J.L.G. Dietz, 2004. Design of software applications using generic business components. In *HICSS'04, 37th Hawaii International Conference on System Sciences*. IEEE'04.