APPLYING COMPONENT-BASED UML-DRIVEN CONCEPTUAL MODELING IN SDBC

Boris Shishkov
Department of Computer Science, University of Twente, 5 Drienerlolaan, Enschede, The Netherlands

Jan L.G. Dietz
Department of Software Technology, Delft University of Technology, 4 Mekelweg, Delft, The Netherlands

Keywords: Software specification; Component; SDBC; UML

Abstract: Properly aligning business process modelling and software specification is crucial for correctly and completely reflecting the business requirements in the design of a software system. Realizing a component-based alignment between these two aspects seems adequate. The SDBC approach possesses a conceptual framework, complemented by step-by-step methodological application guidelines, on how to accomplish this. However, the SDBC framework is yet insufficiently elaborated in terms of (theoretically rooted) concepts/definitions, which could be an obstacle for relating further SDBC to other relevant modelling tools. Aiming at overcoming this, we propose in the current paper some theoretically rooted concepts which are relevant to the approach and are as well useful regarding its application.

1 INTRODUCTION

The current Information and Communication Technology (ICT) is of significant importance for the proper flow of processes belonging to a number of business domains. Software (ICT) applications are to facilitate the utilization of ICT for the mentioned purpose. Thus, more and more research takes place on application development methodologies, and also more and more industrial projects appear related to this issue. However, often such projects are characterized by unrealized goals, low user satisfaction, and increasing budgets.

It is claimed that one frequent cause of software project failure is the mismatch between the business requirements and the actual functionality of the delivered application. This problem relates to the misconception that a business system (process) is a kind of information system (process). Instead, they are systems in different categories: social and rational, respectively. Therefore, in order to adequately reflect the requirements in the software system-to-be, one needs to soundly align the business process modelling and software specification, mapping a pure business-oriented model towards the specification of a software system.

Realizing such an alignment in a component-based way seems feasible and beneficial because component-based business and software models would allow for re-use, and also for good modelling traceability, ease of modifiability and flexible maintainability (Shishkov & Dietz, 2004-2). Next to that, a business component would concern business process modelling concepts while a software component would concern software concepts; hence, a rigorous mapping between the two would be a good foundation for a business-software alignment.

However, to date the component paradigm has only really penetrated the implementation and deployment phases of the software life-cycle, and does not yet play a major role in the earlier analysis and design activities of large software projects. In the software context, components are associated mainly with the current 'physical' component technologies, such as .NET, CORBA, and EJB.

Although some approaches, such as OMG MDA, ODESSA, INSPIRE, and COMET, aim at overcoming this, they have not completely succeeded yet, as it is well-known. Hence, the software community still misses modelling facilities for adequately addressing the component paradigm in the mentioned analysis and design phases, concerning the business-software-alignment context.

Shishkov B. and L.G. Dietz J. (2005).
APPLYING COMPONENT-BASED UML-DRIVEN CONCEPTUAL MODELING IN SDBC.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 417-420
DOI: 10.5220/0002519004170420

This motivates the necessity to propose new (component-based) modelling solutions related to the mentioned phases and context.

The SDBC (SDBC stands for Software Derived from Business Components) approach has been introduced (Shishkov & Dietz, 2004-1; Shishkov & Dietz, 2004-2), which is capable of adequately addressing these issues by considering 'logical' components that represent the logical building blocks of a software system. From this position, SDBC proposes a mechanism for business-software alignment. In particular, the approach allows for deriving pure business process models (called *business coMponents*) and reflecting them in conceptual (UML-driven) software specification models (called *software coMponents*). In the business coMponent identification, SDBC follows a multi-aspect business perspective, guaranteeing completeness. In the business coMponent – software coMponent mapping, SDBC follows rigorous rules, guaranteeing adequate alignment. Being UML-driven, SDBC is in tune with the latest software design standards. The application of SDBC is currently explored in a large Dutch insurance company, and also through several test case studies.

This paper reports further SDBC-related studies. In particular, it proposes several concepts/definitions which are relevant to the SDBC approach, discussing as well their usability with respect to its application.

The outline of the paper is as follows: Section 2 suggests several concepts fundamentally important for the SDBC approach. Section 3 provides elicitation on their usability in applying the approach. And finally, Section 4 contains the conclusions.

2 ESSENTIAL CONCEPTS

As mentioned in the Introduction, this section is to propose some fundamental SDBC-related concepts.

A system consideration would be needed first, taking into account that in any (scientific) discipline, particular kinds of systems need to be studied. Concerning SDBC and in particular the need to align business process modelling and software specification, a consideration of two types of systems would be required, namely business systems and information systems. A clear delimitation between the two is considered necessary, mainly because of the (observed) misconception that a business system is a kind of information system. Instead, as already mentioned, they should be considered in different ways. Although they both are basically social systems, they differ in the kind of production: business services and (internal) information services, respectively (Dietz, 2003).

Before defining business system and information system, we would have to propose our system definition, adopted from the 'classical' system definition of Bunge (Bunge, 1979):

Definition 1. Let T be a nonempty set. Then the ordered triple $\sigma = <C, E, S>$ is system over T if and only if C (standing for *composition*) and E (standing for *environment*) are mutually disjoint subsets of T (i.e. $C \cap E = \varnothing$), and S (standing for *structure*) is a nonempty set of active relations on the union of C and E. The system is *conceptual* if T is a set of conceptual items, and *concrete* (or material) if $T \subseteq \Theta$ is a set of concrete entities, i.e. things.

Taking into account that, considering business and software issues, SDBC approaches business activities as realized by humans, and based on Definition 1, we suggest the following business system definition.

Definition 2. A system should be considered being a *business system* if and only if it is composed of physical persons (humans) collaborating among each other through actions which are driven by the goal of delivering business products to entities belonging to the environment of the system.

As for the information system concept, it should be considered not only in an ontological but also in a functional perspective, because the functional aspect is essential concerning the way in which an information system supports (informationally) a business system. Thus, we will propose an ontological as well as a functional definition of information system.

The ontological information system concept should correspond to our viewing information systems as composed of humans facilitated by (ICT) applications, who collaborate in realizing internal informational support to interorganizational processes. Based on these considerations as well as on Definition 1, we propose the following definition:

Definition 3. A system should be considered being an *information system* if and only if it is composed of humans (often facilitated by ICT applications as well as technical and technological facilities) collaborating among each other driven by the goal of supporting informationally a corresponding business system. Usually the business system and the information system belong to the same organization.

The functional information system concept should correspond to the basic (well-known) functions characterizing a (current) technological support: related to data being created, processed, distributed, and so on. For this reason, we have adopted the following definition (Simon, 1996):

Definition 4. Concerning its functional characteristics, an *information system* is a system which manipulates data and normally serves to collect, store, process and exchange or distribute data to users within or between enterprises or to people within wider society.

Based on the essential definitions set out above, we proceed with introducing the business/software coMponent concepts, starting with a consideration of business coMponents, based on certain related concepts presented below.

Adopting the DEMO *transaction concept* (Dietz, 2003), we define a *business process* as being a structure of (connected) transactions that are executed in order to fulfil a starting transaction.

Further on, considering a *business sub-system* as being a system which is a part of a business system, we propose the following *business component* definition:

Definition 5. A *business component* is a business sub-system that comprises exactly one business process.

Considering SDBC, the theories behind it (Shishkov & Dietz, 2004-2), and their way of viewing a model, we regard a *complete model* as a model that is elaborated at least in four perspectives (Shishkov & Dietz, 2004-1), namely structural, dynamic, factual, and communicative. Hence, we propose the following definition for business coMponent:

Definition 6. A *business coMponent* is a complete model of a business component.

On the basis of these (introduced above) concepts an also on our viewing an ICT application as an implemented software product which realizes a particular functionality supporting in this way the humans who are elements of the composition of an information system, we envision the relation between a business coMponent and an information system as shown in Figure 1.

As seen from the Figure, within SDBC, business coMponents could be used to support the specification of (ICT) applications, being themselves identified based on corresponding business components characterizing the (originally considered) business system. This is how an application which is intended to support informationally a business system is specified being soundly and methodologically rooted in a relevant business process model.

From this perspective, a relevant ontological software component definition (Shishkov & Dietz, 2004-2) is:

Definition 7. *Software components* are implemented pieces of software, which represent parts of an ICT application, and which collaborate among each other driven by the goal of realizing the functionality of the application.

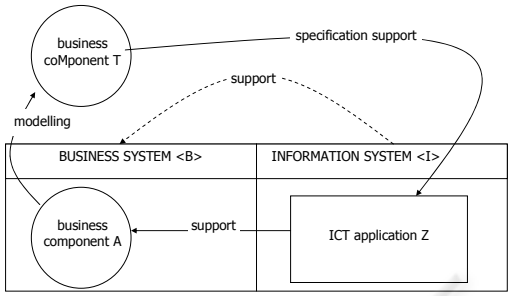

Figure 1: Business coMponents' supporting the applications' specification

Since the software component concept concerns the implementation phase, we would need to propose also a functional definition:

Definition 8. Functionally, a *software component* is a part of an ICT application, possessing a clearly defined function and interface to the other parts of the application.

Since any support from a business coMponent would concern the specification phase, we should (thus) consider another relevant concept. Such a concept must refer to the logical application building blocks (mentioned before). We introduce the term 'software coMponent' to reflect the logical aspect:

Definition 9. A *software coMponent* is a conceptual specification model of a software component.

In summary, we have defined (in this section) some concepts having a fundamental relevance to the SDBC approach and its application.

In the following section, we will relate these concept so the application of the approach.

3 THE CONCEPTS AND SDBC

Aiming at adequately relating the concepts, introduced in the previous section, to the SDBC approach and its application, we will briefly summarize the outline of SDBC and elaborate on the usefulness (in this context) of the mentioned concepts. We will realize this with the help of Figure 2. There we have used the following abbreviations: bc – business component; bk – business coMponent; glbk – general business coMponent; gcbk – generic business coMponent; ssm – software specification model; sc – software component; sk – software coMponent. For more information on SDBC and also on those of the above terms which have not been explained in the previous section, interested readers are referred to (Shishkov & Dietz, 2004-1; Shishkov & Dietz, 2004-2).

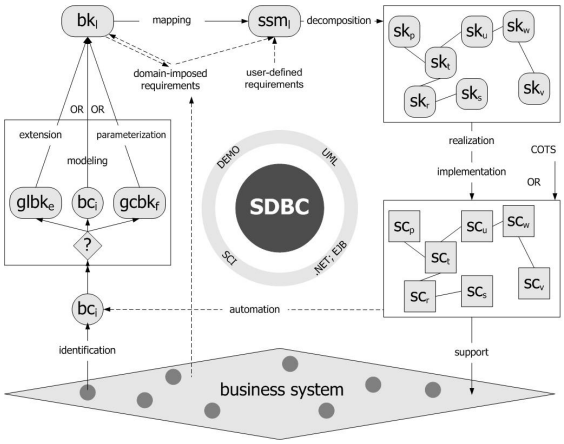

Figure 2: SDBC – outline

As seen from the Figure, we consider a particular business system from which a business component(s) is to be identified, using for this purpose the SCI modelling technique (Shishkov & Dietz, 2004-2) and other useful (modelling) tools. The component should be then reflected in a relevant model – a business coMponent. Another way for arriving at a business coMponent is by applying re-use: either extending a general business coMponent or parameterizing a generic business coMponent. DEMO and other related modelling tools are relevant as far as business coMponents are concerned. The business coMponent should be then elaborated with the domain-imposed requirements, for the purpose of adding elicitation on the particular context in which its corresponding business component exists within the business system (from which it has been identified). Then, a mapping towards a software specification model should take place, driven by the DEMO-UML transformation (Shishkov & Dietz, 2004-3). The mentioned requirements as well as the user-defined requirements are to be considered here, since the derived software model should reflect not only the original business features but also the particular user demands towards the software system-to-be. The (UML-based) software specification model would need then a precise elaboration so that it provides sufficient elicitation in terms of structure, dynamics, data, and collaboration. It needs also to be decomposed into a number of software coMponents reflecting functionality pieces. These coMponents are then to undergo realization and implementation, being reflected (in this way) in software components. This final set of components might consist of such components which are implemented (using software component technologies, such as .NET or EJB, for instance) based on corresponding software coMponents and such components which are purchased. Finally, the (resulting) component-based application would support informationally the target business system, by automating anything that concerns the considered business component (identified from the mentioned system).

4 CONCLUSIONS

This paper has reported further SDBC-related studies, by proposing relevant concepts and providing elicitation on their usability in applying the SDBC approach.

The paper has brought additional evidence on the values of SDBC, following other published results (Shishkov & Dietz, 2004-1; Shishkov & Dietz, 2004-2). All these reported results are supported by case studies such as a case carried out in a large Dutch insurance company and a cultural-heritage-related case considered in another paper within the current Proceedings. This inspires the authors to go forward in further developing the conceptual and application potentials of the SDBC approach.

A significant distinguishable value of the approach is its being capable of adequately aligning in a component-based way business process modelling and software specification.

REFERENCES

Bunge, M.A. Treatise on basic philosophy, Vol. 4: Ontology II: A world of systems. Reidel Publishing Company, Dordrecht, 1979.

Dietz, J.L.G. The atoms, molecules, and fibers of organizations. *Data & Knowledge Engineering*, vol. 47, pp 301-325, 2003.

Shishkov, B. and J.L.G. Dietz, 2004-1. Aligning business process modelling and software specification in a component-based way, the advantages of SDBC. In *ICEIS'04, 6th International Conference on Enterprise Information Systems*. ICEIS Press.

Shishkov, B. and J.L.G. Dietz, 2004-2. Design of software applications using generic business components. In *HICSS'04, 37th Hawaii International Conference on System Sciences*. IEEE Computer Society Press.

Shishkov, B. and J.L.G. Dietz, 2004-3. Deriving use cases from business processes, the advantages of DEMO. Enterprise Information Systems V, Edited by O. Camp, JB.L. Filipe, S. Hammoudi, and M. Piattini, Kluwer Academic Publishers, Dordrecht/Boston/London, 2004.

Simon, E. Distributed information systems: From client/server to distributed multimedia. McGraw-Hill International Ltd., 1996