BRIDGING THE LANGUAGE-ACTION PERSPECTIVE AND ORGANIZATIONAL SEMIOTICS 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

Kecheng Liu
Informatics Research Centre, The University of Reading, Whiteknights, Reading, UK

Keywords: Business process modelling, Language-Action Perspective (LAP), Organizational Semiotics (OS), SDBC.

Abstract: Achieving an adequate business-software alignment should include capturing essential aspects that concern the studied business reality as well as reflecting them in the software specification. This concerns semantic and pragmatic aspects, such as meanings, intentions, negotiations, commitments, and obligations, whose appropriate consideration is beyond the capabilities of current software design methods which focus just on formal language structures and data records. Organizational Semiotics allows for an adequate consideration of essential semantic aspects in conducting a business process modeling. The Language-Action Perspective is capable of grasping pragmatics on top of that. Some examples of combining LAP and OS have demonstrated benefits when conducting business process modeling, for the purpose of further specification of software. To further this work, it is necessary that LAP and OS are incorporated in a modeling framework which integrates business process modeling and software specification. The SDBC approach applies the LAP and OS theories, in order to address the business-software alignment challenge. They support the business process modeling in SDBC, which results in the identification of re-usable business process models and their mapping to software specification, in consistency with the current software design standards. Thus, it is feasible to expect that a LAP-OS combination could bring value to the current application development, in the context of the approach SDBC. In this paper, we further elaborate our SDBC-driven views on how LAP and OS can be combined for a sound business process modeling which builds a foundation for software specification.

1 INTRODUCTION

eBusiness, Tele-Work, telecenters, and other examples of utilizing the latest information technology, alongside the current development of global telecommunications and digital multimedia, are considered to indicate for a just partial success in Society's making use of this emerging technology (Shishkov, 2005). We argue that one of the reasons for this is the limited capability of (software) applications in addressing the role they are to play: to serve as an environment through which users benefit from particular technical possibilities. This limited capability represents an actual challenge which can be seen from numerous examples of mismatch between the requirements and the actual functionality of the delivered application (Shishkov & Dietz, 2004-1). Such examples are evidence that current application development fails in providing users with adequate technical support in their business activities. Not surprisingly, an increasing number of software projects fail (Liu, 2000).

An essential issue in developing applications, as studied by Shishkov and Dietz (2005), is that the modeling of the business processes (to be supported by applications) and the specification of those applications need to be considered as one integrated task. Such an approach assist us in avoiding common problems of designing software without prior adequate consideration of the business processes to be supported by it, or providing a business process modeling output which is inadequately transformable into the specification of software.

Nevertheless, achieving an adequate business-software alignment would mean that all essential aspects concerning the studied business reality are captured and reflected in the software specification. This should include semantic and pragmatic aspects, such as meanings, intentions, negotiations, commitments, and obligations (Stamper, 2000). Current software design methods often focus on formal language structures and data records, which makes them incapable of adequately handling semantics and pragmatics. The limitations of such methods have become more apparent with the increasing penetration of technology and software into social and economic activities.

Organizational Semiotics (OS) allows for an adequate consideration of essential semantic aspects, in conducting a business process modeling (Stamper, 2000; Liu, 2000). The Language-Action Perspective (LAP) is capable of grasping pragmatics on top of that (Dietz, 2004). Some examples of combining LAP and OS have demonstrated benefits when conducting business process modeling, prior to a specification of software. To further this work, it is necessary that LAP and OS are incorporated in a modeling framework which integrates business process modeling and software specification.

The SDBC ('SDBC' stands for Software Derived from Business Components) approach (Shishkov, 2005; Shishkov & Dietz, 2005) addresses the challenge of business-software alignment, relying on the LAP and OS theories. They support the business process modeling in SDBC, which results in the identification of re-usable business process models and their mapping to software specification, in consistency with the current software design standards, such as the Unified Modeling Language (Shishkov, 2005; Shishkov & Dietz, 2004-2; OMG, 2003). Thus, it is feasible to expect that a LAP-OS combination could bring value to the current application development, in the context of the SDBC approach.

In this paper, we further elaborate our SDBC-driven views on how LAP and OS can be combined for a sound business process modeling which builds a foundation for software specification.

The outline of the paper is as follows: Section 2 briefly introduces LAP and OS, and analyzes in general the value of and potentials for their joint application. Section 3 presents the SDBC approach and also elaborates on related aspects that concern the issue on bridging LAP and OS. This is illustrated in Section 4, by means of an example. Section 5 supports analytically the current study, by discussing relevant related work, with Section 6 to present conclusions.

2 LAP AND OS

As stated already, this section will briefly introduce LAP and OS, and then present some general considerations regarding their possible combination.

2.1 Language-Action Perspective

We observe that people demand currently more and more from business processes. For example, it is often required that such processes are modified and/or connected to other business processes (Shishkov, 2005). These increasing demands towards business processes cannot be met adequately anymore by relying only on best practices (Dietz, 2003). A promising perspective, convincingly claiming to soundly approach the modeling of business processes is the Language-Action Perspective (LAP). LAP is a theoretically based approach towards modeling and developing business processes and information systems, that has its roots in language philosophy. The approach recognizes that language is not only used for exchanging information, as in reports, for example, but also to perform actions, as in promises or orders. Such actions are claimed to represent the foundation of communities and organizations, and must be understood to create effective information systems.

The seminal contributions to the development of LAP are one European (Goldkuhl & Lyytinen, 1982) and one American (Winograd & Flores, 1986) publication. In LAP, an organization is considered as a social system. This means that the inter subject relationships among people, brought about and maintained in communication, constitute the real basis of an organization's existence. Business processes then become structures of commitments and the real important activity of the actors in these processes is that they enter into and comply with commitments.

Among other things, LAP has given rise to the notion of Enterprise Ontology, as proposed in (Dietz, 2006). This notion has appeared on top of some LAP-driven considerations which concern the methodology DEMO (Dietz, 2003; Shishkov & Dietz, 2004-3).

2.2 Organizational Semiotics

Considering solely the concept of *information* would not allow for grasping completely all social relationships attached to a business concept (Liu, 2000). In Organizational Semiotics (OS), it is argued that *signs* (standing to someone for something else, in some respect or capacity) offer a rigorous foundation to understand a business reality (Stamper 2000). For example, a bank note is more than just a piece of paper with digits on it. It stands for its holder's wealth, the issuing bank's authority, and so on. This enables for a balanced business study reflecting both technological and social aspects. Further, by adopting a subjectivist philosophical stance, OS acknowledges that nothing exists without a perceiving agent and the agent engaging in actions.

Based on this foundation, Stamper has contributed to the OS development, by adopting from Gibson (1979) the notion of *affordance* – the action possibilities in the environment in relation to the action capabilities of an agent. Illustrating this: in the context of a library, a book affords to be borrowed – this is a potential pattern of behavior. It may or may not be realized in reality. However, once it is realized, new possibilities may emerge. For example, a borrowed book may be returned. Hence, affordances have dependency relationships among them, called *Ontological Dependency*. It could be schematically shown with the antecedents on the left side and the dependencies on the right, and the solid line denotes the ontological dependency:

book – borrow - return

This shows not only the logical inter-concept relationship but also the dependencies getting their meaning from the existence of the antecedents. Since the dependencies result from antecedents' existence, the dependencies' lifecycle is included by that of the antecedents. Their existence thus forms a context for the dependencies. For example, returning a book refers to the fact that it has previously been borrowed.

2.3 The LAP-OS Combination

We argue that LAP and OS are necessarily complementing each other, in allowing for an adequate business process modeling foundation which concerns a realistic reflection of the original business reality. We claim as well that this distinguishes these theories from the ones currently used within the Software Community. The crucial advantage is that a LAP/OS foundation grasps essential semantic and communication issues and allows for their reflection in the specification of an application. We see this as a decisive factor in making the application-to-be adequately operational in the context of the business environment in which it would have to function.

In our discussion on the general foundations of a LAP-OS combination, we refer to the widely considered 'Semiotic Ladder' (Stamper, 2000):

Real-life aspects (**, *****, ******)**
******** S O C I A L W O R L D:**
beliefs, expectations, commitments, contracts, culture, …
******* P R A G M A T I C S:**
intentions, communication, conversations, negotiations, …
****** S E M A N T I C S:**
meanings, propositions, validity, truth, signification, …

IT-platform-related aspects (*, **, *)**
***** S Y N T A C T I C S:**
formal structure, language, logic, data, software, files, …
**** E M P I R I C S:**
pattern, variety, noise, channel capacity, codes …
*** P H Y S I C A L W O R L D:**
signals, traces, physical distinctions, hardware, laws of nature, …

As it is seen above, in a business-process-modeling-driven software specification, we should consider issues that concern the technical aspects of an application (such as empirics and syntactics, for example) as well as issues that concern the real-life aspects (such as semantics and pragmatics, for instance). We argue that, because of the increasing scope of applications in the context of a business system, the real-life aspects must receive greater attention, as a way to overcome the (frequently observed) mismatch between the functionality of an application and the original business requirements. Therefore, we claim that considering semantics, pragmatics, and other real-life aspects is of essential importance. This means that it is necessary to adequately grasp such aspects and also map them properly to the specification and realization of software, where empirics and syntactics do play a role.

Hence, we envision a significant value in OS and LAP because, as it has been discussed in the current section, these theories are powerful with regard to semantics and pragmatics, respectively. Thus, the potentials for combining LAP and OS do concern these real-life aspects. Such a combination would lead, therefore, to a business process modeling output which is adequate in the context of the (considered) business-software-alignment task.

However, a challenge is how LAP and OS could be applied in an integrated way, in tune with the above-presented views.

The following section is going to answer this question, by discussing the approach SDBC in which LAP and OS aspects are integrated, in the course of delivering an adequate business process modeling output to be used as a foundation for specifying software.

3 BRIDGING LAP AND OS IN SDBC

In this section, we will present, on the basis of the general LAP-OS discussion conducted in Section 2, our views on how LAP and OS aspects could be helpfully combined in SDBC. That is why we will first briefly present the SDBC approach in Sub-section 3.1.

3.1 The SDBC Approach

As already mentioned, before discussing the way in which SDBC bridges LAP and OS aspects, we will briefly summarize the outline of the approach; we will realize this using Figure 1. We have used the following abbreviations there: bc – Business Component (a business sub-system that comprises exactly one business process); bk – Business CoMponent (a model of a Business Component, which is elaborated in terms of structure, dynamics, communication, and data); glbk – general Business CoMponent (which is re-usable by extension); gcbk – generic Business CoMponent (which is re-usable by parameterization); ssm – software specification model; sc – Software Component (an implemented piece of software representing a part of an application); sk – Software CoMponent (a conceptual specification model of a Software Component). For more information on the above concepts, interested readers are referred to (Shishkov, 2005; Shishkov & Dietz, 2005).

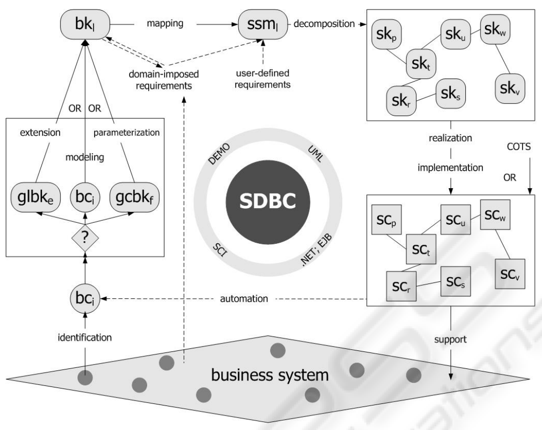

Figure 1: Outline of the SDBC approach.

The figure shows that SDBC is about a business-process-modeling-driven specification of software. The starting point is a consideration of a business system. Business Components are identified from it. This is done by applying an OS-driven analysis leading to the derivation of the so called ‘SCI modeling output’ (Shishkov, 2005). The Business Components should be then reflected in corresponding Business CoMponents, in supplying an adequate modeling foundation for the further software specification activities. Another way of arriving at a Business CoMponent is by applying re-use: either extending a general Business CoMponent or parameterizing a generic Business CoMponent. DEMO (Shishkov & Dietz, 2004-3) and other LAP-driven modeling tools are relevant as far as Business CoMponents are concerned. Each 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 a DEMO-UML transformation mechanism (Shishkov, 2005) and supported by OS. 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 coordination (Shishkov, 2005; Shishkov & Dietz, 2004-2). The model needs also to be decomposed into a number of Software CoMponents reflecting functionality pieces. Then these Software CoMponents are 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 initially considered Business Component(s) identified from the mentioned system.

3.2 Bridging LAP and OS Aspects

The SDBC approach has been briefly presented above. However, for more information on the approach, readers are referred to (Shishkov, 2005). Due to the limited scope of this paper and also because of the availability of information on SDBC as well in other sources (Shishkov & Dietz, 2005; Shishkov & Dietz 2004-1), we will not introduce in detail features of SDBC which we will discuss below in addressing the values of the approach with regard to the challenge of bridging LAP and OS aspects. Figure 2 illustrates these values.

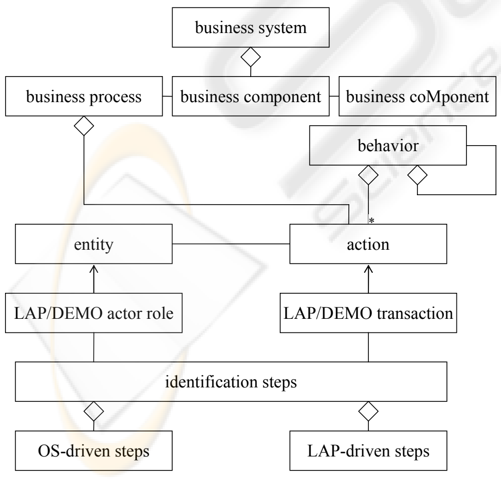

Figure 2: LAP and OS aspects concerning SDBC.

As seen from the figure, LAP/DEMO transactions do represent the elementary business process modeling building blocks in SDBC – actually, we define a Business Process as ‘a structure of connected transactions…’ (Shishkov & Dietz, 2005). In addition to this, taking into account that ‘nothing exists without a perceiving agent engaging in actions’ (Section 2), we do consider in SDBC the LAP/DEMO actor role concept (Shishkov, 2005).

Therefore, a crucial issue about SDBC is the identification of transactions and actor-roles.

The identification mechanisms on which SDBC relies regarding this, are based on LAP and OS. Thus, this is a good starting point for a deeper discussion on the potentials of combining LAP and OS, for the purpose of creating a sound business process modeling foundation in the context of a software specification.

Hence, we will briefly outline the influence of LAP and OS in the mentioned identification mechanisms and then partially illustrate this in the following section.

For this purpose, we use Table 1 where we consider not only business process modeling activities (Figure 2) but also such activities that concern the reflection of a business process model towards software specification. However, the latter are considered to be beyond the scope of this paper because they have been addressed in another one (Shishkov & Dietz, 2004-3).

Table 1: LAP and OS influencing steps in SDBC.

SDBC-Task	Influence
A. Tasks that concern the identification of actor-roles	
- Building a generalization hierarchy for the explored domain	Semantic Analysis (OS)
- Analysis of responsibilities	LAP
B. Tasks that concern the identification of transactions	
- Identification of inter-role relations	LAP
- Elaboration of relations	Norm Analysis (OS)
- Mapping to transactions	LAP
C. Tasks that concern the derivation of a software specification model	
- DEMO – use case mapping	LAP (and also OS)
- use case specification	OS, LAP
- use case elaboration	LAP, OS

As seen from the table, the identification of actor-roles in SDBC comes through building of a generalization hierarchy for the explored domain (which is influenced by OS in general and Semantic Analysis (Stamper, 2000), in particular) and analyzing the responsibilities connected with each of the elements of the hierarchy (which is influenced by LAP). The identification of transactions in SDBC comes through the LAP-driven identification of inter-role relations and also through their normative elaboration (which is OS-driven). Further, the actual modeling of transactions is based on the SDBC (LAP-driven) transaction pattern (Figure 5). And finally, the reflection of the SDBC business process modeling output in the specification of software goes through the identification of use cases. The SDBC use case derivation mechanism (Shishkov & Dietz, 2004-3) is driven by LAP in general and DEMO in particular, and supported by OS. As for the further specification and elaboration of the derived use cases, both LAP and OS add value – OS adds value mainly from the perspective of semantic and normative elaboration while LAP adds value from the perspective of communicative and coordination elaboration.

All this information is available in more detail in the SDBC sources, mainly in (Shishkov, 2005). The following section is going to provide however some relevant illustrative elicitation.

4 CASE EXAMPLE

As stated already, this section will illustrate the influence of LAP and OS, concerning SDBC, and in particular – their combined contribution.

The illustration is founded on a case study, namely the *Icomp Case*. Since the goal is just illustrative and because of the limited scope of this paper, the information to be presented will be partial; for more information on the mentioned case, interested readers are referred to (Shishkov, 2005).

The case concerns the insurance company Icomp. We are particularly interested in a part of Icomp's business, namely the distribution of financial products to end customers through brokers; this means that there are a number of (insurance) financial companies, a number of brokers, and a number of customers, which do concern the mentioned distribution mechanism. A broker distributes products of a number of companies (including Icomp) to a number of customers. A customer might be advised by a number of brokers about the products of a number of (financial) companies. Hence, Icomp uses a number of brokers through which it distributes its products to customers. With respect to the (financial) products distribution, Icomp has relations not only with brokers but also with re-insurance companies (taking over insurance risk from Icomp), product development companies (delivering new financial, insurance products), investigation companies (providing an expert support), and so on.

In conducting the business process modeling in SDBC, we start by structuring and positioning semantically the case information, following modeling steps which are inspired by OS, an in particular by the Semantic Analysis Method (Liu, 2000). Because of the limited scope of this section, we go directly to the output (Figure 3) of such a study, referring readers to (Shishkov, 2005) for more information about the particular modeling steps within SDBC.

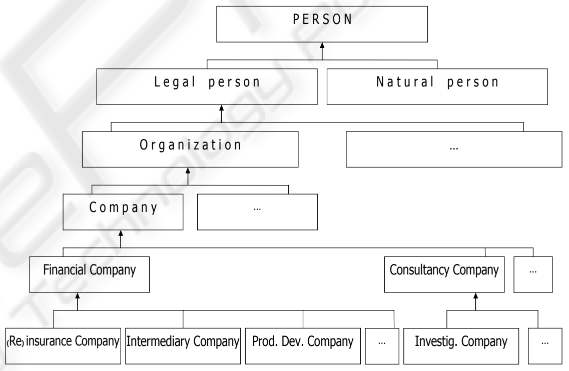

Figure 3: OS-driven business object model.

As seen from the figure, by applying OS, we arrive at a generalization hierarchy for the explored domain, which is easily understandable for both developers and future users. Therefore, an important role of OS is to allow for bringing order in the case information which is usually vague and full of errors.

Next, we conduct in SDBC a LAP-driven analysis of the achieved modeling output. We consider for this purpose the entities in the generalization hierarchy from the perspective of the LAP notion of *actor-role* (Shishkov & Dietz, 2004-3). This allows for a more powerful and flexible modeling – imagine, for example a manager sending a fax, this is not a typical activity for a manager and would therefore make the modeling of such thing complex and confusing, however if we look at this as a role, we could easily model it by introducing the role 'Secretary' (sending a fax is a competence of this role; decision making is a competence of the role ‘Manager’). Thus, SDBC bridges OS and LAP, in allowing to reflect an OS-driven generalization hierarchy in a LAP-driven role model. Again, due to the limited scope of this paper, the modeling steps are not presented in detail; refer to (Shishkov, 2005). Figure 4 presents our resulting role model:

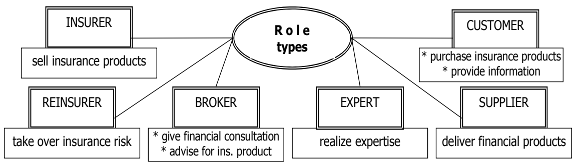

Figure 4: Basic role types within the *Icomp Case.*

Such (identified) role types represent essential modeling constructs in SDBC, as seen from Figure 2. We particularly use the DEMO interpretation (Dietz, 2003) of this LAP-driven concept.

Then we apply the SDBC requirements elicitation apparatus (which includes further information gathering – interviews might be needed, for example) in order to identify the relationships concerning the discovered role types. These modeling steps are inspired by LAP and result not only in identified pairs of role types which have relationship but also provide textual statement describing the relation. For example:

INSURER – sell ins. products – CUSTOMER

Actually, here we need to do partial instantiation because we are interested particularly in the insurance company Icomp:

Icomp – sell insurance products – CUSTOMER

We could have others, as well:

...

BROKER – advice for fin. products – Icomp

We do need to structure further and elaborate the information concerning the relations. For this reason, we apply OS, and in particular – the Norm Analysis Method (Liu, 2000), reflecting each relation in a semiotic norm. This might require additional information gathering and/or clarification.

We take, for example, the first of the above relations and map it to a norm:

Row 1: **Whenever** BROKER has advised CUSTOMER in favour of an Icomp’s product and CUSTOMER fits within Icomp’s policy
Row 2 **If** CUSTOMER decides to purchase this product
Row 3: **Then** Icomp
Row 4: **Is obliged to** insure CUSTOMER according to the concrete product details and based on a CUSTOMER payment, accordingly made.

This represents already a soundly specified and elaborated inter-role relation: Row 1 describes the context, Row 2 – the condition/trigger, Row 3 gives the responsible party, and Row 4 specifies the action through which the roles relate to each other.

This output is to be elaborated further, by relating it to particular units inside Icomp (Financial Department, Account Manager (am), and so on (Shishkov, 2005)). In SDBC, we handle this by conducting a LAP-driven analysis which results in the construction of the SCI Chart (Shishkov, 2005) which in turn allows the identification of *SDBC business process pattern*s, such as:

Icomp (Dept am) start AGREEMENT BROKER

We apply then the *transaction* concept (Figure 5) to each pattern.

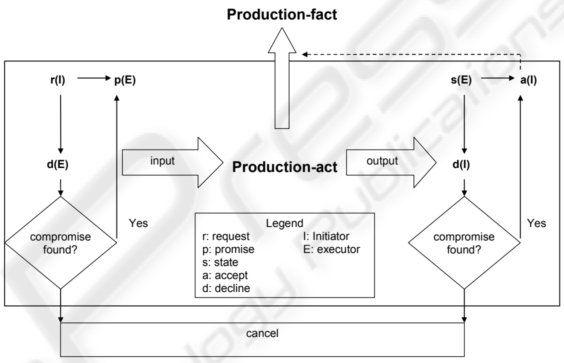

Figure 5: SDBC: LAP-driven transaction concept.

We will not explain this because that concept has been discussed in (Shishkov & Dietz, 2004-1). However, applying this concept, we could identify a number of business process models and present them in DEMO notations (Shishkov & Dietz, 2004-3). An example of such a model is shown on Figure 6 (T01, T02, and T03 are transaction types, while S02 as well as A01 and A02 are role types):

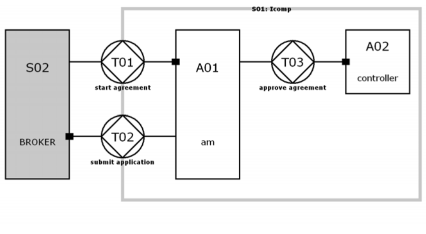

Figure 6: SDBC: identified business process model.

And finally, we apply to such a model a modeling procedure in order to reflect it in a use case software specification model.

How we derive use cases and how then we specify and elaborate them, is discussed in (Shishkov & Dietz, 2004-3) and (Shishkov, 2005).

5 RELATED WORK

The challenge of capturing the essential aspects about business processes for the purpose of further software specification, has been addressed not only by the SDBC approach but also by methods such as Catalysis (D'Souza & Willis, 1998) and Tropos (Tropos Project) as well as by the Model Driven Architecture – MDA (OMG, 2003).

The Catalysis method provides a coherent set of techniques for business analysis and system development as well as well-defined consistency rules across models. However, these techniques concern the software design perspective and have no theoretical roots relevant to the modeling of business processes. Hence, the business process modeling as conducted in Catalysis would inevitably be superficial and therefore the method cannot guarantee an adequate capturing of all related real-life aspects, including semantic and pragmatic ones. In addition to this, Catalysis does not have mechanisms for a mapping between business process models and software specification models. Therefore, a definite strength (in this regard) of SDBC is that, relying on the LAP-OS 'combination', the approach supports adequately the business process modeling task and the software design activities in SDBC stem from a pure business process model, guaranteeing that the application-to-be would function adequately in the business environment in which it would have to be integrated.

The strengths of the method Tropos relate to its capability of conducting a sound requirements analysis, considering the business processes which are to be supported by the application-to-be. From such a business process modeling point of view, the method addresses the software design. The mentioned requirements analysis includes elicitation not only of the 'early requirements' that concern the original business reality but also of the 'late requirements' which are about a corresponding updated (desired) business reality. The analysis is driven by a thorough consideration of the intentions of stakeholders, modeled as goals which are then reflected in the system's global architecture. Its definition is in terms of sub-systems interconnected through data, control, and other dependencies. Then a detailed design follows. Therefore, all this features Tropos as a powerful method for designing software, which appropriately refers to the task of capturing essential real-life aspects that concern the modeling of business processes. However, the method is incomplete with regard to some of these aspects – it is not exhaustive in handling semantics and is insufficiently concerned with essential pragmatic issues, such as communicative actions, negotiations, coordination, and so on (Shishkov, 2005). Further, the method lacks (just like Catalysis) clear and complete guidelines (and elaboration) on how to reflect the business process modeling output in the specification of the application-to-be. Such a specification would therefore inadequately reflect the original business model.

MDA prescribes three viewpoints from which models of the application-to-be (our target software system) should be defined: Computational Independent Models (CIMs) should focus on the environment and requirements of the system, abstracting from the system's construction; Platform-Independent Models (PIMs) should focus on the functionality of the system without revealing details on the specific technological platform on which the system is built, and Platform-Specific Models (PSMs) should define how a PIM is built using some specific platform. Therefore, the Computational Independent Modeling as well as the CIM-PIM transformation relate to the problem addressed in the current paper, namely the achievement of an adequate business-software alignment which is concerned with all relevant real-life aspects. However, bridging business process models and application design by using Computational Independent Modeling and realizing a CIM-PIM mapping, are issues not enough explored, as it is well known. The MDA Community still misses sound guidelines and procedures on how to discover Computational Independent Models and how to reflect them in Platform Independent Models.

Thus, the LAP-OS-driven solution direction considered in SDBC, is actual and is expected to be useful to current application development.

6 CONCLUSIONS

As stated in the Introduction, this paper furthers LAP and OS –related studies, exploring the added value of a LAP-OS combination, in the SDBC-driven modeling of business processes.

Such a combination has been considered from a general perspective (Section 2) and particularly from the perspective of the SDBC approach (Section 3). The justified SDBS-related advantages of combining LAP and OS were then illustrated in Section 4 and supported by an analysis of related work, in Section 5.

The conclusion we reach is that LAP and OS are mutually complementable theories whose combined application contributes to the derivation of an adequate business process modeling output, in the context of a specification of software. Such an adequacy concerns the business process analysis in SDBC, which is supported by a LAP-OS combination, guaranteeing that no essential aspects (regarding the original business reality) are omitted. This guarantee is motivated by the capabilities of OS and LAP to soundly grasp semantics and pragmatics, respectively. This contributes to the creation of a foundation for the further software specification activities.

As it has been studied (and also demonstrated) in this paper, the SDBC approach not only provides a modeling framework suitable for such a LAP-OS incorporation, but it also supports the mapping of a LAP-OS-driven business process modeling output to a software specification model. In this way, the approach guarantees that the software application-to-be would be adequately operational in the business environment in which it would have to function. Hence, these reported developments of SDBS are to be considered in the context of the efforts (within the Software Community) to bring together business process modeling and software specification.

Our agenda for further research includes extension of SDBC and its LAP-OS foundations to Agent Technology, which would concern the modeling of more complex (including autonomous) behavior (Shishkov, 2005). This is connected with the challenge of achieving adequate coordination in Multi-Agent Systems, and should be driven therefore by a sound capturing of semantic and pragmatic aspects.

REFERENCES

Dietz, J.L.G, 2006. Enterprise Ontology – Theory and Methodology. Springer Verlag, Berlin New York, 2006.

Dietz, J.L.G, 2004. Basic notions regarding business processes and supporting information systems. In *CAiSE'04 Workshops in connection with the 16rd International Conference on Advanced Information Systems Engineering*. CAiSE Press.

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

D'Souza, D.f. and A.C. Willis, 1998. Object, components, and frameworks with UML, the Catalysis approach. Addison-Wesley, USA.

Gibson, J.J., 1979. The Ecological Approach to Visual Perception. Boston, USA: Houghton Mifflin.

Goldkuhl, G. and K. Lyytinen, 1982. A language action view of information systems. In: Ginzberg, M., Ross, C.A. (Eds.) Proceedings of the *3rd International Conference on Information Systems Engineering*, TIMS/SMIS/ACM.

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

Object Management Group (OMG), 2003. MDA – Guide, V1.0.1, omg/03-06-01.

Shishkov, B., 2005. Software Specification Based on Re-usable Business Components. Delft, The Netherlands: Sieca Repro.

Shishkov B. and J.L.G. Dietz, 2005. Applying component-based UML-driven conceptual modeling in SDBC. In *ICEIS'05, 7th Int. Conference on Enterprise Inf. Systems*. ICEIS Press.

Shishkov B. and J.L.G. Dietz, 2004-1. Aligning business process modeling 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 Inf. Systems V, Ed: O. Camp, J.B.L. Filipe, S. Hammoudi, M. Piattini, Kluwer Academic Publ., Dordrecht/Boston/London.

Stamper, R., 2000. Organizational Semiotics – Information Without the Computer? Information, Organization, and Technology – Studies in Organizational Semiotics, K. Liu, R.J. Clarke, P.B. Andersen, R.K. Stamper, ed., NL: Kluwer.

Tropos Project. http://www.troposproject.com

Winograd, T. and F. Flores, 1986. Understanding Computers and Cognition: A New Foundation for Design. Ablex, Norwood, NJ.