Combining SDBC and ISDL in the Modeling and Refinement of Business Processes

Boris Shishkov[1] and Dick Quartel[2]

[1] University of Twente, Department of Computer Science, Drienerlolaan 5
7500 AE Enschede, The Netherlands
[2] Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
b.b.shishkov@ewi.utwente.nl, dick.quartel@telin.nl

Abstract. Focusing on the alignment between business requirements and application (software) functionality, the SDBC approach considers three viewpoints complementing each other, namely statics, dynamics, and information. Next to that, the approach addresses systematically and separately business modeling and application design, applying the mentioned viewpoints to both of them. The approach also allows for an adequate extension of its 'dynamic' business modeling, acknowledging the real-life complexity that includes communication and coordination issues, such as meanings, intentions, commitments, and obligations. Hence, in order to consider appropriately these (communication and coordination – related) issues as complementing its dynamic business modeling, SDBC applies at least two modeling techniques. The transformation between them nevertheless complicates the modeling process; furthermore, different techniques use different modeling formalisms whose reflection sometimes causes limitations. For this reason, we explore in the current paper the value which the modeling language ISDL (allowing for useful refinement of business process models) could bring to SDBC, particularly in the elaboration of dynamic (behavioral) business models with real-life aspects. We also explore how SDBC can benefit from ISDL-related methods assessing whether a realized refinement conforms to the original process model. The results reported in this work are usefully supported by an illustrative example.

Keywords: System design, Business process modeling, Refinement, SDBC, ISDL.

1 Introduction

A number of software development approaches have failed because of being insufficiently capable to grasp and utilize the original business information. As claimed in [16], the specification of software and the analysis/modeling of its corresponding business processes, should be considered as one integrated task.

The ***SDBC*** ('SDBC' stands for *Software Derived from Business Components*) approach [13,14,15,16] addresses this challenge, by allowing for a sound *mapping*

Y. Manolopoulos et al. (Eds.): ICEIS 2006, LNBIP 3, pp. 322–335, 2008.

between a business process model and a software specification model. They both are approached through different complementing *viewpoints*, the consistency among which is certainly crucial [4].

SDBC considers three essential modeling viewpoints, namely: *statics* (about the static relationships among entities), *dynamics* (about behavior), and *information* (about data). Next to that, the approach addresses systematically and separately business modeling and application design, applying the mentioned viewpoints to both of them. The approach also allows for an adequate extension of its behavioral business modeling, acknowledging the real-life complexity that includes *communication and coordination issues*, such as meanings, intentions, commitments, and obligations. These could usefully be reflected in another (complementary) viewpoint, namely *communication viewpoint* (as in the SDBC terminology), which plays a role with respect to real-life semantics and pragmatics [14].

Thus, in SDBC the behavior viewpoint and the communication viewpoint are considered in combination, as complementing each other. In applying SDBC, for example, one could firstly use the *DEMO Process step model* [3] for capturing meanings, intentions, commitments, obligations, and so on, and secondly - reflect this in a *Petri Net business process model* [1]. Hence, in order to combine properly these two viewpoints, SDBC would have to use at least two modeling techniques. The transformation between them nevertheless complicates the modeling process; furthermore, different techniques use different modeling formalisms whose reflection sometimes causes limitations.

Since the modeling language ***ISDL*** [6,8] - 'ISDL' stands for *Interaction Systems Design Language*, is powerful as it concerns the refinement of business process models and corresponding assessment for correctness, it is feasible to expect that ISDL can usefully complement SDBC, by allowing refinement of business process models, from the perspective of communication and coordination.

This has motivated our studying potentials for combining SDBC and an integrated modeling facility based on ISDL. In particular, we explore in the current paper the value which this modeling facility could bring to SDBC, particularly in the elaboration of behavioral (dynamic) business models with real-life aspects. We also explore how SDBC can benefit from ISDL-related methods assessing whether a realized refinement conforms to the original process model; actually, the existence of such ISDL-related conformance assessment methods further justifies the claim that ISDL can be useful in the refinement of SDBC dynamic business models. ISDL can also add value in the SDBC-driven mapping of such models towards software specification, particularly in the context of the design of software services [9], since a mapping mechanisms exist between ISDL and BPEL/WSDL specifications.

The outline of this paper is as follows: Section 2 considers SDBC, paying particular attention to concepts that concern the dynamic (behavior) and communication viewpoints. Then we present in Section 3 an illustrative example to be used in our further studies. On this basis, we discuss in Section 4 how SDBC and ISDL can be usefully combined in the modeling and refinement of business processes. Section 5 then analyzes the value of applying SDBC and ISDL in combination. And finally, Section 6 presents the conclusions.

2 SDBC

As suggested in the Introduction, SDBC is envisioned to be a useful modeling framework that approaches business processes from the perspective of related software-specification. This claim has been motivated in [13], where relevant strengths of SDBC are justified: (i) business process modeling based on the theories of LAP and OS [14]; (ii) component-based business-software alignment; (iii) re-use of modeling constructs; (iv) software specification consistent with the Unified Modeling Language – UML [12]. However, the problem addressed in this paper is the business process modeling consistency, particularly with regard to the dynamic and communication viewpoints. Thus, in this section, we firstly outline SDBC, and we secondly consider SDBC-related concepts that concern the mentioned problem.

2.1 Outline and Relevant Features

In summarizing SDBC, we use the following abbreviations as applied in Figure 1: 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 statics, dynamics, data); glbk – *general business coMponent* (which is re-usable by extension); gcbk – *generic business coMponent* (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 [13].

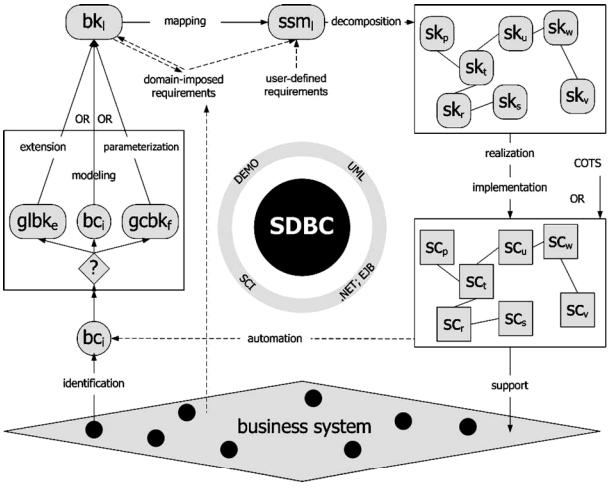

Fig. 1. Outline of the SDBC approach [13]

The figure shows that SDBC is about a component-based business-process-modeling-driven specification and realization of software. The starting point is the consideration of a business system that might be identified and elicited either by using

a scenario or by an abstract business modeling input [14]. Business components are then identified (denoted with textual descriptions), by applying the *Semantic Analysis Method* – SAM leading to the derivation of the so called '*SCI modeling output*' [7,13]. They are 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 and other *Language-Action-Perspective*-(LAP)-driven modeling tools [18] are relevant as far as business coMponents' specification is concerned. Each business coMponent should then be 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. Then, a mapping towards a software specification model should take place, possibly driven by the DEMO-UML transformation mechanism introduced in [17]. The *domain-imposed 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. The (UML-based) software specification model would then need a precise elaboration, achieved partially through its decomposition into a number of software coMponents reflecting functionality pieces [15]. 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 and EJB) based on corresponding software coMponents and such components which are purchased. The resulting component-based application would support the target business system, by automating anything that concerns the initially identified business component(s).

SDBC is thus not only capable to adequately capture semantic and pragmatic real-life aspects but it can also support their further mapping towards software specification, consistently with the *de facto* standard, UML. The SDBC business-software alignment itself is component-based, founded in the *CBD paradigm* – 'CBD' stands for *Component-Based Development* [13]. Such an alignment allows for good traceability between business and software modeling constructs. Finally, the component-based business-software alignment allows for re-use of modeling constructs. This essentially improves the modeling process since building new models includes the re-use of previously built modeling constructs.

2.2 Concepts

SDBC addresses the communication viewpoint, by applying the LAP theory, providing an innovative interpretation of the LAP-driven *transaction* concept.

The generic process of a transaction is depicted in Figure 2. If everything goes smoothly, a *request* is followed by a *promise*, which is followed by a *statement* (preceded by a non-communicative *production act*) which is followed by the *acceptance* of the production fact. However, an entity could also enter discussions (negotiations). For example, if Mary asks for a pizza, it might happen that the sales-person (Paul) says that the shop is closing soon and only hamburgers could be offered – so, this is the discussion, Mary could accept this or not. If she accepts, Paul states a

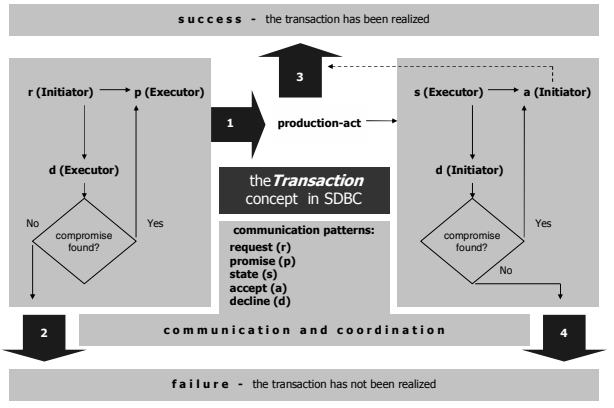

Fig. 2. The transaction concept in SDBC

promise regarding this updated request. Next, if she does not like the hamburger, when Paul states it is ready, they again enter a discussion (whether another hamburger should be delivered or the money – returned back, for example). Depending on the outcome of such discussions, a transaction could reach failure and no *production fact* would then have appeared. That is why Figure 2 presents success and failure 'layers'.

Hence, we have four possible communication outcomes concerning the *initiator* of the transaction (Mary, in the example) and its *executor* (Paul), as shown on the figure: 1(2) – agreement is (not) reached and the executor will (not) realize a production act; 3(4) – the initiator has (not) accepted the delivered result and a Transaction has (not) appeared.

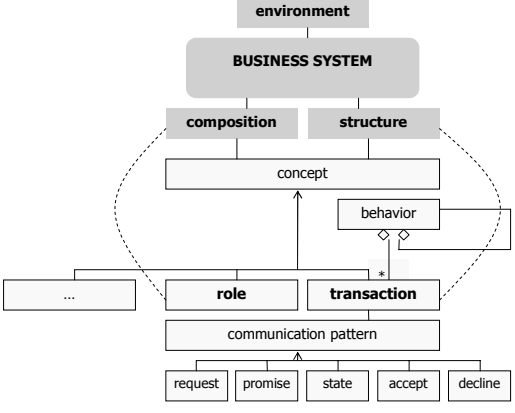

Fig. 3. SDBC concepts

Thus, the elementary business process modeling building blocks in SDBC are transactions; we consider the communication patterns (discussed above), namely: **request**, **promise**, **state**, **accept**, and **decline**, needed for the elaboration of a

transaction. Furthermore, by adopting a subjectivist philosophical stance, SDBC acknowledges that nothing exists without a perceiving/acting agent [7], and especially addresses the entities related to corresponding transactions. However, instead of considering the particular agent (entity) involved (human/artificial), SDBC adopts the *actor-role* (*role*) concept [3]. This allows for a sound and flexible modeling, where for example, a manager sending a fax would fulfill the 'secretary role'.

We have depicted the mentioned SDBC concepts in Figure 3. In positioning the concepts, we follow the classical views of Bunge [2], according to which: a (business) system is characterized by *composition* (it consists of some entities), *structure* (the entities relate to each other), and *environment* (entities and relationships outside the system). As seen from the dashed lines, we consider **role** as a composition-related concept, and **transaction** as a structure-related concept. The five communication patterns are about the transaction elaboration.

SDBC elaborates a transaction via DEMO, expresses multi-transaction structures via Petri Nets, and maps these to UML Activity diagram, in deriving a dynamic software specification model. By applying ISDL, especially in elaboration and refinement, we expect to reach a simpler and smoother representation, benefiting from ISDL's capability to model and refine a broad range of dynamic patterns [8].

In Sect. 3, we introduce an example and partially approach it through SDBC. Then ISDL is introduced and applied to the example, as a complement to SDBC (Sect. 4).

3 The FM Example

The illustrative example addressed in this section, namely the FM example, concerns the *Icomp Case*. Information about the case can be found in [13].

'FM' stands for *Financial Mediator*. The FM facilitates insurance companies. In order to use the mediator, a company should subscribe (registering for its service).

The support provided by FM to registered companies includes advice and product delivery to their customers: (i) a customer can ask FM's advice on which of the companies' products best satisfies a need; (ii) a customer can also ask FM to deliver a product, on behalf of the particular company. We focus on advice delivery only.

To receive advice from FM, the customer should firstly position his(her) request, making it clear whether it is about a health insurance, car insurance, and so on. Secondly, the customer has to specify the particular demand, for instance: to insure a car against theft with the highest possible coverage. Based on this, a *request processing unit* within the FM generates a standardized specification regarding the customer's request, which is delivered to a *match-making unit* (again within FM). The match-making unit realizes then a match, supporting in this way the FM in its advice delivery. This match is driven by a particular criterion that is chosen by the customer. For instance: a preference for the cheapest or the most reliable product available. In order to deliver such a criterion-driven match, the match-making unit uses a data bank that contains relevant rules and procedures. Besides the request processing unit's specification, the match-making unit needs as well an input from a *data search and processing unit* within FM, in order to realize the match. The data search and processing unit searches through the information that concerns registered companies,

and applies procedures to this information. This allows for a precise identification of candidate-matches, relevant to the particular customer's request. Thus, the match-making unit puts the candidate-matches list (delivered by the data search and processing unit) against the standardized request specification (delivered by the request processing unit), and realizes a match, by applying rules and procedures, as mentioned above. All presented information, concerning the current example, is partial and we only use it for illustrative purpose.

In applying SDBC, we start with the initial information structuring, identification of role types, and so on [13]. However, we omit for brevity all initial SDBC analysis and modeling steps and 'arrive' directly at a constructed structural (static) business model – Figure 4. For more information on the SDBC initial analysis and modeling, readers are referred to [13,14]. As for the mentioned model, we have constructed it, using the notations of DEMO, considering the essential concepts role and transaction.

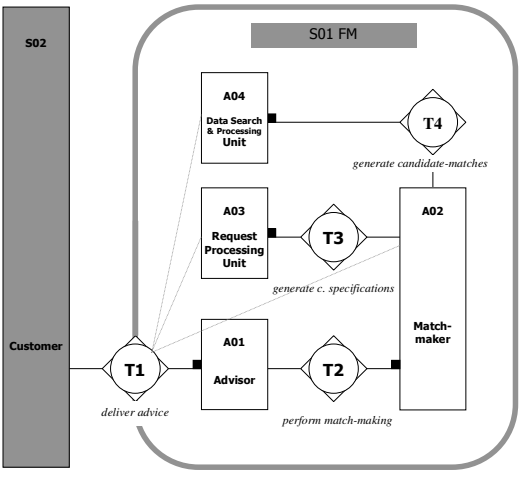

Fig. 4. Static (structural) business model in SDBC

As seen from Figure 4, an external role type is identified (Customer) as well as four internal role types (Advisor, Match-maker, Request Processing Unit, and Data Search and Processing Unit) and four transaction types (Deliver advice; Perform match-making; Generate customer's information specification; Generate candidate-matches). The rounded rectangle indicates our system's boundary. The black boxes indicate which role holds the responsibility for a transaction.

The further task is hence to proceed towards modeling that concerns the communication and dynamic viewpoints. This is to include elaboration of the modeled transactions in terms of communicative acts and coordination (staying consistent with the transaction notion – Figure 2), and also modeling of the flows of production facts.

This all is addressed in the following section which will explore the relevant value that ISDL can bring to SDBC.

4 Complementing SDBC with ISDL

The strengths of ISDL, particularly in the perspective of a SDBC-ISDL combination, are considered in this section, after a brief introduction of ISDL that is actually a language focusing on (business) process modeling at high abstraction level.

4.1 ISDL: Concepts and Notations

ISDL [6, 8] provides a small, but expressive set of basic and generic concepts for behavior modeling, aimed at modeling the behavior of systems from varying domains and at successive abstraction levels [11]. The semantics of ISDL has been defined formally; a method for conformance assessment has also been defined. Furthermore, an integrated editor and simulator is available, and tools supporting conformance assessment and model-to-model (code) transformations are being developed. Figure 5 depicts part of the behavior conceptual model of ISDL, including the *entity* concept; Figure 6 shows how these concepts are represented.

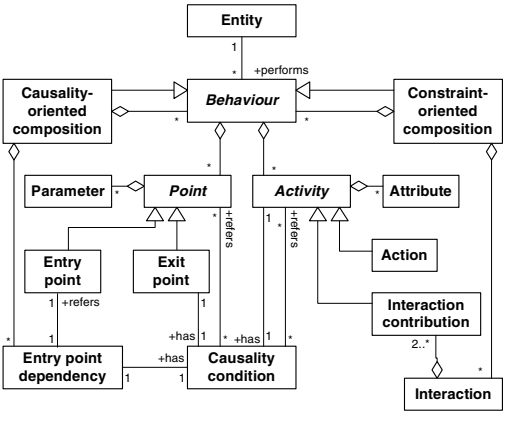

Fig. 5. ISDL concepts

The *entity* concept represents a system part that can perform some behavior. A *behavior* is essentially a set of causally related activities. An *activity* represents some unit of behavior that is atomic, i.e., cannot be split at the abstraction level at which it is defined. Further, an activity either happens, in which case reference can be made to its result, or does not happen at all, in which case no reference can be made to any result, not even to partial results. We distinguish three types of activities. An *action* is performed by a single behavior (entity). Actions are graphically expressed by ovals (or circles). An *interaction* is performed by two or more behaviors in cooperation. An interaction is expressed as two or more connected *interaction contributions* which represent the participation of the involved behaviors. Interaction contributions are expressed by oval (or circle) segments.

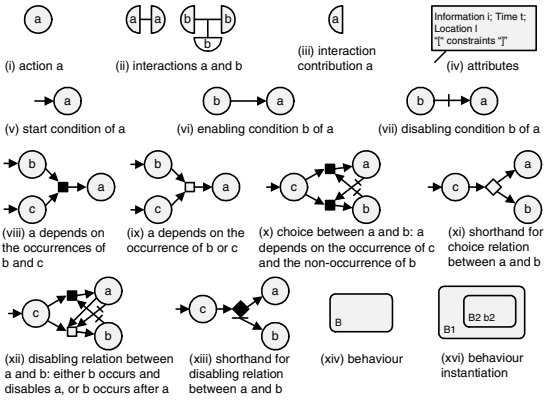

Fig. 6. ISDL language elements

An activity can have *attributes* to represent the relevant characteristics of the occurrence of the modeled real-world activity. Predefined attributes are the information, time and location attribute (see Figure 6 (iv)), representing the activity result (e.g., some information or product), the time of occurrence at which the result is available, and the location where the result is available, respectively. Constraints can be defined on the possible attribute values. The constraints also specify the relation between attribute values established in causally dependent activities. ISDL does not prescribe a language for defining attribute types and constraints, but provides bindings to existing languages that can be used for that purpose. Currently, bindings to *Z*, *Java* and *Q* exist.

Relations between activities are modeled by *causality conditions*. Each activity has a causality condition, which defines how this activity causally depends on other activities. An activity is enabled, i.e., allowed to occur, if its causality condition is satisfied. Three types of basic causality conditions are identified as illustrated in Figure 6: (v) the start condition represents that activity a is enabled from the beginning of some behavior and independent of any other activity, (vi) enabling condition b represents that activity b must have occurred before a can occur, and (vii) disabling condition ¬b represents that activity b must not have occurred before nor simultaneously with a to enable the occurrence of b. These elementary conditions can be combined using the *and*- and *or*-operator to represent more complex conditions. Figure 6 depicts also some simple examples.

Containment of one behavior by another (the composite), is represented by behavior instantiation. A behavior instantiation represents that some behavior instance is created in the context of the behavior that contains the instantiation.

4.2 Activity Refinement

An activity cannot be split at the abstraction level at which it is considered. A more detailed model of an activity can be obtained by decomposing this activity into multiple sub-activities and their relationships. The relevant characteristics of these

sub-activities can be modeled again by the activity concept (i.e., actions, interactions or interaction contributions). Therefore, the activity concept is independent of the abstraction level or granularity at which specific activities are modeled.

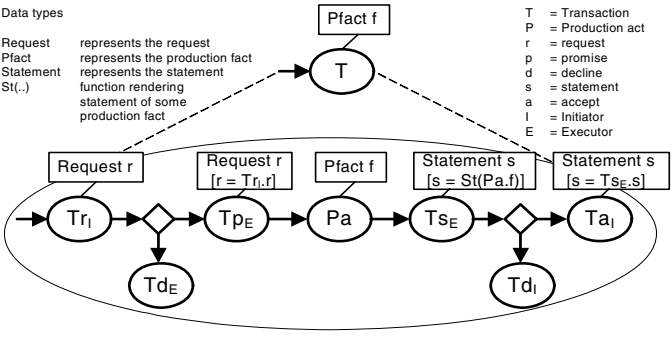

Fig. 7. The ISDL transaction models

In the context of SDBC, the activity concept is used to model transactions as well as their corresponding communication patterns. Figure 7 reflects the generic process of a transaction, modeled at two different abstraction levels. At the highest level, the transaction is represented by a single action, which models the production fact that is established. Characteristics of the production fact are modeled using action attributes. At a lower abstraction level, the transaction's communication aspects are modeled, conforming to the transaction concept (Fig. 2). Separate actions are used to model the transaction's request, promise, state, accept, and decline, and the production act. Observe that actions Td_E and Td_I correspond to the decline of a transaction followed by an unsuccessful negotiation (see Fig. 2), and actions Tp_E and Ta_I represent the promise and acceptance, respectively, which are possibly preceded by an 'initial decline' followed by a successful negotiation.

The result of the transaction behavior at the lower abstraction level should conform to the result of the transaction as modeled at the higher abstraction level. In this case, the result of the transaction behavior is either the occurrence of action Ta_I, which corresponds to the occurrence of T, or the occurrence of Td_E or Td_I, which corresponds to the non-occurrence of T. Consequently, to assess conformance one should assess whether the results as modeled by actions Ta_I and T are equivalent.

A method has been defined for ISDL to assess the conformance of any abstract behavior to a concrete behavior that refines the abstract behavior. The example in Figure 8 represents a special case of this method. For a detailed explanation of the method, interested readers are referred to [10].

4.3 Modeling the FM Example

Using the ISDL transaction models presented in Sub-section 4.2, Figure 8 depicts the modeling of the FM example (Section 3) at three successive abstraction levels. At

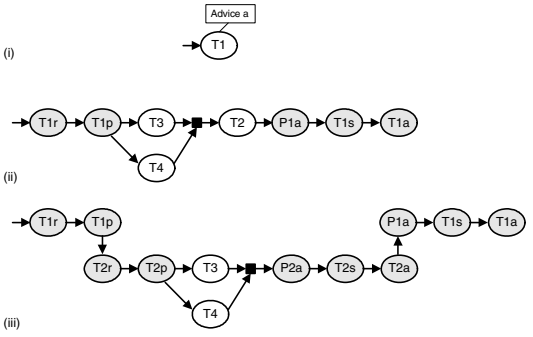

Fig. 8. ISDL models of the FM behavior

level (ii), some detail is added on how the advice is delivered, eliciting part of the internal behavior of the FM: in this case the communication aspects of T1 and the use of transactions T2, T3 and T4. More detail is added in (iii), by elaborating the communication aspects of T2. A similar elaboration can be made for T3 and T4, but has been omitted for brevity. For the same reason, action attributes are not modeled and it is assumed that transactions will not be declined. To clearly distinguish between the abstraction levels at which a transaction is modeled, the communication patterns of a transaction are indicated in grey.

The ISDL models presented so far do not consider the roles involved in each transaction. This aspect can be modeled explicitly using the *interaction* concept. For example, Figure 9 (i) models transaction T1 as an interaction between the role type Customer and FM, where roles are represented by ISDL behaviors. The interaction concept allows one to model the constraints each role may have on the possible results (production facts) of the interaction. For example, a customer may restrict the advices (s)he is interested in to car insurances, whereas FM may only consider insurances from particular companies.

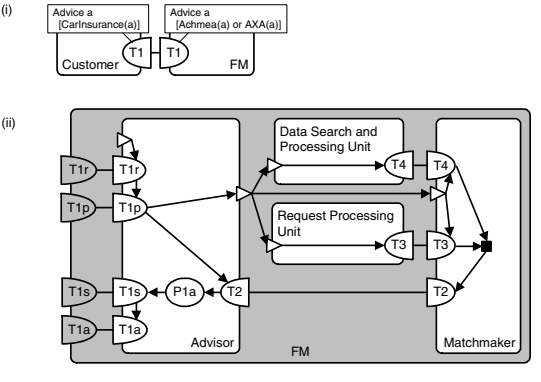

Fig. 9. ISDL models of the FM behavior

Figure 9 (ii) presents the ISDL model corresponding to the SDBC model depicted in Figure 4 (it is elicited which roles are involved in which transactions). In this case the behavior of FM is represented as a *composite behavior* (indicated in grey). Behaviors in a composite behavior can be related using: (i) *constraint-oriented composition*: interactions that relate the interaction contributions of the component behaviors; and/or (ii) *causality-oriented composition*: entry and exit points that represent a causality condition entering a behavior or a causality condition exiting a behavior, respectively. The condition that an entry point represents is associated to it via an entry point dependency. Entry and exit points are represented by triangles that point into or out of a behavior, respectively. Interaction contributions of a component behavior can contribute to interactions of their composite behavior. This is represented by drawing a line between the interaction contributions of the component and interaction contributions of the composite (having the same name in the example).

5 Analysis

As already stated, this section analyzes the suitability and adequacy of combining the SDBC approach and the ISDL language.

Our basic conclusion is that the essential value of combining SDBC and ISDL concerns the possibility to adequately grasp (driven by SDBC) real-life business aspects and realize mapping towards software specification, facilitated by a powerful language instrumentarium (ISDL) that allows one to combine (applying the powerful graphical notations of ISDL) issues concerning the communication and dynamic viewpoints; ISDL can be used at different abstraction levels and its method for conformance assessment allows one to relate successive abstraction levels. In all this, only a single formalism is needed. Further, the ISDL concepts (such as the activity concept) prove to naturally correspond to the SDBC behavior concepts (such as the transaction concept), i.e., ISDL can represent the properties modeled by SDBC concepts. Thus one can smoothly apply ISDL in the context of the SDBC approach.

Complementing SDBC by ISDL, allows not only for an adequate consideration of the notions of role and transaction – these are essential for a business process modeling driven by SDBC [15], but also for modeling transactions (through the interaction concept of ISDL) between different roles. Transactions modeled in such a way, can be defined at a high level of abstraction in contrast to e.g. modeling languages using message passing as the basic interaction concept. When using message passing, one is often forced to define transactions closer to implementation level, since one may need multiple messages to exchange the information that is required to establish the desired transaction result. Instead, the interaction concept in ISDL allows each role involved to define its constraints on the possible interaction result, while abstracting from how these constraints are implemented (e.g. through message exchange).

Therefore, this strong point of ISDL can add value in the context of SDBC – it would be possible that a transaction is decomposed into transaction contributions, defining the responsibility of each role in the transaction (still at an abstract level). When defining a transaction as an action, one abstracts from the contribution/ responsibility of each role in the transaction.

Finally, ISDL could usefully complement SDBC in a mapping towards BPEL/WSDL, for the purpose of business processes implementation using the Service-Oriented Paradigm [9,5], which is nevertheless beyond the scope of this paper.

6 Conclusions

In this paper, we have reported studies that concern the actual challenge of aligning business requirements and software functionality, driven by an adequate identification of a business model and its mapping to a software specification model. These models need to be however appropriately elicited from different perspectives. The SDBC has relevant strengths not only with respect to the business-software alignment challenge in general but also with respect to such a desired multi-viewpoint modeling. Nevertheless, SDBC appears to need further support in achieving consistency with regard to different (complementing) viewpoints, in particular: the dynamic viewpoint and its necessary elaboration (at the phase of business modeling) from the perspective of real-life communication and coordination (where issues, such as meanings, intentions, commitments, and obligations play a role). Thus, it is essential that the SDBC business modeling allows for an appropriate combination between behavior modeling and communication/coordination aspects. However, in realizing this, SDBC uses at least two modeling techniques, the transformation among which unnecessary complicates the modeling process and causes limitations. Hence, if SDBC is applied through an integrated language facility (based on one formalism and possessing powerful modeling expressiveness), the alignment between behavior models and related communication/coordination aspects would be improved.

We have identified ISDL as a good candidate in the mentioned context, given its refinement and conformance assessment capabilities as well as powerful graphical notations. In the course of the current study, we have justified this choice, by finding evidence of particular relevant strengths of ISDL. Next to that, we have demonstrated those strengths and the value of the SDBC-ISDL combination, by means of an illustrative example.

The ISDL notations, driven by one formalism, proved to work usefully in the context of the SDBC approach; they can support the approach in the alignment of behavior business models and (related) communication/coordination aspects, presenting them in a coherent whole. Further, ISDL can be used for refinement at different abstraction levels, as demonstrated in Section 4, supported by mechanisms allowing one to assess whether a refinement conforms to the original process model. Finally, with regard to service-oriented platforms, it is expected that ISDL could support SDBC in mappings to BPEL/WSDL, which although not addressed in the current work, is in the scope of further studies. Besides this, we are also planning to conduct a bigger scale real-life case study, in order to bring more practical evidence in support of our findings. Next to that, we intend to further explore the SDBC-ISDL combination, particularly from the perspective of aligning issues that concern the static and dynamic business modeling viewpoints, and to study possibilities for simulation-driven validation of business process models.

Acknowledgements

This work is part of the Freeband A-MUSE project (http://a-muse.freeband.nl). Freeband is sponsored by the Dutch government under contract BSIK 03025.

References

1. Aalst, W.V.D., Best, E.: Applications and Theory of Petri Nets. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, Springer, Heidelberg (2003)
2. Bunge, M.A.: A World of Systems, Treatise on Basic Philosophy, vol. 4. Reidel Publ. Company, Dordrecht (1979)
3. Dietz, J.L.G.: Understanding and Modeling Business Processes with DEMO. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, Springer, Heidelberg (1999)
4. Dijkman, R.: Consistency in Multi-Viewpoint Architectural Design. University Press, Enschede (2006)
5. Dirgahayu, T.: Model-Driven Engineering of Web Service Compositions: a Transformation from ISDL to BPEL. University Press, Enschede (2005)
6. ISDL home: n.d, http://isdl.ctit.utwente.nl
7. Liu, K.: Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge (2000)
8. Quartel, D., Dijkman, R., Van Sinderen, M.: An Approach to Relate Business and Application Services Using ISDL. In: EDOC 2005, 9th IEEE International EDOC Enterprise Computing Conference (2005)
9. Quartel, D., Dijkman, R., Van Sinderen, M.: Methodological Support for Service-Oriented Design with ISDL. In: 2nd International Conference on Service Oriented Computing (2004)
10. Quartel, D., Ferreira Pires, L., Van Sinderen, M.: On Architectural Support for Behaviorur Refinement in Distributed Systems Design. Journal of Integrated Design and Process Science 6(1) (2002)
11. Quartel, D., Ferreira Pires, L., Van Sinderen, M., Franken, H.M.: On the Role of Basic Design Concepts in Behaviour Structuring. Computer Networks and ISDN Systems (1997)
12. Rational, UML Resource Center: http://www.rational.com
13. Shishkov, B.: Software Specification Based on Re-usable Business Components. Sieca Repro, Delft (2005)
14. Shishkov, B., Dietz, J.L.G., Liu, K.: Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In: ICEIS 2006, 8th International Conference on Enterprise Information Systems (2006)
15. Shishkov, B., Dietz, J.L.G.: Applying Component-Based UML-Driven Conceptual Modeling in SDBC. In: ICEIS 2005, 7th International Conference on Enterprise Information Systems (2005)
16. Shishkov, B., Dietz, J.L.G.: Aligning Business Process Modeling and Software Specification in a Component-Based Way, The Advantages of SDBC. In: ICEIS 2004, 6th International Conference on Enterprise Information Systems (2004)
17. Shishkov, B., Dietz, J.L.G.: Deriving Use Cases from Business Processes, The Advantages of DEMO. In: Camp, O., Filipe, J.B.L., Hammoudi, S., Piattini, M. (eds.) Enterprise Information Systems V, Kluwer Academic Publisher, Dordrecht, Boston (2004)
18. Winograd, T., Flores, F.: Understanding Computers and Cognition: A Foundation for Design. Ablex, Norwood (1986)