Context Based Reasoning in Business Process Models

Pavel Balabko, Alain Wegmann
Systemic Modeling Laboratory (LAMS)
IC-EPFL, Lausanne, Switzerland
Pavel.Balabko@epfl.ch; Alain.Wegmann@epfl.ch

Abstract - *Modeling approaches often are not adapted to human reasoning: models are ambiguous and imprecise. A same model element may have multiple meanings in different functional roles of a system. Existing modeling approaches do not relate explicitly these functional roles with model elements. A principle that can solve this problem is that model elements should be defined in a context. We believe that the explicit modeling of context is especially useful in Business Process Modeling (BPM) where the meaning of any model element should be defined precisely. The contribution of our work is at the context-aware modeling framework for BPM. We model a system as the composition of small roles, where each role of a system is defined in its own context.*

Keywords: Context, role, business process modeling, model, human reasoning.

1 Introduction

In the design of information and business systems, modeling plays two roles: an implementation role and a communication role.

In the implementation role, a model can be considered as a bridge between a developer's understanding of a problem domain and a code that has to be implemented. In this case models can be used for the verification of some critical properties of software, for code generation, for simulation etc. This role of modeling is well studied: there are many formal approaches for program verification, automatic code generation and etc.

In the communication role, a model can be considered as a means that allows system stakeholders (users, developers, etc) to talk about systems to be designed. This role of modeling is underestimated. Modern modeling approaches often are not adapted to human reasoning. This problem was clearly stated in [5]: "We would like to emphasize informal and yet conceptually precise and practically significant approaches, rather than merely formal languages theory using different formalisms and therefore making them hard to comprehend and compare". We therefore need to make existing approaches more convenient for human reasoning.

A shift from an implementation oriented modeling approach to a "human-friendly" one should be based on a set of disciplines (like philosophy and system sciences) that pay attention to a cognitive side of modeling. These disciplines can provide a set of principles that can integrate human factors in existing modeling techniques. In this work we consider one principle that states that "all of human inquiries occur within contexts" [7]. We argue that an explicit modeling of a context plays an important role in system modeling and makes models more comfortable for human reasoning. This observation is based on the sign model of the American pragmatist philosopher Charles Sanders Peirce. In his sign model he emphasized that a model element stands for an entity in the Universe of Discourse, not in all respects, but in reference to a sort of idea or situation modeled as context.

We believe that an explicit modeling of a context is especially important for business system and business process modeling (BPM). However this explicit modeling is impossible with existing approaches for BPM. Therefore, in our work, we propose a framework for BPM that makes contexts explicit in models. This framework is called SEAM[1]. In our framework a system is modeled as a set of roles, where each role is modeled in its own context. To reason about a system as a whole, one should analyze relationships between different roles of this system.

The structure of this paper is the following. In Section 2 we explain a problem that is addressed in our paper. Section 3 is a main contribution of our paper that gives a solution to the addressed problem. In this section we explain the theoretical foundations (Section 3.1) and description (Section 3.2) of our context-aware modeling framework for BPM. In Section 4 we give an example of a model of Simple Banking System built in our modeling framework. We finish with the conclusion.

2 Problem Statement

Our belief in the importance of the explicit modeling of a context in the field of BPM is based on two evidences.

The first evidence is the business process definition of the Workflow Management Coalition that accumulates the experience of over 300 member organizations worldwide. This definition states that an objective (goal) of a business process has to be defined within a context:

[1] SEAM stands for Systemic Enterprise Architecture Methodology. See [16] for details.

Business Process [17] is "a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, within the context of an organizational structure defining functional roles and relationships". The notion of *context* is made explicit in this definition. In section 2.1 we consider how this definition corresponds to the existing business process modeling techniques.

As the second evidence, in section 2.2 we consider a practical example that illustrates the core of a problem: the necessity of the explicit context modeling for the understanding a meaning of a model element in different functional roles of a system.

2.1 State of the Art

The Workflow Management Coalition definition of a business process is based on four key elements: a business process goal (or a business objective), context for this goal, collaborative activities and roles that participate in these activities. *Object* (or business object) is yet another concept that should be considered when the business process has to be implemented. An object is the model of an entity in the Universe of Discourse. It plays roles in a business process by means of participating in different activities. In all, we have to address five key elements: *goals*, *context*, *activities*, *roles* and *objects*. [10].

To model these five business process elements, some Process Modeling Technique (PMT) should be used. Carlsen in [3] refers to the following PMT types: Traditional Input Process Output (IPO) techniques; Conversation Based techniques; PMTs based on role modeling; System thinking and system dynamics techniques; Constrained-based representations techniques. None of the existing PMTs model a context explicitly. However the PMT based on role modeling is a technique that can be extended to model a context. This is because the concept of context is tightly related with the concept of role. The role of a system is a partial specification of a system behavior in a given context.

We use the PMT based on role modeling as a basis for our framework. There are several PMTs based on role modeling. The following three seem to be the most important: RIN – Role Interaction Networks [14], RAD – Role Activity Diagram [12] and OORAM – the Object-Oriented Role Analysis Method [13]. These approaches are quite similar. Roles are considered as sets of sequentially ordered actions and/or interactions. The main drawback of these approaches is that goals are difficult to model with these PMTs. Another problem is that states are defined in such a way that it is difficult to split the state into subsets (for different contexts). These two problems are related to the fact that the PMTs do not have a state structure (state is considered as an instant between connective actions). Therefore we are looking for a new PMT based on role modeling that makes explicit the five key elements of a business process (*goals*, *context*, *activities*, *roles* and *objects*) and the *state structure* of a system.

2.2 The Core of a Problem

The model of a complex system has many functional roles. Each role is an abstraction of a system behavior in some given context. In a typical situation of time shortage, a system designer does not model all functional roles independently. Instead he tries to build one "universal" model (a tradeoff between model size and its understandability) as an implicit composition of all functional roles of a system. In this case, the meaning of a model can be difficult to understand: the meaning of any model element depends on the functional role of a system. Since a system has many functional roles, the same model element can have multiple meanings. As a result, it becomes difficult to reason about different roles of a system.

To illustrate, we give an example that we use throughout our paper. Let's consider a business system model that models a Simple Bank (see figure 1).

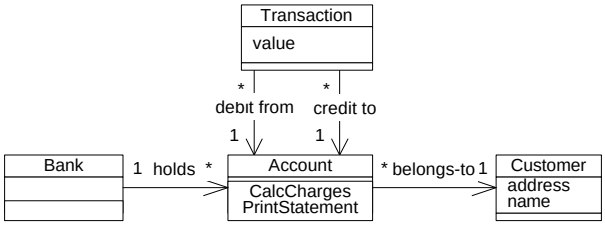


Figure 1. Simple Banking System Object Model

This example was taken from the Ratio Group's white paper [6] that explains the UML notation for Object-Oriented Analysis and Design. This is a pure UML example that does not have any notion of context. So, what is the problem? The model in figure 1 was built as a tradeoff between the model's size and its understandability. This model represents some (but not all) concepts that can be used to explain two roles of a Simple Banking System in two different contexts.

First, this model can be used to explain the role of the Simple Banking System in the context of the relation between a customer and a bank. It can be considered as a model of a bank from the customer's point of view. A customer creates an account in a bank and then makes credit/debit transactions with its account. Note that in order to explain the relation between a customer and a bank, a customer does not need to know that a bank holds many accounts. Therefore only one account can be modeled.

Second, this model can be used to explain a role of the Simple Bank System in the context where there are multiple accounts. However, in this model, all these bank accounts are not related to each other explicitly. Each account is responsible only for keeping its own balance. We cannot see the whole idea of a bank: how a bank accumulates money from depositors and invests it.

We can see that the model in figure 1 is a mixture of concepts used to reason about different roles of a business system. Concepts are not specified with roles they belong to, which complicates the overall comprehension of a

model. In our work we propose the explicit modeling of context can make the meaning of model elements more precise and improve model understandability.

3 Our Approach for Context Modeling

This section presents the results of our research work. In section 3.1 we begin with theoretical foundations of our context-aware modeling framework for BPM. In section 3.2 we continue with a description of our modeling framework.

3.1 Theoretical Foundation of Context

In this section we introduce the five Context Modeling Principles that provide the basis of our modeling framework. As we mentioned in the introduction, to make models more "human-friendly" we should use principles based on a set of disciplines (like philosophy and system sciences) that pay attention to the cognitive side of modeling. The first three principles have a philosophical foundation. The forth and fifth principle are based on system sciences.

First we show where the term *context* originates from. The simple observation that all of human inquiry occurs within contexts triggered the use of the concept of context. This observation was clearly seen by the American pragmatist philosopher Charles Sanders Peirce in the end of ninetieth century. In his sign model[2] he emphasized that a model element (in a model) stands for an entity (in the UoD), not in all respects, but in reference to a sort of idea or situation (in the UoD). Based on this observation, Pieter Wisse extended the Peircean sign model with the notion of context (in a model) that models a given situation (in the UoD). The result is a hexad shown in figure 2. "The original three triadic elements of Peirce reappear as dimensions [in this hexad]" [15]. Figure 2 shows what we have just explained. Model element sense and context sense in figure 2 are defined in a semantic domain that means that they have a well defined meaning.

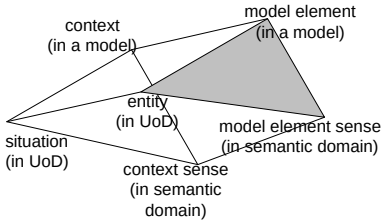


Figure 2. Pieter Wisse hexad sign model.

[2] Peircian sign model is considered as a triad <entity, model element, model element sense> [we use our terminology, the original Peirce triad is <object, sign, interpretant>], where entity is "any concrete or abstract thing of interest" [9] in the Universe of Discourse (UoD). The UoD corresponds to what is perceived as being reality by a business analyst. Identified entities are modeled as model elements in a model. Model element sense is a meaning of a model element.

As a result we can conclude that for the correct interpretation of a model element in a model, any model element should be specified explicitly within a context (1st context modeling principle).

"The history has been dominated by invariantism" [Vassallo01]. This means that only one epistemological interpretation of any model element was considered, i.e. the knowledge was considered as context invariant. However in recent times humans face the need to reason about complex systems. These complex systems can be observed in different situations and interpreted differently, which means that the notion of context allows for multiple models of the same system. Systems can be modeled differently depending on the situations in which they are "situated". This is especially useful in the analysis of complex systems because it allows for dealing with complexity in a systematic way. "An essential feature is that contexts provide means to focus on aspects that are relevant in a particular situation while ignoring others" [11]. We call this principle as a "Multiple contexts" principle (2nd context modeling principle).

As we explain in the next section, each model is a set of related model elements (attributes, actions, etc). Each model element models an entity from the UoD. Moreover, some model elements (modeled in different contexts) can model the same entity in the UoD. We call such model elements *identical*. The modeling process for a complex system consists in building models in specific contexts and in finding identical model elements in these models. We call this an "Identity" principle (3rd context modeling principle).

The 4th modeling principle is based on the System Science [4] principle that any system can be explained with a *goal*. A goal shows the results to be achieved by a system, placed in a given context. This gives us the "Goal driven context" principle that states a context has a goal (4th context modeling principle).

The 5th context modeling principle is based on the System Science [4] principle that a system is defined through its relations with an environment or as a set of its internal properties. Therefore a context can be perceived dually. First, it can be perceived as a set of possible influences from a model environment. In this case, the most common way to define context is to specify it as a set of objects and events that can influence the behavior of an object (see for example [11]). Second, a context can be perceived as states that can be changed by external influences. We have found in literature[3] the following two definitions that take in account both meaning of context:

"*Context (1st definition)* [Cambridge International Dictionary of English]: *the influences and events that helped cause a particular event or situation to happen*".

[3] See [2] where he reviews different context definitions.

"Context (2nd definition) is a subset of the complete state of an individual [a system] that is used for reasoning about a given goal"; this definition is given in the field of Knowledge Representation and Reasoning (KRR) in [8].

In a summary of this section we repeat the five Context Modeling Principles that we have used as a basis for our context-aware modeling framework:

Table 1. Context Modeling Principles

1. "Explicit context" principle	Any model element should be specified explicitly within a contexts;
2. "Multiple contexts" principle	Multiple models of the same system are possible, where each model is defined in its proper context
3. "Identity" principle	Identity of model elements from different contexts should be specified explicitly.
4. "Goal driven context" principle	Context should define consequences of the behavior of an object that is placed in this context.
5. "Duality" principle	Context should be specified dually: as influences form the environment of an object and as object states that can be changed by these influences; these two specifications should not be contradictory;

In the next section we show how these principles have been used in our framework.

3.2 SEAM Modeling Framework for BPM

In this section we introduce our context-aware modeling framework for BPM. The core of this framework is the SEAM Visual Language (SEAM VL). In section 3.2.1 we give definitions for model elements in SEAM VL. Then in section 3.2.2 we explain the notation of SEAM VL. Section 3.2.3 gives an overview of the SEAM methodology that explains how SEAM VL should be used for building multicontextual models.

3.2.1 Definition of the main concepts.

In this subsection we present concepts that we use in SEAM VL. In order to give rigorous definitions for concepts that we use in our VL, we had to choose a consistent semantic framework. We use the ISO/ITU standard "Reference Model for Open Distributed Processing" – part 2 [9] as a framework.

Based on RM-ODP, modeling consists of identifying entities in the universe of discourse and representing them in a model. The *universe of discourse* (UoD) corresponds to what is perceived as being reality by a business analyst and *entity* is "any concrete or abstract thing of interest" [9] in the UoD. Identified entities are modeled as *model elements* in a *model*. Model elements are different modeling concepts (object, action, behavior etc). We give definitions of some modeling concepts necessary for the understanding of our paper (other definitions see in the RM-ODP). We begin with the definition of an object. If in the UoD we have entities that can be modeled with state and behavior, we model these entities as objects:

Object: "An object is characterized by its behavior and dually by its state" [9].

The duality of state and behavior means that the state of an object determines the subsequent behavior of this object. The definition of an object is based on the definition of behavior and state:

Behavior: A collection of actions and a set of (sequential) relations between actions.

State: A collection of attributes, attribute values and relations between attributes.

Attributes can change their values; relations between attributes can be created or deleted. To specify these changes we use pre- and postconditions. Based on the definition of behavior we define a role:

Role: "An abstraction of the behavior of an object" intended for achieving a certain common goal in collaboration with other roles.

Up to this point we have specified all necessary terms that we need to give a definition of context. For the following definition (1st notion of context from section 2), we were inspired by OOram [13]:

Context (1st definition) is the set of collaborating roles along with their state and behavior.

This definition allows for the explicit specification of external roles that can influence the behavior of an object. To specify context as "a subset of the complete state of an individual" (the second notion of context from section 2) we use the following definition:

Context (2nd definition) is state and behavior of a role.

The last definition that we need is:

Goals (of a system in a context) are postconditions for all actions in a role. This role is a role of a system in a given context.

Based on the above-mentioned definitions, we can see that we specified all four elements of the business process: roles, activities (behavior of collaborating roles), goals and objects (that play roles in the implementation).

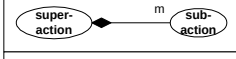
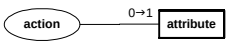
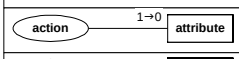
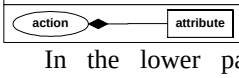
3.2.2 Notation

To visually represent context, we use a notation inspired by UML (see an example in figure 3.a and 3.b). We represent a context (a set of collaborating roles) by a rectangle that includes a set of collaborations (dashed ovals), set of roles (stick men) and role names (names below stick men). The name of a context is given in the upper part of a box. We represent objects with cubes; object names are given below cubes.

For each role in a collaboration we can show a detailed specification (see an example in figure 3.c). We show it as a box with three panes. This notation is similar to the representation of a class in UML. The difference is that instead of an attribute compartment in UML (middle pane

in each box) we use graphical notation based on a UML class diagram. It contains not only attributes and relations between them, but also actions. Actions are used to specify the belonging of attributes to a certain context. Each role in the middle pane should contain at least one action of the same name (like the "Create Account" action in the "Create Account" role). This action is associated with attributes that are defined in the context of this role. In the following table we explain the semantics of relations between actions and attributes:

Table 2. Semantics of relation between actions and attributed in the SEAM Visual Language

	The super-action includes m sub-actions. This means that m instances of an action happen in the life cycle of super-action.
	Action is responsible for the creation of an attribute (attribute multiplicity changes from 0 to 1). We emphasize newly created attributes and relations with thick lines (postconditions).
	Action is responsible for the destruction of an attribute (attribute multiplicity changes from 1 to 0).
	Action is responsible for the persistence of an attribute

In the lower pane of each box, instead of the compartment that holds a list of operations in UML, we use the graphical notation based on a UML activity diagram to represent a role behavior.

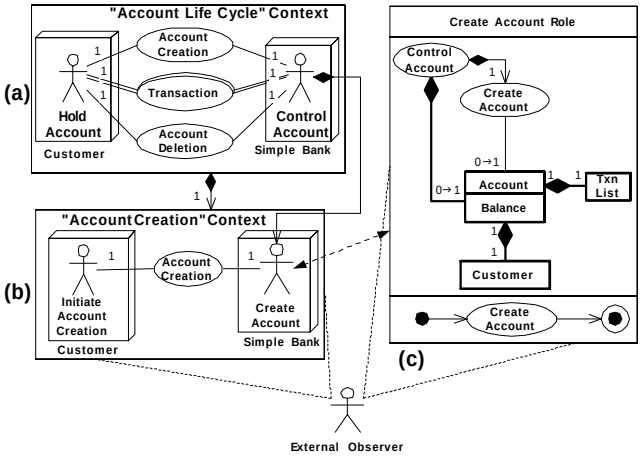


Figure 3. Roles of a Simple Bank object in different contexts (a) "Account Life Cycle" context; (b) "Account Creation" context; (c) Detailed specification of the "Create Account" role

Figure 3.b gives an example of a model of a Simple Bank object in the "Account Creation" context. It models a situation where a customer creates an account in a simple bank. This context includes two roles: "Initiate Account Creation" and "Create Account" and limits the scope of the model to communications that can be observed by an External Observer between Customer and Simple Bank objects. These communications have a final goal of creating an account by the Simple Bank object. This goal is reflected in a diagram from figure 3.c. An Account is created with TxnList (transaction list) and Customer attributes. Note that creating an account is not an end in itself. This account will be further used by the Customer object in the "Account Life Cycle" context (figure 3.a) for making debit and credit transactions and then for the account deletion. To reflect a fact that a created account will be used in a "higher" context, we include the "Control Account" action in the middle pane of the "Create Account" role (figure 3.c). We connect this action to the Account attribute with a thick line (to emphasize postconditions). This line shows that Account, Customer and TxnList attributes will be used in another context.

Model elements, presented up to this point, show the usage of the three context modeling principles given section 2. The context of model in figure 3 was specified explicitly ("Explicit Context" principle), the goal of a Simple Bank object in the "Account Creation" context was specified as a set of postconditions ("Goal Driven Context" principle) and we have specified the context dually ("Duality Principle") as a set of roles in the environment of a Simple Bank object and as a set of internal properties (attributes). Two remaining principles ("Multicontextual Paradigm" and "Duality principles") make sense in a multicontextual model. In the next subsection we consider SEAM methodology, which explains how we work with multicontextual models.

3.2.3 SEAM methodology

The SEAM methodology explains how a system considered in several situations is modeled using the SEAM VL. In SEAM methodology we do not prescribe how a modeler should build a model. We only give modeling constraints that a modeler should follow in our methodology:

1st constraint: A system of interest can be considered in a number of meaningful situations. Each situation should be modeled as a group of collaborating objects (1st notion of context) with a certain goal. We recommend using the "Create Collaboration, Do Collaboration, Delete Collaboration" pattern. This pattern makes explicit the life cycle of relations between roles. Thus we can specify how a relation is created, how it is used and how it is deleted. For example in Figure 4 we have specified the "Account Life Cycle" context as three collaborations: "Account Creation", multiple "Transactions" and "Account Deletion".

2nd constraint: The hierarchy of contexts should be specified using the whole-part relation. It specifies context containment and the multiplicities of context containment. For example, in the model of a Simple Bank we have to specify that the "Account Life Cycle" context contains one "Account Creation" context, multiple "Transaction" contexts and one "Account Deletion" context.

3rd constraint: For each context (a group of collaboration roles) a separate model of a system of interest should be built (2nd notion of context). For example, in the "Account Creation" context a model of a Simple Bank object is given in figure 3.c.

4th constraint: Based on the hierarchy of contexts, a composition of roles from lower level contexts should be made. For example, the "Control Account" role (in the

"Account Life Cycle" context) in figure 4 is composed of one "Create Account" role, multiple "Make Transaction" roles and one "Delete Account" role. Roles are composed basically by finding identical attributes and putting compositional constraints. Detailed information about the composition of roles in our methodology can be found in [1].

The first modeling constraint of the SEAM methodology reflects the "Goal Driven Context" and "Multiple Context" principles (see Section 3.1). The fourth constraint defines a composition of roles from different contexts by means of finding identical attributes (the "Identity" principle). The first and third reflect the "Duality" principle and all constraints are based on the "Explicit context" principle. Thus SEAM modeling framework reflects all five context modeling principles that we have identified in section 3.1.

Due to the limit of space we can not give a more extended description of the SEAM methodology. However we believe that SEAM methodology can be better understood through examples. In the next section we give a practical example that shows how the SEAM context-aware framework for BPM (SEAM visual language and SEAM methodology) can be used to specify a Simple Bank model.

4 Example

In this section we give an example that illustrates how the SEAM modeling framework is used to build models. This example's goal is to build a model of the Simple Bank object in the "Simple Bank Life Cycle" context that is the broadest context in our model (upper part of figure 4).

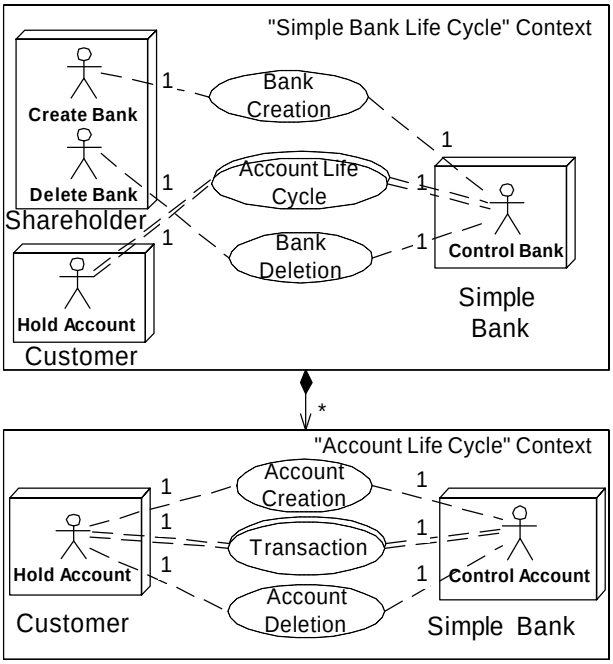
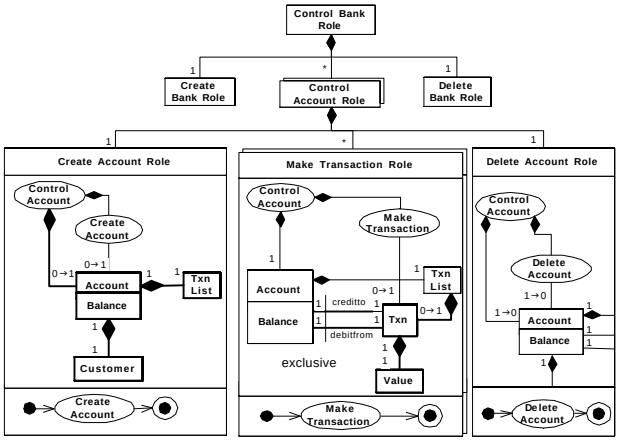


Figure 4. Hierarchy of contexts for a model of the Simple Bank object

The "Simple Bank Life Cycle" context includes multiple "Account Life Cycle" contexts, one "Bank Creation" context and one "Bank Deletion" context. The "Account Life Cycle" context (lower part of figure 4) in its turn includes one "Account Creation" context, multiple "Transaction" contexts and one "Account Deletion" context (see figure 3 from the previous section). Diagrams from figures 3 and 4 represent a hierarchy of contexts for a Simple Bank object. Therefore up to this point we have satisfied the first two SEAM methodology constraints: we have different contexts that model different situations for a Simple Bank object and we have a hierarchy of contexts.

Now we can build a model of Simple Bank for each of the specified contexts (third SEAM methodology constraint). We start with the three lowest contexts in the hierarchy: "Account Creation", "Transaction" and "Account Deletion". The three models of a Simple Bank ("Create Account", "Make Transaction" and "Delete Account") in these contexts are shown in figure 5.



$CreateAccount.Account \leftrightarrow MakeTransaction.Account$

$(for\ \forall\ MakeTrasaction\ in\ ControlAccount) \leftrightarrow DeleteAccount.Account$

$CreateAccount \rightarrow MakeTransaction$

$MakeTransaction \rightarrow MakeTransaction$

$MakeTransaction \rightarrow DeleteAccount$

Figure 5. "Control Account" role as a set of three lower level roles and synthesis constraints

The context, one level higher, is the "Account Life Cycle" context. By means of composing the three roles ("Create Account", "Make Transaction" and "Delete Account"), we can obtain the "Control Account" role for this context. The "Control Account" role is a model of a Simple Bank from the point of view of customer's account. It specifies how a customer works with a single account that he creates in a Simple Bank.

The fourth SEAM methodology constraint states that in order to compose roles we have to specify identical attributes and put compositional constraints. We give them in the lower part of figure 5.

The three models in figure 5 help us to reason about attributes and actions in their original contexts. However this diagram does not help us to reason about the "Control Account" role as a whole, without looking into details of

the base roles. In order to do this, we define a *multicontextual* view that we show in figure 6.

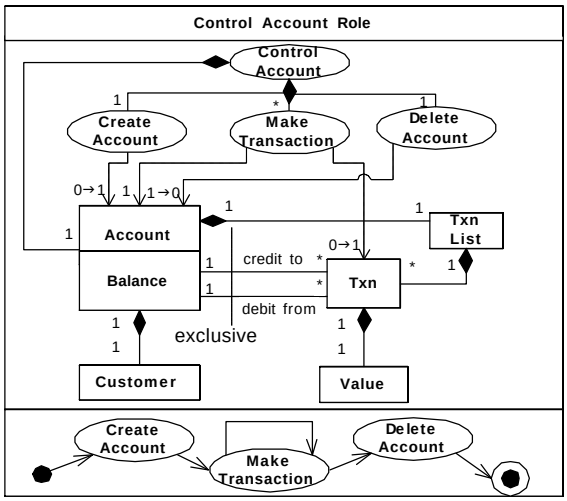


Figure 6. "Control Account" role

A multicontextual view shows the result of the composition of base roles. It allows for the reasoning about a composite role as a whole by means of hiding details about base roles. Base roles are represented as actions (ovals in the middle pane of role model). A multicontextual view shows the relationship of attributes to lower level roles with dashed lined arrows. This helps the reader of a diagram to understand the meaning of attributes for lower-level roles. If the meaning is not clear, a reader can always refer to a detailed model of a lower-level role. Another important property of a multicontextual view is that it preserves multiplicity constraints. Multiplicities in figure 5 are consistent with multiplicities in figure 6. For example, TxnList in the "Make Transaction" role (figure 5) includes zero or one Txn attribute because in the context of making a transaction, we are interested in the modeling of only one debit/credit transaction. The "Control Account" role includes multiple "Make Transaction" roles (figure 6). Therefore TxnList in figure 6 includes multiple Txn attributes. In another example, a "Make Transaction" role (figure 5) includes one Account. There are multiple "Make Transactions" roles in a "Control Account" role (figure 6). However, Account in any "Make Transaction" role is identical with Account in the only one "Create Account" role. Therefore Accounts in all "Make Transaction" roles are identical and there is only one Account in the "Control Account" role. This kind of reasoning allows for the automatic generation of figure 6 based on the three based roles and composition constraints presented in figure 5. The composition of roles as it is shown in this example addresses the scalability problem of VLs. A visual model can be specified as a composition of smaller visual models (roles) and a composed model (multicontextual view) can be generated automatically.

Let's resume with the specification of a Simple Bank object. Once again we can go to the higher context: "Simple Bank Life Cycle". By means of composing the three roles ("Create Bank", "Control Account" and "Delete Bank"), we can obtain the "Control Bank" role for this context. The composition of the tree roles is done in a similar way as in the previous level of the context hierarchy (therefore we do not show details of composition). However this composition is different in one important aspect: there is a new "Bank Assets" attribute in a model (see figure 7). This attribute illustrates an emergent property appeared as a result of composition. This property has an important business value. The idea of this attribute is to relate all accounts of a Simple Bank. This can be expressed with composition constraints:

$$\text{ControlBank.BankAssets.Balance} = \sum_{\text{Control Account}} \text{ControlAccount.Account.Balance} \quad (1)$$

$$\text{ControlBank.BankAssets.Balance} > 0 \quad (2)$$

These constraints show that the balance of a Simple Bank is the sum of all account balances in this bank. Thus we can see the idea of a Simple Bank: it accumulates money from depositors (customer's credit transaction). A Simple Bank can also invest money (customer's debit transactions) if the overall balance of a Simple Bank is positive (see the second constraint above).

Based on the composition constraints given above, the three roles ("Create Bank", "Control Account" and "Delete Bank") can be composed. The result of composition is given in figure 7.

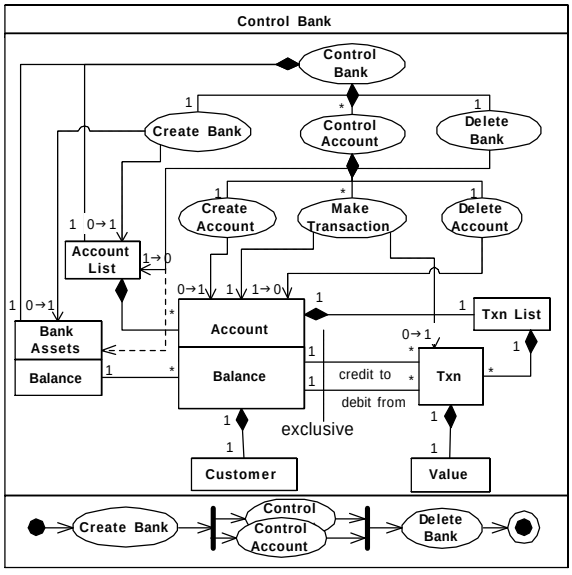


Figure 7. "Control Bank" role

The "Control Bank" role is a model of a Simple Bank that includes several customers' accounts. The goal of this model is to show how a Simple Bank accumulates money from depositors and invest them.

Putting actions on the same diagram with attributes (or concepts), like in figures 6 and 7, provides several advantages for system modeling. It allows for:

- Linking a model in a diagram with a concrete context where this model is valid. This helps to avoid diagram

misunderstanding like in the case with the diagram in figure 1. In our approach we explicitly make a choice of a situation that we model. For example, for a Simple Bank, depending on the situation, we can get diagrams in figure 6 or 7.

- Relating system attributes and behavior. This helps in the analysis of models. Analysis becomes easer because a business modeler can immediately see how actions can influence the state of a system. Therefore a modeler can "play" with a model by mentally "executing" actions. This can be important in discussions with non-professionals (customers, business people).
- Making explicit the life cycle of attributes. Our framework forces a modeler to think about attribute creation, persistence and deletion, which helps to avoid mistakes in later phases. In case of IT systems, modeling of attribute persistence is useful for its later implementations with data bases.

Note that if we remove all actions from the middle pane of the "Control Account" and "Control Bank" roles, we can obtain usual UML class diagrams (figure 8).

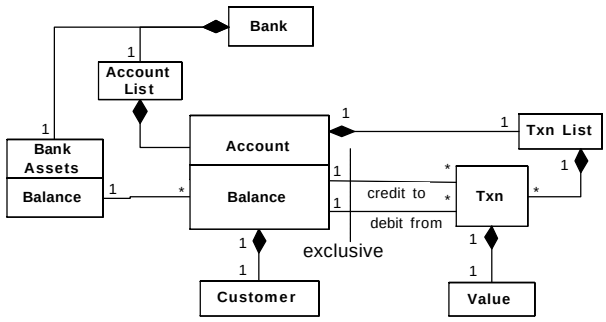


Figure 8. UML models of a Simple Bank object.

The diagram in figure 8 originates from figure 1. The problem with the diagram from figure 1 was that the purpose of this diagram was not clear. It represented some (but not all) concepts that were used to explain two roles of a Simple Bank object ("Control Account" and "Control Bank"). As a result neither of two roles was modeled accurately. In the diagram from figure 8 we do not have such a problem. It specifies only concepts necessary for reasoning about both roles of a Simple Bank System in corresponding contexts.

5 Conclusion

In this work we introduced a context-aware modeling framework. Our framework supports the explicit modeling of contexts. We believe that the explicit modeling of contexts makes system stakeholders' (users, developers, etc) reasoning about models easier. Due to easier reasoning, our framework is useful for business process modeling where a business analyst needs to get an immediate feedback on models. We explained how our framework can be used for the business process modeling. Our framework allows for the modeling of all elements of a business process (goals, activities, roles, objects and context).

Our framework is based on a role modeling technique proposed in [13] and adopted for business process modeling in [1]. In the context of role modeling, our contribution is that we have shown that a context can be modeled as a role model, where a role represents a subset from the complete set of attributes and actions for the whole model.

Our belief, that the proposed framework makes the reasoning about models easier, is based on our practical use of this framework for undergraduate courses. To make a quantitative analysis, we are planning to work out a questioning that can show the comparison of the human reasoning efficiency of our framework comparing with other modeling frameworks that use "UML-like" visual languages.

We expect the best output of our modeling framework when using a case tool, which will be the goal of our future work. This case tool must support the specification of a business process as a composition of smaller models, such that each of them has its own goal and is small enough to get immediate feedback on it.

6 References

[1] Balabko, P., Wegmann, A., "A Synthesis of Business Role Models", Proc. of *ICEIS conference*, 2003, Anger, France.

[2] Bouquet, P., et al., "Theories and Uses of Context", *Knowledge Representation And Reasoning*, 2001, technical report, Centro Per La Ricerca Scientifica e Tecnologica: Povo (Trento), retrieved from http://www.itc.it

[3] Carlsen, S., *"Conceptual Modeling and Composition of Flexible Workflow Models"*, Norwegian University of Science and Technology, PhD Thesis, 1997.

[4] Checkland, P., *"Systems Thinking, Systems Practice"*, Chichester, UK: Wiley, 1999.

[5] Chang, S.K., et al. "The Future of Visual Languages", Proc. *IEEE Symposium on Visual Languages*, 1999, Tokyo, Japan.

[6] Collins-Cope, M., *"Object Oriented Analysis and Design using UML"*, white paper, 1998, Ratio Group Ltd.: London, retrieved from http://www.ratio.co.uk/white.html

[7] Dembski, W.A., "The Fallacy of Contextualism", *The Princeton Theological Review*, 1994, retrieved from http://www.arn.org/docs/dembski/

[8] Giunchiglia, F., *"Contextual Reasoning"*, Technical Report #9211-20, University of Trento, 1993.

[9] ISO/IEC, (1996). 10746-1 | ITU-T Recommendation, *"Open Distributed Processing - Basic Reference Model - Part 2: Foundations"*.

[10] Kueng, P., et al., "How to compose an Object-Oriented Business Process Model?" Proc. *IFIP Conference on Method Engineering*, 1996

[11] Motschnig-Pitrik, R., "Contexts as means to decompose Information Bases and represent relativized Information", Proc. *CHI Workshop #11: The Who, What, Where, When, Why and How of Context-Awareness*, 2000, Hague, Netherlands.

[12] Ould M. A., "*Business Processes: Modeling and analysis for re-engineering and improvement*", John Wiley & Son, 1995

[13] Reenskaug, T., et al., "*Working With Objects: The OOram Software Engineering Method*". ed: Manning Publication Co, 1996

[14] Singh, B. & Rein, G.L., "*Role Interaction Nets (RINs): A Process Description Formalism*", Technical Report #CT-083-92, MCC, 1992.

[15] Wisse, P., "*METAPATTERN: information modeling as enneadic dynamics*", 2001, Technical report, University of Amsterdam: Amsterdam, Netherlands, retrieved from http://primavera.fee.uva.nl/html/

[16] Wegmann A., "The Systemic Enterprise Architecture Methodology (SEAM)", Proc. *ICEIS conference*, 2003, Anger, France

[17] Workflow Management Coalition, "*The Workflow Reference Model*", January 1995, retrieved from: http://www.wfmc.org/standards/standards.htm

[18] Vassallo, N., "Contexts and Philosophical Problems of Knowledge", Proc. *CONTEXT conference*, 2001, Springer-Verlag.