Supporting Business Process Compliance in Financial Institutions – A Model-Driven Approach

Jörg Becker
ERCIS – University of Muenster
Leonardo-Campus 3
48149 Münster, Germany
+49 (251) 83 38100
becker@ercis.de

Philipp Bergener
ERCIS – University of Muenster
Leonardo-Campus 3
48149 Münster, Germany
+49 (251) 83 38067
bergener@ercis.de

Patrick Delfmann
ERCIS – University of Muenster
Leonardo-Campus 3
48149 Münster, Germany
+49 (251) 83 38083
delfmann@ercis.de

Mathias Eggert
ERCIS – University of Muenster
Leonardo-Campus 3
48149 Münster, Germany
+49 (251) 83 38084
eggert@ercis.de

Burkhard Weiß
ERCIS – University of Muenster
Leonardo-Campus 3
48149 Münster, Germany
+49 (251) 83 38089
weiss@ercis.de

ABSTRACT

Recently, several approaches have been developed to check process models for compliance with laws and regulations. In this paper a contribution is made with respect to reducing the complexity of compliance checking by partially automating business process compliance (BPC) checking. We present a model checking approach that is able to check process models for BPC. In particular, we apply a generic pattern matching approach to the Semantic Business Process Modeling Language (SBPML) allowing for extended model checking not being restricted to predecessor-successor relationships. Finally, we apply the BPC checking approach to the example of a credit approval process from a real-world bank scenario using a demonstrator modeling software.

Keywords

Business Process Compliance, Financial Sector, Banks, BPM, Compliance, Model Checking, Pattern Matching

1. INTRODUCTION

The financial crisis has demonstrated impressively how difficult it is to adhere to legal regulations and internal as well as external compliance requirements concerning business processes. One reason for the fatal failure of controlling and supervisory boards is the complexity of the surveillance subject. SOX, MiFID and MaRisk are just three examples of regulations that must be considered when designing and controlling bank processes. The financial crisis even aggravates the complexity by introducing tighter and more regulations for financial institutions [1-3]. A recent empirical study by ABDULLAH ET AL. [4] states that “the financial sector is the most highly regulated industry.” This trend towards more regulation has also had an impact on IS research. Throughout the last years, an increasing number of approaches to solve compliance issues were published in this research area [5].

The handling of complexity in business processes of financial institutions is a challenging task. According to MOORMANN ET AL. the most important complexity drivers are: [6]

- the high rate of business rule changes
- the heterogeneous and inconsistent business vocabulary
- the redundant documentation of processes and business rules.

The handling of these complexity drivers is the core requirement for modern business process model checking approaches and automated compliance checking alike.

Without automation support, compliance managers are no longer able to fulfill their function, which ultimately leads to an involuntary toleration of compliance violations within the enterprise. Thus banks seek new approaches that are capable of combining compliance modeling and checking necessities within a holistic business processes management approach [7]. The major goal of the approach introduced in this paper is to support compliance managers in financial institutions in designing and checking business process compliance. This goal will be reached by presenting a compliance model checking approach for the Semantic Business Process Modeling Language (SBPML) [8]. This language is specially tailored for the requirements of the financial sector and is already evaluated in real-world application scenarios [8]. In particular, we realize the specification and checking of compliance rules through a generic structural model pattern matching approach with a corresponding tool support.

The remainder of this paper proceeds as follows: The status-quo of business process compliance management will be introduced in Section 2. Section 3 introduces SBPML as well as the basic con-

10th International Conference on Wirtschaftsinformatik,
16th - 18th February 2011, Zurich, Switzerland

cepts of the pattern matching method used for process model checking. In Section 4 the compliance patterns as well as their application in a real-world scenario are demonstrated. After a discussion of the key findings, further potentials of the introduced approach are revealed in section 5.

2. RELATED WORK

The concept of business process compliance denotes the execution of certain processes that comply with a set of regulations [9]. KHARBILI ET AL. classify the implementation of control mechanisms in three time-dependent phases "Design-Time Compliance Checking", "Runtime Compliance Checking" and "Backward Compliance Checking" [10].

The first phase is related to modeling compliance rules and process models. Notable research in this field has been done by [11] and [12], who use a non-monotonic deontic logic, implemented in a formal rule language called FCL (Formal Contract Language) and petri-nets for process modeling. STIJIN & VANTHIENEN [13] present the logical language PENELOPE, which provides the ability to verify temporal constraints, taken from compliance requirements with respect to business processes. In contrast, WÖRZBERGER ET AL. [14] develop a language for visualizing compliance requirements. Their approach is called the Business Process Compliance Language (BPCL) and allows annotating compliance requirements to business processes. According to them, three requirement types have been identified: "inclusions" (A process must contain a particular activity), "existence" (the occurrence of activity A implies the occurrence of activity B) and "precedence" (the existence of activity A requires the existence of activity B, which must be a direct or indirect successor) [15]. SADIQ ET AL. [9] develop an approach that allows the annotation of control objectives to process models. They argue that the visualization of the relationship between compliance requirements and business processes is easier to understand. The control rules are described by using the Formal Contract Language (FCL) [9]. Within this approach, the identified controls are transformed into control rules and classified with particular control tags. These control tags combine control rules with process models. This allows for an (automatic) assignment to corresponding process elements [9]. LIU ET AL. [15] define a graphical language for expressing compliance rules in the BPEL (Business Process Execution Language) standard, and then transform processes using a pi-calculus approach as well as finite state machines to subsequently check these processes, using linear temporal logic. The approach is not developed particularly for a banking environment, but evaluated with a business process from a real-world banking scenario. They show that temporal constraints, such as activity A must be executed before activity B starts, can be modeled and checked. This approach can also be classified as a Runtime Compliance Checking method.

The second phase (Runtime Compliance Checking) is addressed by the approaches from [16], [17] and [15]. To enable an independent view on business objectives and compliance objectives in processes, NAMIRI AND STOJANOVIC [16] develop an abstract layer (semantic mirror) that contains internal controls and interacts with process models. The approach comprises three phases: "control design" (in which control rules are designed that can be added in the semantic mirror and the process activities), "recovery action design" (which refers to new control rules and counteractive measures that will be checked by process managers) and "business process execution" (that focuses on updating the semantic mirror with information from the process instances) [16, 18]. In the approach from LY ET AL. [17] formal definitions of the compliance requirements act as a basis for the development of a constraint repository, which stores all conditions for a process design. These constraints are used during runtime of an application to review process instances [17].

The third phase (Backward Compliance Checking) contains approaches that examine already run process instances. The approach from VAN DER AALST ET AL. [19] presents a model checker based on linear temporal logic and verifies if certain rules for process instances apply. Another backward-oriented approach is introduced by [20]. This technique checks the conformance of a given control flow process model and matches it with a certain process instance to show a violation or difference in a process execution [20].

Further research in the area of business process compliance was done by [21]. In order to support Sarbanes-Oxley internal controls, AGRAWAL ET AL. [21] present an approach to workflow modeling, active enforcement, workflow auditing, as well as anomaly detection. Their approach spans all three compliance checking phases from a workflow management perspective.

Since the modeling language and model checking approach presented in this paper will not concern runtime environment and backward compliance checking, it can be classified as an approach of design-time compliance checking. Our approach can be distinguished from the existing ones due to two core characteristics: First, the applied modeling language SBPML is specifically tailored to meet the requirements of business process modeling in the financial sector. It considers especially the vocabulary of the according domain and provides a semantic standardization avoiding ambiguous modeling. Second, the underlying pattern matching approach is generic and thus applicable to SBPML. Due to its generic nature, it is expected to be possible to develop arbitrary structural model patterns being able to represent any possible design-time compliance rule.

3. A BUSINESS PROCESS COMPLIANCE CHECKING APPROACH

3.1 SBPML

SBPML was developed with the aim to allow for a more efficient modeling and analysis of business process models in comparison to generic process modeling approaches [8]. This is implemented by using domain semantics (e.g. domain vocabulary) in the form of predefined and thus reusable process building blocks to model activities in banks [8]. The modeling notation consists of four views, comprising a process view ("how is a service delivered?"), a business object view ("what is processed or produced?"), an organizational view ("who is involved in the modeling process?") and a resource view ("what resources are used?").

The core construct of this language is a set of domain-specific process building block (PBB) types. A PBB represents a certain kind of activity within a banking process. PBBs are atomic as they are the lowest level of detail to model activities. They have a well-defined level of abstraction and are semantically specified by a domain concept. With PBBs, problems like naming conflicts during model comparison are avoided, because the name of a PBB is specified by the language designer rather than the modeler. Examples for PBB types are "Document / Information Comes In", "Perform a Formal Verification", "Enter Data into IT", or "Archive Document". To capture the properties of activities in detail, each PBB has a specific set of attributes. For example, a possible attribute for the PBB "Enter Data into IT" is "Duration". Attributes provide the core information for a subsequent process analysis. They establish a connection to the business object, organizational, and resource view.

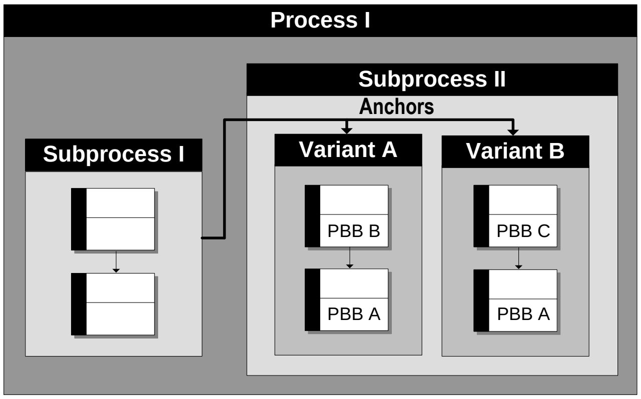


Figure 1. SBPML Process View Concepts

PBBs are part of the process view. By using PBBs sub-processes and processes are assembled. A process is the top level construct, delivering a service or a product. A process is divided into subprocesses with each sub-process representing the activities performed within a single organizational unit. A sub-process consists of sequential flows of PBBs. To model alternative flows of activities within a sub-process, multiple alternative sequences of PBBs can be defined, called variants. Each variant consists of one sequence describing the sub-process from beginning to end (cf. Figure 1). Hence, a PBB can occur in several variants of a subprocess. This sequential order restricts the degrees of freedom of the modeler and promotes the construction of structurally comparable process models, since they are linear on the variant level. This makes the models also easier to analyze in the context of compliance checking. Additional facts about processes, subprocesses and variants can be collected with the help of corresponding attributes.

To realize the control flow across boundaries of sub-processes the construct of an anchor is introduced. An anchor connects two PBBs across different sub-processes or processes (to connect core processes with support processes). This allows for modeling parallelism between sub-processes.

In the organizational view, the organizational structure is depicted by a hierarchy of organizational units. These can have job positions or roles assigned. To relate the organizational view to the process view, positions are annotated on the level of PBBs as activity operators. Organizational elements, as well as external partners (customers, business companies, and government institutions) are annotated on the level of PBBs as communication partners. Furthermore, each process is owned by an organizational unit and its sub-processes are operated by different organizational units.

A business object is either information, a document, or a material object. A resource is of a resource type, which is structured hierarchically. Both business objects and resources are annotated to PBBs to denote the process objects in and the needed resources for an activity.

3.2 Model Checking Approach

Assuring compliance in business process models requires considering two aspects:

- The models should contain sections that conform to business process compliance *requirements*. This means that certain structures *should be contained* in the models.
- The models should not contain sections that represent compliance *violations*. This means that certain structures *should not be contained* in the models.

Consequently, an appropriate model checking approach has to allow for *specifying patterns* of according structures and for *finding occurrences* of them in process models. As a result, models lacking required structures, as well as models including forbidden structures can be considered as candidates for compliance validations.

In this contribution, we make use of a generic pattern matching approach, which was available from a previous research project [22]. It is generic in terms of being applicable for multiple modeling languages. Since no special model checking approach currently exists for SBPML, a generic approach was an appropriate choice.

The idea of this approach is to apply set operations to a set of model elements, representing the model to be analyzed. Coming from graph theory, the approach recognizes any conceptual model as a graph G, consisting of vertices V and edges E, where $G=(V,E)$ with $E \subseteq V \times V$. Therefore, the approach distinguishes model objects, representing nodes, and model relationships, representing edges, interrelating model objects. Starting from a basic set that contains all model elements, the approach searches for pattern matches by performing set operations on this basic set. By combining different set operations, patterns are built up successively. Given a pattern definition, the matching process returns a set of model subsets representing the pattern matches found. Every match found is put into a separate subset.

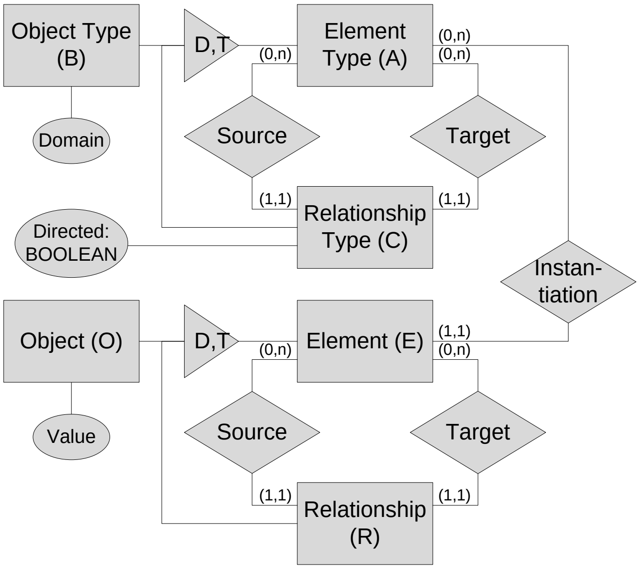


Figure 2. Generic Specification Environment for Conceptual Modelling Languages and Models

As a basis for the definition of model patterns, the approach makes use of a generic specification environment for conceptual modeling languages and models. The specification mainly consists of three constructs (cf. Figure 2). *Element types*, representing any atomic part of a model, are specialized as *object types* (i.e., model vertices) and *relationship types* (e.g., model edges and links). Each relationship type has a *source* element type from which it originates, and a *target* element type to which it leads. Relationship types are either *directed* or *undirected*. Whenever the attribute *directed* is *FALSE*, the direction of the relationship type is ignored. N-ary relationship types are represented as object types connected to n relationship types.

Particular model *elements* are instantiated from their distinct element type. They are specialized as *objects* and *relationships*. Each of the latter leads from a *source element* to a *target element*. Objects can (but do not need to) have *values*, which belong to a distinct *domain*, specified in the object type, to which the object belongs to. For example, the value of an object "name" contains the string of the name (e.g., "product"). As a consequence, the domain of the object's object type has to be "string" in this case. Thus, attributes are considered as objects.

Table 1. Object and Relation Types of SBPML

Relation Type	Source Object Type	Target Object Type	Cardinality
PBB Sequence	Abstract PBB	Abstract PBB	1:1
Anchor	Abstract PBB	Abstract PBB	1:1
SuprocessInProcess	Process	Subprocess	1:n
VariantInSubprocess	Subprocess	Variant	1:n
PBBInVariant	Variant	Abstract PBB	n:n
PPBAttribution	PBB	Attribute	n:n
AcivityOperator	Abstract PBB	Job Position	1:n
ProcessOwnership	Process	Orga.Unit	1:n
SubprocessExecution	Subprocess	Orga.Unit	1:n
CommunicationPartner	Abstract PBB	Orga.Element	n:n
ResourceUsage	Abstract PBB	Resource	n:n
HandledBusinessObject	Abstract PBB	BusinessObject	n:n
PBBInheritance	Abstract PBB	PBB Create Document	1:n
…	…	…	…

Table 1 shows how element types, relevant for the remainder of the paper, can be specified using this specification environment. Cardinalities are read in the way that e. g. a process is owned by one organizational unit while an organizational unit can own multiple processes. For reasons of brevity, the relation type between a PBB and a corresponding attribute is shown only once in a generic fashion instead of showing it for each individual PBB type. For the same reason, the inheritance relation between the abstract PBB object type and the PPB of type "Create Document" is given on a representational basis for all PBB types.

The pattern matching approach makes use of set operations, extracting elements, objects and relationships, with particular characteristics from the sets of the specification environment shown and thus builds up pattern matches successively. For example, such an operation could analyze all elements available and returns only process building blocks, being related to anchors. This exemplary pattern represents a change of the organization.

In the following, we introduce the available operations of the approach briefly. For a detailed formal specification cf. [22].

Each operation has a defined number of input sets and returns a resulting set, where the initial input sets used come from the specification environment (cf. abbreviations in the objects of Figure 2). In the explanation of the operations, we use additional sets (*X*: arbitrary set of elements; *Y*: arbitrary set of objects; *Z*: arbitrary set of relationships), specifying which kinds of inputs an operation expects. The first category of operations reveals specific properties of model elements (e.g., type, value, or domain):

- *ElementsOfType(X,a)* returns a set of all elements of *X*, belonging to the given element type *a*.
- *ObjectsWithValue(Y,value)* returns a set of all objects of *Y*, whose *values* equal the given one.
- *ObjectsWithDomain(Y,domain)* returns a set of all objects of *Y*, whose *domains* equal the given one.

In order to assemble complex pattern structures successively, the following operations combine elements and their relationships and elements, being related, respectively:

- *ElementsWithRelations(X,Z)* returns a set of sets containing all elements of *X* and their undirected relationships of *Z*. Each inner set contains one occurrence.
- *ElementsWithOutRelations(X,Z)* returns a set of sets containing all elements of *X* and their directed, outgoing relationships of *Z*. Each inner set contains one occurrence.
- *ElementsWithInRelations(X,Z)* is defined analogously to *ElementsWithOutRelations*. In contrast, it only returns incoming relationships.
- *ElementsDirectlyRelated* (X_1,X_2) returns a set of sets containing all elements of X_1 and X_2 that are connected directly via undirected relationships of *R*, including these relationships. Each inner set contains one occurrence.
- *DirectSuccessors* (X_1,X_2) is defined analogously to *ElementsDirectlyRelated*. In contrast, it only returns relationships that are directed, where the source elements are part of X_1 and the target elements are part of X_2.

A further category of operation is needed to build patterns representing recursive structures (e.g. a path of an arbitrary length):

- *{Directed}Paths*(X_1,X_n) returns a set of sets containing all sequences with undirected {directed} relationships, leading from any element of X_1 to any element of X_n. The elements that are part of the paths do not necessarily have to be elements of X_1 or X_n, but can also be of $E \backslash X_1 \backslash X_n$. Each path found is represented by an inner set.
- *{Directed}Loops(X)* is defined analogously to *{Directed} Paths*. It returns a set of sets containing all undirected {directed} sequences, which lead from any element of *X* to itself.

To avoid infinite sets, only finite paths and loops are returned. To provide a convenient specification environment for structural model patterns, we define some additional functions that are derived from those already introduced:

- *ElementsWith{In|Out}RelationsOfType(X,Z,c)* returns a set of sets containing all elements of *X* and their {un}directed, {incoming|outgoing} relationships of *Z* of the type *c*. Each occurrence is represented by an inner set.
- *ElementsWithNumberOf{In|Out}Relations(X,n)* returns a set of sets containing all elements of *X*, which are connected to the given number *n* of {un}directed {incoming|outgoing} relationships of *R*, including these relationships. Each occurrence is represented by an inner set.
- *ElementsWithNumberOf{In|Out}RelationsOfType(X,c,n)* returns a set of sets containing all elements of *X*, which are connected to the given number *n* of {un}directed {incoming|outgoing} relationships of *R* of the type *c*, including these relationships. Each occurrence is represented by an inner set.
- *{Directed}PathsContainingElements*(X_1,X_n,X_c) returns a set of sets containing elements that represent all undirected {directed} paths from elements of X_1 to elements of X_n, which

each contain at least one element of X_c. The elements that are part of the paths do not necessarily have to be elements of X_1 or X_n, but can also be of $E \backslash X_1 \backslash X_n$. Each such path found is represented by an inner set.

- *{Directed}PathsNotContainingElements(X_1,X_n,X_c)* is defined analogously to *{Directed}PathsContainingElements*. However, it returns only paths that do not contain any element of X_c.
- *{Directed}Loops{Not}ContainingElements(X,X_c) is defined analogously to {Directed}Paths{Not}ContainingElements.*

By nesting the functions introduced above, it is possible to build structural model patterns successively. The results of each function can be reused adopting them as an input for other functions. In order to combine different results, the basic set operators *union* ($\cup$), *intersection* ($\cap$), and *complement* (\) can generally be used. Since it should be possible to not only combine sets of pattern matches (i.e., sets of sets), but also the pattern matches themselves (this refers to the inner sets), the approach incorporates additional set operators. These operate on the inner sets of two sets of sets respectively.

The ***Join*** operator performs a *union* operation on each inner set of the first set with each inner set of the second set. Since we regard patterns as cohesive, only inner sets that have at least one element in common, are considered. The ***InnerIntersection*** operator *intersects* each inner set of the first set with each inner set of the second set. The ***InnerComplement*** operator applies a *complement* operation to each inner set of the first outer set combined with each inner set of the second outer set. Only inner sets that have at least one element in common are considered.

As most of the set operations introduced expect simple sets of elements as inputs, further operators are introduced that turn sets of sets into simple sets. The ***SelfUnion*** operator merges all inner sets of one set of sets into a single set performing a ***union*** operation on all inner sets. The ***SelfIntersection*** operator performs an ***intersection*** operation on all inner sets of a set of sets successively. The result is a set containing elements that each occur in all inner sets of the original outer set.

A simple exemplary pattern searching for two particular PBBs (named “Activity A” and “Activity B”), following each other over a path of arbitrary length, is specified as follows:

```
DirectedPaths(
   ObjectsWithValues(
      ElementsOfType(O,PBB),‟Activity A‟
   ),
   ObjectsWithValues(
      ElementsOfType(O,PBB),‟Activity B‟
   )
)≠∅
```

4. APPLYING COMPLIANCE CHECKING IN FINANCIAL BUSINESS PROCESSES

4.1 Example Case

We will demonstrate our approach on a real-life example case of a credit application process. The example was chosen as it represents a complex core banking process that is also one of the most well-researched processes in the financial industry, and can be found in the vast majority of banks. The process was modeled at a large German bank, which is specialized on providing consumer credits.

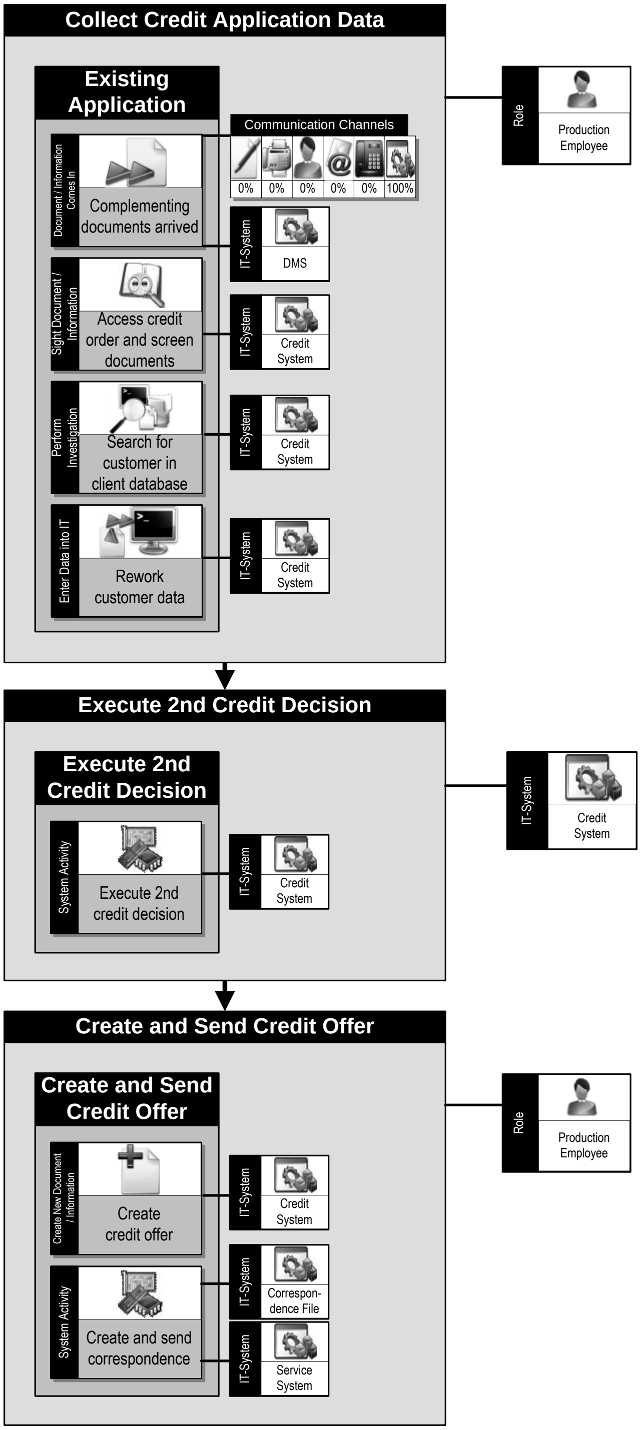


Figure 3. Credit Application Process Section in SBPML

The process is initiated by a credit application, arriving by postal service. This is scanned by an external service company and made available in the document management system of the bank. It arrives in the production department once the contractor sends an electronic message to the bank’s workflow management system. The bank employee searches for the customer in a database. It may either be that the customer is identified as an existing customer or that the new customer has to be registered in the database upfront. After data completion, the customer’s data needs to be approved in order to decide for an initial credit approval step. The approval can be supported by also taking data from an external credit rating agency regarding creditworthiness, if the client has approved this check beforehand.

Only if the first approval check is (semi-)positive the bank will check further documents such as the income statement or further obligations. Once the first approval step has been successful or semi-successful the second credit decision will be performed.

The second credit decision can again lead to a positive, semi-positive or negative decision. It is also possible that a second decision will be postponed due to a missing document that needs to be supplied, before a final decision can be made. In such cases, the process is restarted as soon as these documents come in. Again, a negative decision will lead to a credit order rejection. A positive decision will lead to the creation of a credit offer. Furthermore, the credit decision can be semi-positive due to contextual or technical problems.

Contextual problems can be any problems due to inconsistencies in the data that the customer has supplied and need to be settled directly with the client and possibly also with the credit rating agency. Errors will be corrected and a final credit decision will be initiated again. Technical problems are for problems with the IT system so that the second approval has to be performed again. Once all problems are solved, and the client is rated to be creditworthy, a credit offer will be issued. Figure 3 shows a section of this process, depicting the process applied to existing credit applications.

4.2 Compliance Rules

Compliance-related business rules can be categorized into four different types called tags [9, 23]: i) flow tags represent rules regarding the business process control flow and thus the execution of certain activities (e.g., order of activities, existence of certain activities etc.), ii) time tags represent rules depicting temporal conditions or restraints within process flows (e.g., maximum time that may be needed to respond to a customer request), iii) resource tags represent rules regarding the resources used when executing activities (e.g., authorization rules for IT systems or separations of duties within a process flow), iv) data tags represent rules regarding the (business object) data used throughout a process (e.g., special checks if a credit amount is higher than a certain amount).

In terms of SBPML, this means that there are business rules that refer to the process view solely (flow tags and time tags), the business object view, possibly in conjunction with the process view (data tags), and the resource view as well as the organizational view, possibly in conjunction with the process view (resource tags). Below we give a graphical representation of the different compliance rules for banks, which were derived from the initial literature review. We follow the notation developed by Awad and Weske [24], describing process control flow business rules for BPMN, but use the elements of SBPML [23]. Since time tags can only be evaluated during run-time, they will not be considered further.

According to [24] control flow business rules define the sequence in which activities may or should be performed. As general concepts, predecessor relations (Activity A “leads to” Activity B) and successor relations (Activity A “precedes” Activity B) are introduced. Furthermore, there are existence or non-existence constraints. In addition, depending upon an activity’s position within a process or sequence of activities, different scopes can be distinguished. The sequence as well as the existence or non-existence of activities is defined within the “scope” of a process. The scope of a constraint can either be “global”, or with respect to another activity “before” or “after” that activity, or with respect to two other activities following each other.

In Figure 4 (a) Activity A must be contained somewhere in the process whereas in (b) Activity A may not be part of the entire process. (c) describes the classical predecessor constraint and (e) the successor constraint.

For example, the predecessor constraint, depicted in (c), is specified as follows using the pattern matching approach:

First, it has to be checked whether an Activity B exists (i.e., the returned element set of the following pattern specification must not be empty):

```
ObjectsWithValues(
   ElementsOfType(O,PBB),“Activity B“
)≠∅
```

If Activity B exists, then a path from Activity A to Activity B has to exist in order to satisfy the predecessor constraint (i.e., the returned element set of the following pattern specification must not be empty):

Should the second pattern search return no result, then a compliance violation is detected. If Activity B does not exist, the second check is not necessary, as then it is not possible to violate a predecessor constraint related to Activity B (this applies analogously for the following patterns).

In (d) Activity A may not be executed before Activity B is finished; in (f) Activity B may not be executed after Activity A is finished. (g) and (h) describe the non-existence constraint of Activity B between Activity A and Activity C, with Activity A and Activity C either in a predecessor or successor relation.

For example, the non-existence constraint of Activity B between Activity A preceding Activity C, which is depicted in (h), is specified as follows using the pattern matching approach:

First, it has to be checked whether an Activity C exists (i.e., the returned element set of the following pattern specification must not be empty):

```
ObjectsWithValues(
   ElementsOfType(O,PBB),“Activity C“
)≠∅
```

If Activity C exists, then there has to be a path from Activity A to Activity C that is in turn not allowed to contain Activity B. (i.e., the returned element set of the following pattern specification must not be empty):

```
DirectedPathsNotContainingElements(
   ObjectsWithValues(
      ElementsOfType(O,PBB),“Activity A“
   ),
   ObjectsWithValues(
      ElementsOfType(O,PBB),“Activity C“
   ),
   ObjectsWithValues(
      ElementsOfType(O,PBB),“Activity B“
   )
)≠∅
```

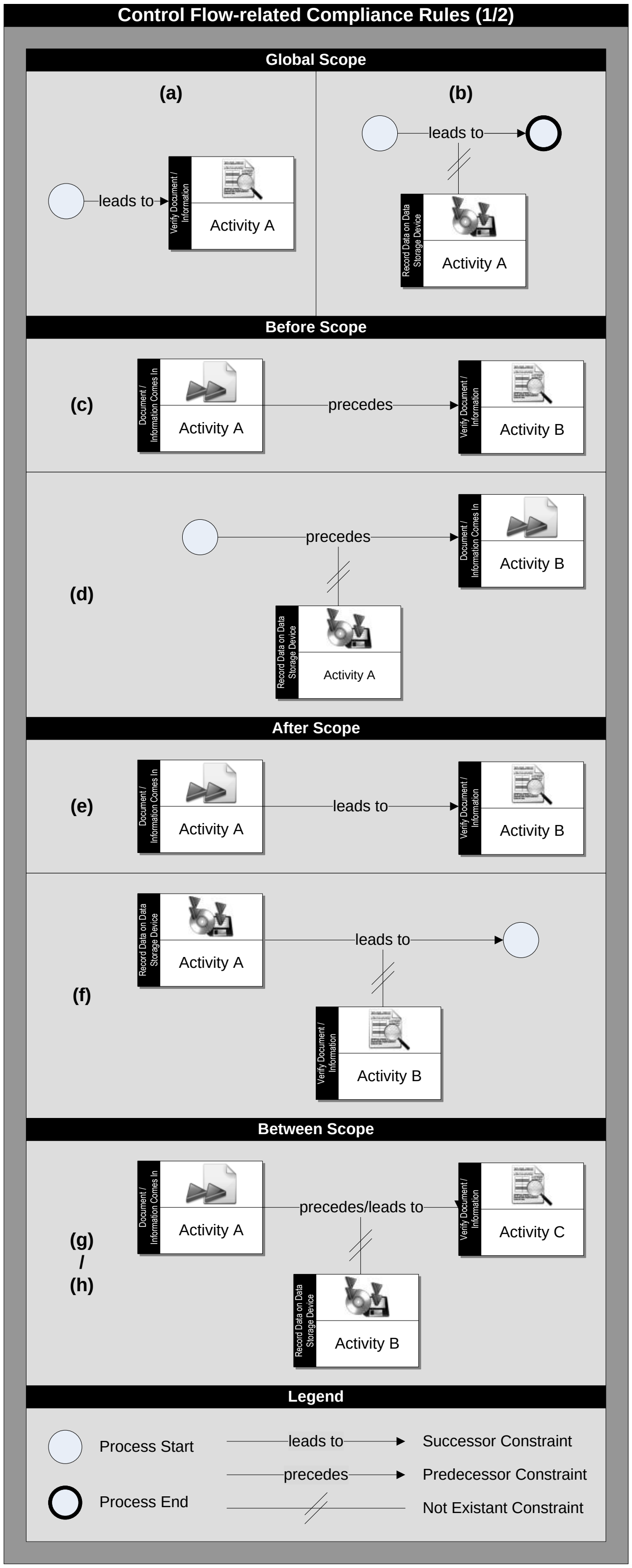

Control Flow-related Compliance Rules (1/2)
Global Scope
(a)
leads to
Verify Document / Information
Activity A
(b)
leads to
Record Data on Data Storage Device
Activity A
Before Scope
(c)
Document / Information Comes In
Activity A
precedes
Verify Document / Information
Activity B
(d)
precedes
Document / Information Comes In
Activity B
Record Data on Data Storage Device
Activity A
After Scope
(e)
Document / Information Comes In
Activity A
leads to
Verify Document / Information
Activity B
(f)
Record Data on Data Storage Device
Activity A
leads to
Verify Document / Information
Activity B
Between Scope
(g) / (h)
Document / Information Comes In
Activity A
precedes/leads to
Verify Document / Information
Activity C
Record Data on Data Storage Device
Activity B
Legend
Process Start
Process End
leads to
Successor Constraint
precedes
Predecessor Constraint
Not Existant Constraint

Figure 4. SBPML Flow Tags (1/2)

In (i) and (j) we use a “variable activity” PBB, which stands for a PBB of an arbitrary type, to define direct sequences (cf. Figure 5). In (j) Activity A must be a direct predecessor of Activity B or vice versa Activity B must be a direct successor of Activity A.

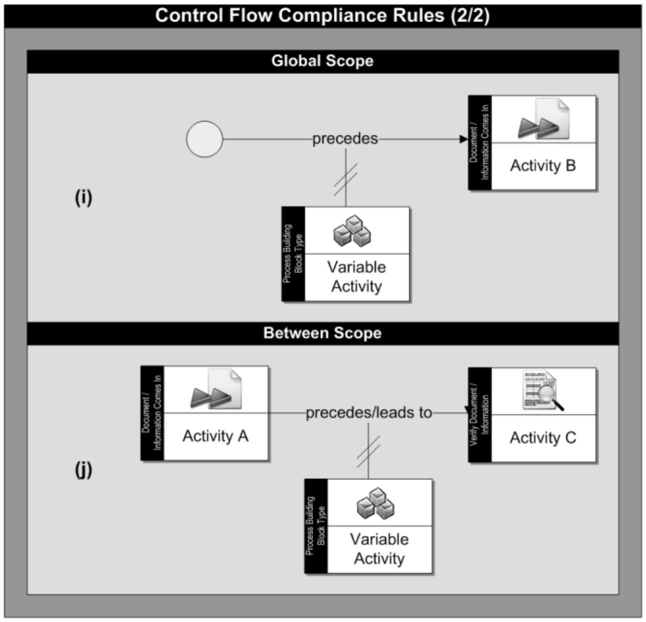

Control Flow Compliance Rules (2/2)
Global Scope
(i)
precedes
Document / Information Comes In
Activity B
Process Building Block Type
Variable Activity
Between Scope
(j)
Document / Information Comes In
Activity A
precedes/leads to
Verify Document / Information
Activity C
Process Building Block Type
Variable Activity

Figure 5. SBPML Flow Tags (2/2)

For example, this compliance rule is specified as follows using the pattern matching approach:

First, it has to be checked whether Activity A exists (or alternatively, whether Activity B exists) (i.e., the returned element set of the following pattern specification must not be empty):

```
ObjectsWithValues(
   ElementsOfType(O,PBB),“Activity A“
)≠∅
```

If Activity A (or alternatively, Activity B) exists, then Activities A and B have to be direct successors (i.e., the returned element set of the following pattern specification must not be empty):

```
DirectSuccessors(
   ObjectsWithValues(
        ElementsOfType(O,PBB),“Activity A“),
   ObjectsWithValues(
        ElementsOfType(O,PBB),“Activity B“)
)≠∅
```

In (i), we define that Activity A must be the first activity within an entire process, since no other activity is allowed to precede it. Similarly, one could also predefine the last activity that must be at the end of a process. All rules introduced so far may not only be applied to activities in the SBPML notation, but also to processes, sub-processes and sub-process variants. Furthermore, more complex patterns can be derived through the combination of these simple patterns.

From a resource tag based view (corresponding organizational view and resource view in the SBPML terminology), further rules can be specified. Focusing on the organizational view of the SBPML terminology, there are two further very common compliance requirements, which need to be captured by business rules. These are the application of a four eyes principle (cf. Figure 6 (l)), where one person executes Activity A and a second person verifies if Activity A was done correctly, and the aspect of separation of duties (cf. Figure 6 (k)) denoting that certain activities have to be performed by different persons. This is also possible on the level of processes, sub-processes and variants.

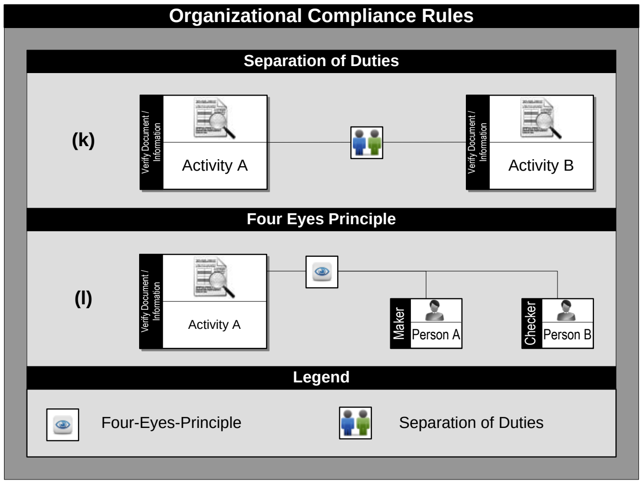


Figure 6. SBPML Resource Tags

For example, the separation of duties compliance rule is specified as follows using the pattern matching approach:

First, it has to be checked whether Activity A and Activity B exist succeeding each other (i.e., the returned element set of the following pattern specification must not be empty):

```
DirectedPaths(
   ObjectsWithValues(
        ElementsOfType(O,PBB),“Activity A“),
   ObjectsWithValues(
        ElementsOfType(O,PBB),“Activity B“)
)≠∅
```

If succeeding Activities A and B exist, then they have to be assigned to different persons, or more generally speaking, organizational entities (i.e. the returned element set of the following pattern specification must not be empty):

```
DirectedPaths(
   ElementsOfType(O,PBB)
   INNER_INTERSECTION
   ElementsDirectlyRelated(
      ObjectsWithValues(
         ElementsOfType(O,PBB),“Activity A“),
      ObjectsWithValues(
         ElementsOfType(
            O,Organization_Attribute),
         Org1)),
   ElementsOfType(O,PBB)
   INNER_INTERSECTION
   ElementsDirectlyRelated(
      ObjectsWithValues(
         ElementsOfType(O,PBB),“Activity B“),
      ObjectsWithValues(
         ElementsOfType(
            O,Organization_Attribute),
         Org2))
)≠∅;     Org1≠Org2
```

Finally, compliance rules can also be modeled regarding the data tags using the corresponding business object view in SBPML (cf. Figure 7). Following ZOET ET AL. [25] rules for “effect sequencing” (m) are defined by describing that business objects with certain characteristics imply further activities to be executed (e.g. credit applicants applying for credits worth more than 75,000 € must receive an additional positive vote inside a bank). Analogous rules can also be defined to relate such business objects to processes, sub-processes, and variants.

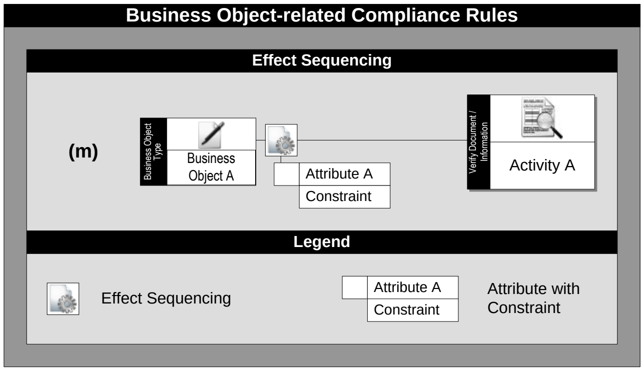


Figure 7. SBPML Data Tags

For example, a compliance rule, requiring the assignment of business objects with specific characteristics to specific activities, is specified as follows using the pattern matching approach:

First, it has to be checked whether a business object exists that is related to an attribute, whose value describes a specific characteristic (i.e., the returned element set of the following pattern specification must not be empty):

```
ElementsDirectlyRelated(
   ElementsOfType(O,Business_Object),
   ObjectsWithValues(
      ElementsOfType(O, Attribute),
      Constraint)
)≠∅
```

As the characteristic could be anything, we indicate this characteristic by the term “constraint” in the example. If there is a business object having this characteristic, it has to be checked if it has been assigned to the required activity – in this case Activity A (i.e., the returned element set of the following pattern specification must not be empty):

```
ElementsDirectlyRelated(
   ObjectsWithValues(
      ElementsOfType(O, Attribute),
      Constraint),
   ObjectsWithValues(
      ElementsOfType(O,PBB),
      ”Activity A”)
)≠∅
```

4.3 Compliance Checking

To apply and evaluate the approach, we developed a prototypical implementation. As the underlying pattern matching approach is generic, we chose a meta modeling tool as an implementation basis, which was available from a previous research project. This way, we only had to define SBPML using the meta modeling environment of the tool, rather than implementing a new tool. The pattern matching approach was implemented as a plug-in, accessing the model data base of the tool.

As an application scenario, we modeled the credit application process, introduced in Section 4.1, using SBPML. The compliance rules, introduced in Section 4.2, were defined using the pattern matching plug-in.

The left hand side of Figure 8 depicts the tool’s definition environment for compliance rules. The left hand frame shows the pattern tree of the compliance rule “separation of duties” (k). The right hand frame outlines the variable specification area (i.e., in this area the variable constraint “ORGA==ORGB” is defined). The application of this compliance rule to an exemplary section of the credit application process leads to highlighting those PBBs that follow each other and that require and comply with the separation of duties (cf. right hand side of Figure 8). In the case of a compliance rule violation, the pattern matching instance would have returned no result and so would have indicated the violation.

5. DISCUSSION AND OUTLOOK

The approach introduced in this paper applies a generic structural model pattern matching approach to SBPML to address the problem of business process compliance checking in the financial sector. Thus, we combined the advantage of a semantically standardized modeling language with the expressive power of generic model checking. The pattern matching approach enables us not only to specify arbitrary compliance rules, not restricted to temporal relationships, but also avoids problems occurring in model checking due to semantic ambiguities. Hereby, we addressed the three complexity drivers identified by MOORMANN ET AL. [6]:

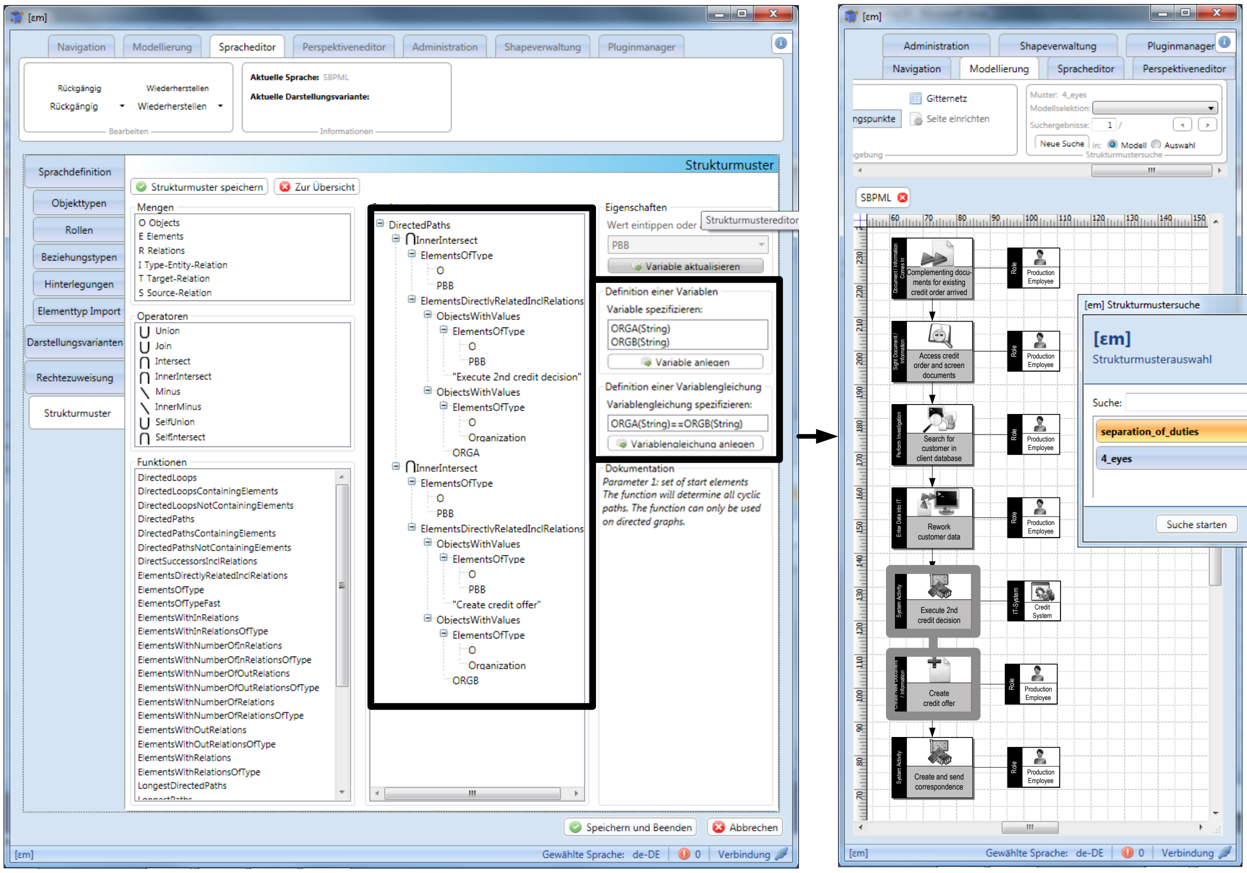
High rate of business rule changes: Because business rules and legal requirements are changing frequently, it is necessary to make it as simple as possible to identify process elements that contain a particular law or business rule. Through the presented model checking approach, it is possible to define and to check nearly all possible business constraints that may emerge over time. Furthermore, the rules can be managed independently from the process models.

Heterogeneous and inconsistent business vocabulary: Complexity arises when compliance and business process managers develop processes and constraints separately. This is addressed by the semantic standardization provided by the building block concept.

Redundant documentation of processes and business rules: The last requirement is addressed through separated model and compliance rule management, based on a common specification environment. A change of the model does not necessarily imply a change of the compliance rules and vice versa. Only if compliance rules are violated, the models have to be changed.

However, the approach requires an extensive evaluation. Besides implementing the language in a tool and the exemplary application of one financial business process, more processes must be analyzed to get a final proof for the validity of the first evaluation results. Although the relevance was confirmed in preliminary discussions with compliance experts in the financial sector, the support of compliance management experts through the introduced method must be shown in detail through qualitative and quantitative studies. Furthermore, we have to question the technical efficiency of the model checking approach. The results of our exemplary applications to selected process models showed a satisfactory efficiency. However, it is necessary to apply the approach as well to large-scale scenarios. This applies especially as the graph pattern matching problem is known to be NP-hard. Moreover, in order to further reduce semantic ambiguities, we plan to support also the free text fields of SBPML through semantic standardization (e.g., proposed in the research areas of ontologies and computational linguistics).

Through the tool implementation of SBPML and its model checking approach it is now possible to evaluate hypotheses regarding a better efficiency of compliance management. First evidence from



Parameter 1: set of start elements
The function will determine all cyclic paths. The function can only be used on directed graphs.

Figure 8. Tool-supported Business Process Compliance Checking

accompanying research to this paper in conjunction with compliance experts suggests that the complexity of compliance management will be better handled and thus the effort for compliance checking will be lower by using the presented approach.

Both the created modeling language, using the business rules and pattern matching approach, as well as an appropriate tool support make a contribution to compliance management. To create and evaluate hypotheses for an increasing efficiency and law-conformity in compliance management, methods are needed that will allow for modeling and model analysis of business processes from a legal perspective. With the presented work a first essential step for reaching this goal is done.

6. REFERENCES

[1] Caldwell, F. 2009. The Worldwide Economic Crisis will Bring Real-Time Reporting for Risk Management. Gartner Research, Gartner, Inc.

[2] Caldwell, F., Bace, J., and Lotto, R. J. D. 2009. U.S. Financial System Regulatory Overhaul Brings More Scrutiny. Gartner Research. Gartner, Inc.

[3] Opromolla, G. 2009. Facing the Financial Crisis: Bank of Italy's Implementing Regulation on Hedge Funds. Journal of Investment Compliance 10, 2, 41-44.

[4] Abdullah, S. N., Sadiq, S., and Indulska, M. 2010. Emerging Challenges in Information Systems Research for Regulatory Compliance Management. In Proceedings of the CAISE.

[5] Abdullah, S. N., Indulska, M., and Sadiq, S. 2009. A Study of Compliance Management in Information Systems Research. In Proceedings of the ECIS.

[6] Moormann, J., Vetter, D., and Hilgert, M. 2009. Die Komplexität reduzieren. die bank 11/2009.

[7] Becker, J., Breuker, D., Weiss, B., and Winkelmann, A. 2010. Exploring the Status Quo of Business Process Modelling Languages in the Banking Sector – An Empirical Insight into the Usage of Methods in Banks. In Proceedings of the ACIS.

[8] Becker, J., Weiß, B., and Winkelmann, A. 2009. Developing a Business Process Modeling Language for the Banking Sector - A Design Science Approach. In Proceedings of the AMCIS.

[9] Sadiq, S., Governatori, G., and Namiri, K. 2007. Modeling control objectives for business process compliance. Business Process Management, 149-164.

[10] Kharbili, M. E., de Medeiros, A., Stein, S., and van Der Aalst, W. M. P. 2008. Business Process Compliance Checking: Current State and Future Challenges. Lecture Notes in Informatics: Modellierung betrieblicher Informationssysteme (MobIs) 141, 107-113.

[11] Governatori, G., Hoffmann, J., Sadiq, S., and Weber, I. 2008. Detecting regulatory compliance for business process models through semantic annotations. In Proceedings of the Workshop on Business Process Design.

[12] Governatori, G., and Rotolo, A. 2010. A Conceptually Rich Model of Business Process Compliance. In Proceedings of the APCCM.

[13] Stijin, G., and Vanthienen, J. 2006. Designing Compliant Business Processes with Obligations and Permissions. In Proceedings of the BPM 2006 Workshops, LNCS 4103.

[14] Wörzberger, R., Kurpick, T., and Heer, T. 2008. Checking Correctness and Compliance of Integrated Process Models. In Proceedings of the 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 576-583.

[15] Liu, X., Müller, S., and Xu, K. 2007. A static compliance-checking framework for business process models. IBM Systems Journal 46, 2, 335-361.

[16] Namiri, K., and Stojanovic, N. 2007. Pattern-based Design and Validation of Business Process Compliance. In On the Move to Meaningful Internet Systems 2007: CoopIS,DOA, ODBASE, GADA, and IS 2007, R. Meersman,Z. Tari, Eds. Springer, Berlin Heidelberg.

[17] Ly, L. T., Rinderle-Ma, S., Göser, K., and Dadam, P. 2010. On enabling integrated process compliance with semantic constraints in process management systems. Information Systems Frontiers, in press.

[18] Namiri, K., and Stojanovic, N. 2007. A Formal Approach for Internal Controls Compliance in Business Processes. In Proceedings of the 8th Workshop on Business Process Modeling, Development and Support (BPMDS07). 1-9.

[19] van der Aalst, W. M. P., de Beer, H. T., and van Dongen, B. F. 2005. Process Mining and Verification of Properties: An Approach Based on Temporal Logic. In Proceedings of the OTM Conferences (1), 130-147.

[20] Rozinat, A., and van der Aalst, W. M. P. 2008. Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33, 1, 64-95.

[21] Agrawal, A., Johnson, M., Kiernan, J., and Levmann, F. 2006. Taming compliance with Sarbanes-Oxley internal controls using database technology. In Proceedings of the 22nd International Conference on Data Engineering.

[22] Delfmann, P., Herwig, S., Lis, L., Stein, A., Tent, K., and Becker, J. 2010. Pattern Specification and Matching in Conceptual Models. A Generic Approach Based on Set Operations. Enterprise Modelling and Information Systems Architectures 5, 3, in press.

[23] Becker, J., Ahrendt, C., Coners, A., Weiss, B., and Winkelmann, A. 2010. Business Rule Based Extension of a Semantic Process Modeling Language for Managing Business Process Compliance in the Financial Sector. Proceedings of the MobIs Conference. LNI 175, 201-206.

[24] Awad, A., and Weske, M. 2009. Visualization of Compliance Violation in Business Process Models. In Proceedings of the 5th Workshop on Business Process Intelligence, 1-12.

[25] Zoet, M., Welke, R., Versendaal, J., and Ravesteyn, P. 2009. Aligning Risk Management and Compliance Considerations with Business Process Development. Lecture Notes in Computer Science 5692, 157-168.