Rule Determination and Process Verification Using Business Capabilities

Thomas Stuht[1], Andreas Speck[2], Sven Feja[2], Sören Witt[2], and Elke Pulvermüller[3]

[1] PPI AG, D-22301 Hamburg, Germany,
Thomas.Stuht@ppi.de
[2] Christian-Albrechts-University of Kiel, Institute of Computer Science, D-24098 Kiel, Germany,
{aspe,svfe,swi}@informatik.uni-kiel.de
[3] University of Osnabrueck, Department of Mathematics and Computer Science, D-49076 Osnabrück, Germany,
elke.pulvermueller@informatik.uni-osnabrueck.de

Abstract. Business architectures are an important part of any enterprise architecture containing business processes and business capabilities. High quality business processes are key factors for the success of a company. Hence, the quality and the correctness or compliance have to be verified. We propose to use the business capabilities for an efficient and easily understandable definition of rules to perform this verification. The rule specification is based on rule patterns to define requirements from an operational point of view. These patterns are derived from experience gained in projects for modeling and optimization of business processes with extensive manual checks. For the rule validation we rely on model checking as an established technology to cope with the dynamic properties of processes. We present a tool based approach to automate this verification integrated in a unique system with a common user interface.

Key words: enterprise architecture, business capabilities, business process model quality and correctness or compliance, integrating verification technique, rule patterns

1 Introduction

In enterprise architectures process models are an essential part to document the business processes. In that sense the business processes are besides the data models crucial for the success of an enterprise. Business process models are the base of the implementation of systems. This important role of business processes (and their models) in enterprises and enterprise architectures lead to the question of the quality and correctness or compliance.

The required checks of the business process models are often performed manually by modeling experts and domain experts. This is very time-consuming and prone to errors, if not done carefully. A more efficient way is to transform the implicit knowledge used by the experts to perform the checks into rules. The rules are suitable for automatic testing using a tool. However, common approaches require detailed rules described at the activity level. Approaches allowing for a higher degree of reusability would reduce the effort of rule specification.

Business architectures as part of enterprise architectures are agreed by the responsible persons within the company. They include not only the business processes, but also the business capabilities. These capabilities represent a harmonized starting point for clustering activities from an operational point of view.

The proposed approach is based on the experience of process management projects in the financial sector mainly at small and medium insurance companies. The findings analyzing the frequent manual checks are incorporated in this paper.

The examples we present in this paper are of the financial systems domain. Certainly, we observed similar challenges in other domains such as e-commerce or enterprise resource planning systems.

2 Business Architecture as Base of Compliance Rules

In order to assure the conformity and correctness of business processes in enterprise architectures we require the specification of rules as base of the process verification. The base of our approach is the efficient and easily understandable definition of these rules from an operational point of view. Furthermore, the reusability of rule specifications is a key aspect of our approach. Hence, a rule can be used for more than one process only by abstracting from the level of the single activities. Thus it does not depend on the actual names of the activities. But it is still possible to use specific labels in a rule description if necessary.

We propose to use a so-called "operational architecture" representing the business capabilities of a business architecture to support an architectural abstraction as used in section 3. The capabilities are an integral part of an enterprise architecture and provide an operational structuring of business processes [1, 2].

There is no single definition of the term "business capability". The following explanation describes the term how it is used in this paper. A definition that is consistent with our understanding is given in the architecture framework MODAF. Capabilities "are a high level specification of the enterprise's ability. A capability is a classification of some ability - and can be specified regardless of whether the enterprise is currently able to achieve it." [3, p. 1] Therefore, a business capability clusters of one or more actions that produce a result or have an effect. But it does not describe how the actions take place. A business capability abstracts from the activities at the lowest level of detail.

2.1 Operational Architecture Structure

Concrete operational architectures and business capabilities may be developed by an association of companies (e.g. the German Insurance Association (GDV)[1]) or by specific companies. Because of these different creators the key aspects of their

[1] http://www.gdv.de

architectures may vary. For example, the “operational architecture” as a part of the more comprehensive “insurance applications architecture” (in German: “VersicherungsAnwendungsArchitektur”) of the GDV contains the characteristics of business processes and business components [4]. Other descriptions we are concerned with in projects focus also on IT aspects (e.g. systems) or workflows.

Our real-life example of business capabilities structured in an operational architecture is the one developed by the PPI AG for the insurance sector. It has been proven as a reference model in several projects (e.g. [5]). We now apply it as base for checking the operational compliance of business processes.

The PPI operational architecture structures the functionality an insurance company offers based on a three-level hierarchy. It does neither contain IT details (software, hardware) nor business process details (sequences, events). Figure 1 shows an excerpt of the architecture and illustrates each level explained below.

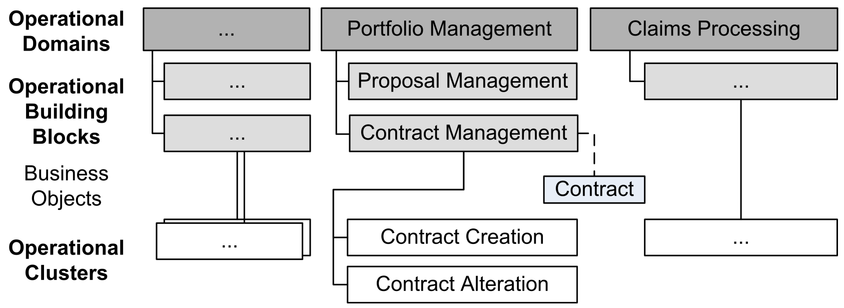


Fig. 1. Excerpt from the operational architecture showing the three-level hierarchy of the business capabilities

The top level (“Operational Domains”) contains the core business functions of an insurance company, e.g. “Portfolio Management” and “Claims Processing”. The next level (“Operational Building Blocks”) concretizes the top level elements and contains the building blocks from an operational point of view. For example, “Proposal Management” and “Contract Management” belong to the top level function “Portfolio management”. This level also defines the set of business objects that are assigned to their corresponding operational building block, i.e. “Contract” belongs to “Contract Management”.

The third level (“Operational Cluster”) is a set of conceptual functions (i.e. the business capabilities) the company has to perform. Examples of operational clusters for the building block “Contract Management” are “Contract Creation” or “Contract Alteration”. The combination of these reusable clusters visually spoken builds a business process. A process can also be represented by an operational cluster because it is a function performed by the company.

The relevant stage for the proposed abstraction from the activity level is the third level. As shown in figure 2 multiple activities used in one or more process models can be conceptually connected with one operational cluster (“n:1”). Note that an activity is always related to exactly one constant operational cluster to guarantee uniqueness. This allows abstracting from activities with slightly different names that represent an identical function. Besides it is possible to sum up activities and use the abstract operational cluster instead. This abstraction mechanism is used in section 4 to define the rule patterns.

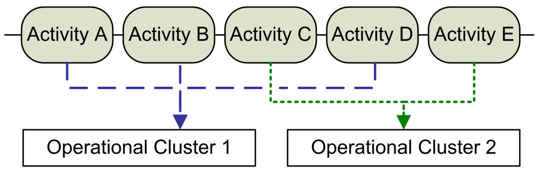


Fig. 2. Mapping of concrete activities to operational clusters

2.2 Rules Selection Using Scopes

In addition to the clustering we propose a second mechanism for reusing rule specifications: the definition of scopes for a rule. This allows automatically selecting the relevant rules for checking as depicted in the example in section 6.

Each process model is characterized by its type and the section the process is applied in. The type should be based on a standardized set of type names like the third level of the operational architecture. Based on the intension of the operational architecture an operational cluster represents a whole business process. For example a process can have the type “Contract Creation”.

The section depends on how the company is organized. Typical sections of German insurance companies are “personal liability” or “household contents”. This would be a categorization using the offered classes of insurance. In addition, supporting areas such as collections and disbursement or fraud could be used.

For each rule it may be specified for which process types the rule is applicable. The rule is only applied if the process takes place in one of the denoted sections.

Thus it is possible to define universal rules, which can be applied to different processes. For example a rule related to “Contract Creation” can be specified and no individual rules for “Contract Creation for Personal Liability” or “Contract Creation for Household Contents” are required. Only if there are special requirements, a distinct rule must be specified. Another example is the inbox for all sorts of communication (e.g. mail, e-mail, fax). A general rule can be used for verifying all processes where the inbox is involved.

3 Verification Technique for Business Process Models

In the previous section we present the key concepts of our approach to specify rules for operational conformity checking. The verification requires a suitable technique like *model checking*. First research concerning the verification of concurrent programs was done 30 years ago [6]. Subject of the model checking is to verify automatically if a model fulfills a rule [7, 8]. If it does not, a counterexample indicate the part which violates the rule [6, 9].

A rule can be considered as a machine-readable requirement description about the activities and objects of a model. A rule definition based on *temporal logic* (an extension of Boolean logic with temporal operators) specifies *relative* statements about the temporal order of the elements [6]. One variant of the temporal logic is the *branching-time logic* (Computation Tree Logic; CTL), which is suitable for the verification of business processes. CTL presumes that every point in time can be branched into multiple potential futures each specifying a different state. Theoretically, each of these states is in a different path of the resulting tree [10, 6, 9].

CTL is a textual formalization that contains operators and path quantifiers. First it defines the X- (next), F- (eventually) and G-operators (globally). They indicate that a statement should be valid in the next state/ at least in one state in the future/ in every state in the future. Besides there exists the binary U-operator (until) to define that a property has to be valid until a condition is met. Additionally, the A-quantifier (all paths) means that the property must hold for every path in the tree. The E-quantifier (at least one/ exists) means that the property must hold for at least one path in the tree [7, 9, 11].

The textual form of CTL is an obstacle of its general use in industry. Domain experts, i.e. people in operating departments, are in contrast to modeling experts less trained and willing to neither reading nor writing textual rules. But they have a clear understanding of the requirements a business process must fulfill. A graphical notation brings together the domain and modeling experts.

[9] presented a visual notation called G-CTL. It defines graphical representations of the CTL-elements. Figure 3 shows on the left the temporal operators for the E-quantifier (exists) and the operators for the A-quantifier (all paths) in the middle. The Boolean operators and constants (i.e. true, false) can be used in conjunction with these temporal operators. The operands "a" and "b" are placeholders for concrete (process) elements. This enables to specify a process in the well-known visual manner and the rules in the very same way.

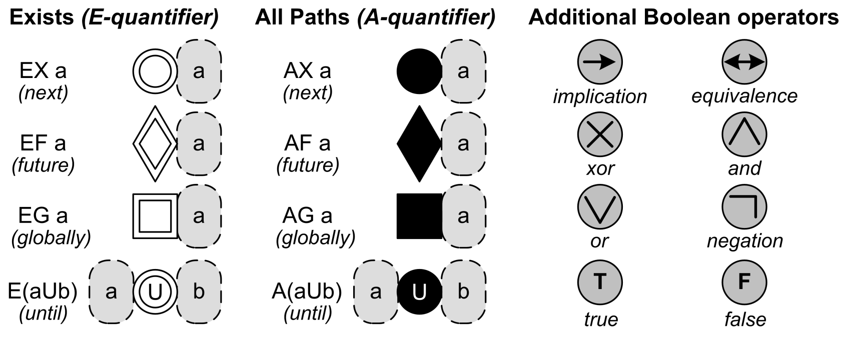


Fig. 3. Overview of the G-CTL operators [9]

4 Rule Patterns Set for Operational Conformity Checking

This section presents a set of rule patterns for operational conformity checking. The set is not intended to be a fully comprehensive definition of all possible types of rules. It is rather a collection of frequently occurring structures. The patterns result from the experience of process management projects (i.e. modeling and optimizing processes) mainly at small and medium insurance companies.

All these projects required an extensive manual verification of the designed processes. Applying our automated verification approach may considerably reduce this effort. Therefore, we aim at formalizing the implicit knowledge involved in the manual actions to automate the verification. Due to the operational cluster abstraction (cf. section 2.1) and scope (cf. section 2.2) concepts the majority of rules are reusable. However, it is still possible to use concrete activities within rules if necessary. The collection is not restricted to insurance companies but can be used for business processes in general.

Each rule pattern will be described in natural language and graphically formalized in the G-CTL notation (extended by the concept of operational clusters).

The first part of the rules principally targets the structure of a process model. In figure 4 the formalization of each rule pattern in G-CTL notation is shown:

RP1: *Activity/ Cluster **A** occurs at least once at [all | one or more] paths*
An activity or a cluster has to exist at least once. It has to be defined if ***A*** has to be executed in every "computational tree path" of a model (all; A-quantifier) or if it just has to occur on at least one path (exists; E-quantifier).

RP2: *Activity/ Cluster **A** does not occur*
An activity or a cluster must not exist in the process model.

RP3: *Activity/ Cluster **A** is performed [before | after] activity/ cluster **B***
B must not be performed "before" ***A*** has been executed. Or it must apply that ***A*** is not executed until "after" ***B***. If ***A*** and ***B*** are activities then it is the trivial case. But if one is a cluster there are two cases to distinguish because a cluster can contain activities that are maybe not placed directly behind each other. However the cluster concept simplifies the rule definition:
first occurrence: The first activity related to cluster ***A*** has to be performed before/ after the first activity related to cluster ***B***. A second activity of ***A*** might be existing after/ before the first activity of ***B***.
completion: All activities belonging to the cluster ***A*** have to be performed before/ after the first/ last activity belonging to the cluster ***B***.
In cases when ***A*** or ***B*** is a single activity, the explanation applies accordingly.

RP4: *Activity/ Cluster **A** is performed [first of all | last]*
An activity ***A*** or the first activity of cluster ***A*** respectively has to be the first activity right after the start event ("first of all"). Vice versa, an activity ***A*** or the last activity of cluster ***A*** respectively has to be the last activity right before the end event ("last").

RP5: *Cluster **C** [does | does not] contain the activity **A***
If the cluster ***C*** exists in the process model, then it has to contain the activity ***A*** ("need to") or the activity ***A*** has to be absent ("must not") respectively.

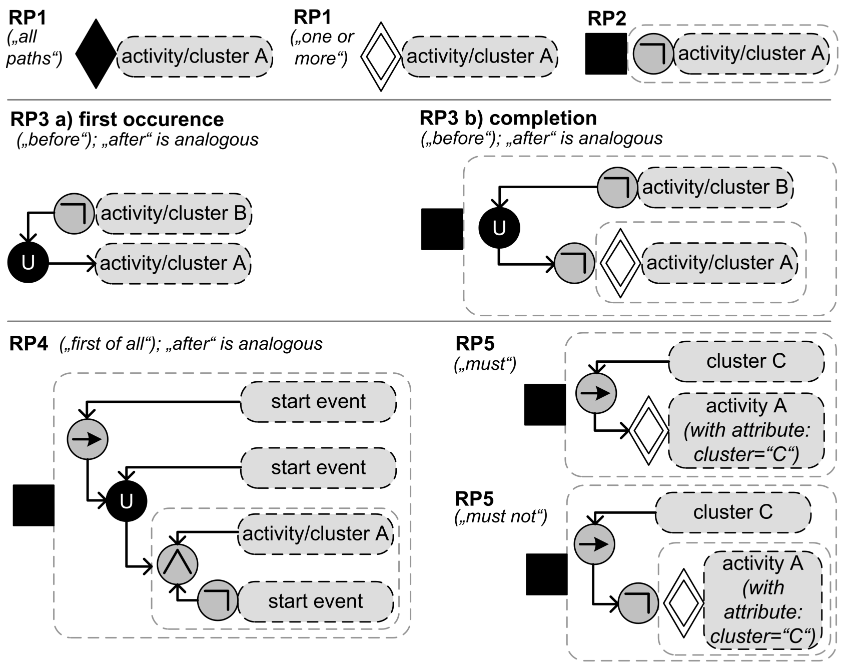


Fig. 4. Formalization of the rule patterns targeting the structure (G-CTL notation)

The second part targets the accomplishment of a process element from an operational point of view. Figure 5 shows the formalization using G-CTL:

RP6: *Activity/ Cluster **A** uses the system **S***
It has to be modeled that the activity/ cluster uses the defined system.

RP7: *Position **P** takes responsibility for the activity/ cluster **A***
This rule specifies that position ***P*** have to be responsible for performing **A**.

RP8: *Activity **A** is executed [manually | semi-automatically | script-based | service-based]*
This rule refers to the types of BPMN tasks. It has to be assumed that the activity is represented by an adequate task in the model.

RP9: *Activity/ Cluster **A** needs to [read | write] the business object **O***
An activity or a cluster has to use the business object (e.g. contract) as an input ("read") or it has to create/ modify the object ("write").

RP10: *Activity/ Cluster **A** needs to [read | write] the business object **O** before the object is [read | written] by activity/ cluster **B***
This rule specifies that a business object has to be first read/ written by the activity/ cluster ***A*** before the activity/ cluster ***B*** can use it either for reading or writing. This is to ensure the correct operating sequence, e.g. that the object needed by ***B*** has been created by ***A***.

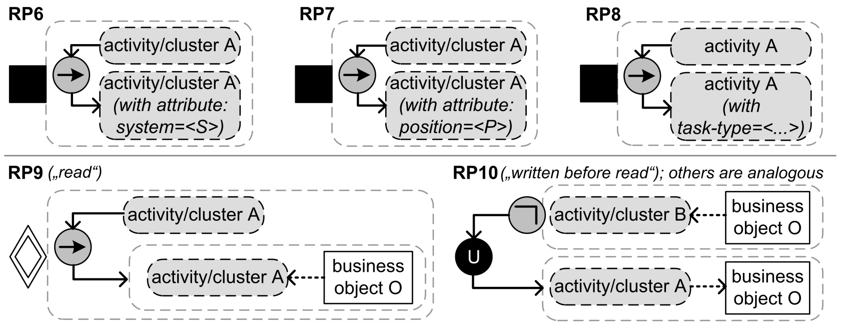


Fig. 5. Formalization of the rule patterns in G-CTL targeting the accomplishment

5 Integrated Platform for Modeling and Verification

The tool "Business Application Modeler" (BAM) is an integrated platform for business process modeling plus the specification of rules using G-CTL and verification of the models. It is based on Eclipse[2] and provides extension points to enhance the functionality by using plug-ins [12]. BAM supports EPCs besides the presented BPMN notation for process modeling and the G-CTL for rule specification. Elements of process models may be directly inserted within the rules. The new BPMN meta model specifies the elements and connections allowed in process models and rules. It uses slightly modified graphics compared to the default BPMN ones to improve readability as depicted in figure 9. To reduce complexity regarding the usability and understanding we decide to use only a subset of all BPMN 2.0 [13] elements. This subset contains tasks (manual, user, service, script, business rule, send, receive, abstract), subprocesses, gateways (xor, and), untyped events (start, intermediate, end), the data object and flows (sequence, data). Moreover, a special element called "operational cluster" (see figure 7 in the example) has been added to the rule editor. The corresponding operational cluster, based on the operational architecture, is stored as an attribute of this element (cf. section 2.1). Also the systems used by an activity or the position responsible are stored as an element's attribute. One feature supported by BAM is using a wildcard instead of assigning a concrete label to the element's name value [12]. This allows checking if the cluster (i.e. an arbitrarily named element with exactly defined attributes) exists within the verified process model. For performing the model checking, a plugin has been developed which uses the extension points offered by BAM. The verification is done automatically and only the process model and the rules have to be chosen manually. On the basis of our scope mechanism (cf. section 2.2) the plugin automatically uses only the rules that are relevant to the actual process model. First the business process model and the visually specified rules are transformed into a machine-readable textual form for the model checker. Then the model checker performs the verification. This plugin gets back the result ("true" or counterexample) and uses the given

[2] http://www.eclipse.org

handling of counterexamples that occur if the process model does not fulfill a certain rule (cf. section 3). BAM retranslates the counterexample and highlights the path which causes the failure [12]. This counterexample is displayed so that the domain expert can identify the affected part of the model.

6 Exemplary Illustration of the Approach

This section presents a typical example taken from a project to describe our approach to automate the verification of the operational conformity. The goal of the project was to assimilate and optimize the processes of different sections. The manual verification was very time-consuming because it has to be done carefully to ensure that all requirements are fulfilled. In the following we focus on the process type "Contract Creation" in the section "personal liability".

The personal liability insurance is very common in Germany. It protects the policyholder and co-insured persons from paying for any damages they cause with respect to the civil law. Therefore, the policyholder pays an insurance premium and the insurance company will pay in case of damage to compensate it [14].

Imagine that the process model is given and the activities are already annotated with their clusters. Figure 6 shows it in terms of the slightly modified BPMN notation. For convenience the operational clusters (cf. section 2.1) are highlighted in this paper. Then the process model has to be imported into BAM. In BAM the membership to a cluster is denoted as an attribute like mentioned before. Also the process type and the section may be specified in BAM for using the automatic rule selection based on the scope (cf. section 2.2).

The process starts with a subprocess for incoming communications. This handles all operations until passing the proposal to a responsible employee. Then a partner record will be created in the system if the inquirer has been no customer yet. Otherwise the existing data record is used. In both cases the proposal is used. After this the data for debt collection is created or the existing data is used. Thereafter the contract is created and the initial contract information is entered. The proposal is the input and a contract data object is the output. Optionally a discount can be granted. After that the provision of an involved insurance agent may be modified. If required it is possible to enter additional conditions and/ or contractual terms. Then the risk object to be insured and the desired scope of cover will be documented in the system. Finally the creation is completed and the insurance policy will be sent to the policyholder (subprocess).

In the following, two rules are illustrated which have to be checked in fact. They correspondents to the rule patterns in section 4.

The first rule is: "Subprocess *Incoming Communication* needs to write the business object *Proposal* before the object is read by the cluster *Partner Information*" (RP10). The cluster "Partner Information" can only perform its operations if the proposal has been created. The rule only applies to processes of type "Contract Creation". The sections are not limited. Figure 7 shows the rule in G-CTL. The model illustrated in figure 6 trivially fulfills this rule.

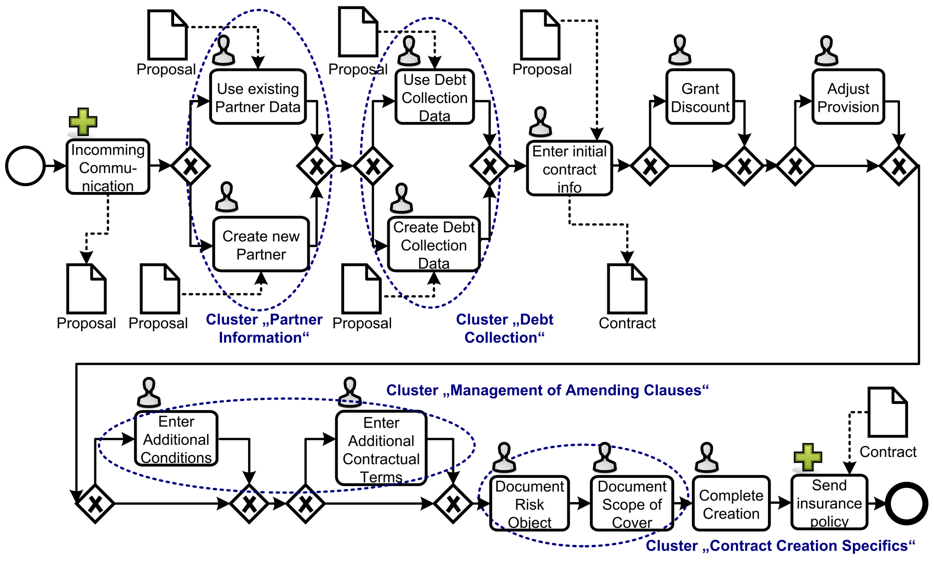

Incomming Commu-nication
Proposal
Use existing Partner Data
Create new Partner
Cluster „Partner Information“
Use Debt Collection Data
Create Debt Collection Data
Cluster „Debt Collection“
Enter initial contract info
Contract
Grant Discount
Adjust Provision
Cluster „Management of Amending Clauses“
Enter Additional Conditions
Enter Additional Contractual Terms
Document Risk Object
Document Scope of Cover
Complete Creation
Send insurance policy
Cluster „Contract Creation Specifics“

Fig. 6. Example process "Contract Creation" in the section "personal liability" with highlighted operational clusters

The second rule is: "Cluster *Contract Creation Specifics* is performed before the cluster *Management of Amending Clauses*" (RP3 - b (completion)). Assume that this rule only applies to processes of type "Contract Creation" and section "personal liability". Any other process of type "Contract Creation", e.g. section "household contents", does not have to fulfill this rule. Figure 8 shows the formalization of the rule. We choose the second interpretation of the rule pattern ("completion"). *All* activities of the first cluster thus have to be performed before the first activity of second cluster is executed. As to be seen in figure 6, the process model does not fulfill this requirement. Not all activities of the cluster "Contract Creation Specifics" are performed before the first activity of the cluster "Management of Amending Clauses".

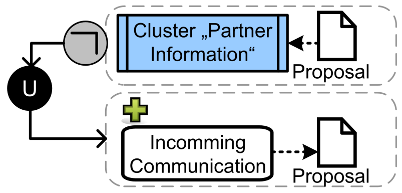


Fig. 7. First example of an operational rule (RP10)

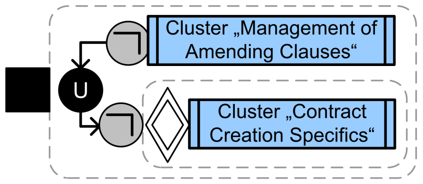
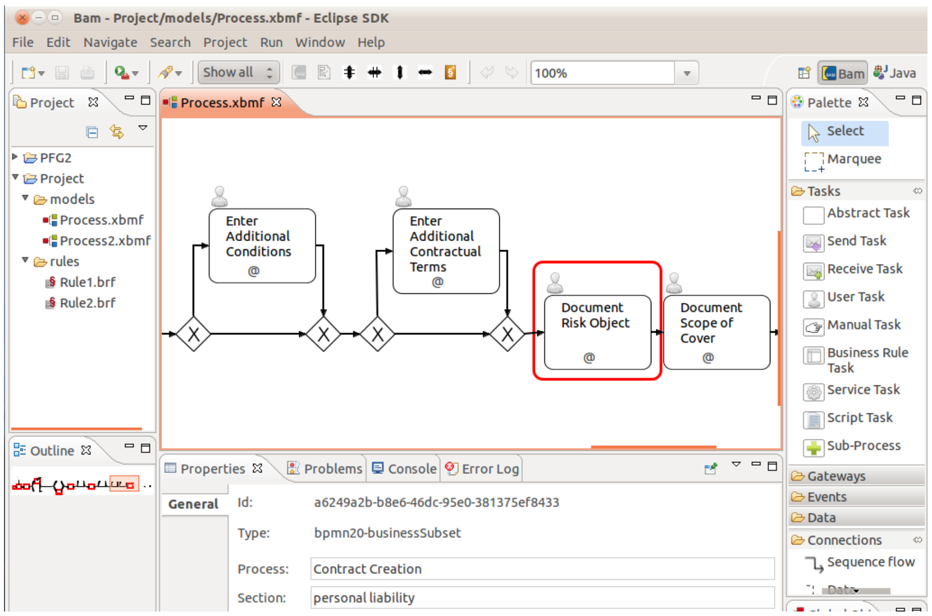

Cluster „Management of Amending Clauses“
U
Cluster „Contract Creation Specifics“

Fig. 8. Second example of an operational rule (RP3 - b (completion))

After importing the process model and specifying the rules the user can start the verification. BAM takes only the rules into account which in this case apply to all process types or to those of type “Contract Creation”. As a second criterion it will be checked if the section is not restricted or is valid for “personal liability”. After that, BAM transforms the model and the rules into an appropriate form for the model checker which performs the checking (cf. section 5).

Both rules are relevant for the verification. BAM will display the counterexample returned by the model checker according to the violated rule and highlights the affected tasks on the path leading to the violation. A detail of this is shown in figure 9. It has to be analyzed if the model is still valid and just the rule is incorrect or how the model can be modified to meet the requirement. In our case the model needs to be modified according to the correct order specified in the rule. It is a requirement to perform the tasks “Document Risk Object” and “Document Scope of Cover” of the cluster “Contract Creation Specifics” before any task of the cluster “Management of Amending Clauses”. Hence, BAM highlights the task “Document Risk Object” as illustrated because the tasks “Enter Additional Conditions” and “Enter Additional Contractual Terms” are each placed before the cluster “Contract Creation Specifics” represented by the task “Document Risk Object” which is the first violation of the rule.



Id: a6249a2b-b8e6-46dc-95e0-381375ef8433
Type: bpmn20-businessSubset
Process: Contract Creation
Section: personal liability

Fig. 9. Screenshot of BAM showing the counterexample of the violated second rule (RP3 - b (completion))

The advantage of the automatic checking is that a repeated verification (“regression testing”) is quickly done and ensures that all previous tested rules are checked again. A manual test would be time-consuming and probably miss a newly created error.

7 Related Work

The verification of the correctness or compliance of business processes and their models is a key problem of enterprise architecture management in general and business process management in particular.

A common technique to perform the verification is model checking. It has been proven to be applicable to software models (e.g. [15] or [16]) as well as to business processes (e.g. [17] or [18]). A common base for the checking is the semantic of the process metamodel [19].

There exist different approaches to check a business process model. One possibility is to use process reference models as a basis. For example, [20] takes a given reference model and tests if logs produced by systems conforms to this model. An algorithm to measure if a process model is compliant to a reference model is described in [21]. Therefore, both approaches need an assured reference model that has been modeled and agreed before.

Further approaches use rules to specify requirements about properties to be checked. The authors in [22] specify the control objectives in a normative way (i.e. the effect of an action on an object after execution). That is different to the use of temporal logic. The rules in [23] focus on elements of the lowest level of detail, i.e. an activity, without providing an abstraction mechanism. Another example is [24]. This approach limits the rule specification to predefined graphical patterns based on temporal logic. [25] also provides a visual notation for rule specifications but it differs from the process model notation. A similarity to our approach is the idea of abstraction levels in the rule definitions not assuming that all activities of similar models must have the same labels.

Rules may also be used to drive the transformation process from models to concrete systems and thus limiting the errors [26]. Here errors in the model, which are the focus of our work, may be not detected.

[27] also abstract from the names of the activities by using higher level concepts (i.e. superordinate terms) of an ontology (like [28]). However, the textual notation does not support unambiguous rule specifications because there is no unique assignment from activities to one superordinate term.

A generic approach is the workflow pattern approach presented in [29] (control-flow patterns) and [30] (data patterns). It demonstrates that patterns are a suitable mechanism to define the quality of processes and workflows. This work and [31] are a base for our rule patterns which are by far more domain specific while not being applicable to only one domain. Another approach is described in [32]. The authors use one visual notation (UML Activities) for business process modeling and rule specification. The proposed patterns to verify the process quality are rather technical. Similar to our approach the authors use model checking for verification.

These papers are a motivation for our work as they indicate the general applicability of patterns to define requirements. One key aspect for us is the integration of specifying process models and rules. Another key aspect is bridging the gap between domain experts and modeling experts based on using available parts of an enterprise architecture - the business capabilities.

8 Conclusion

An essential part of any enterprise architecture is a business architecture containing the business processes and the business capabilities the company aims for. These capabilities provide an operational structuring of the business processes and may be represented in a hierarchy called "operational architecture" like our real world example with the operational clusters on the third level.

Using operational clusters is a key aspect of the efficient and easy understandable rule definition from an operational point of view. The rules define statements about the quality and correctness or compliance of business processes. Ensuring high quality and correct processes is crucial for the success of an enterprise and therefore a main activity in any enterprise architecture management.

Our approach uses the operational clusters within rule specification to abstract from the activity level. This allows specifying more general rules, which can be reused for many processes. Hence, a rule repository can be created with relatively little effort. But it possible to use activities in combination with the operational clusters where considered necessary. Furthermore, we propose the use of a meaningful graphical notation for specifying rules like the G-CTL which is suitable for the verification of business processes with their dynamic structure. This can bring together domain and modeling experts because the domain experts can model the rules based on their understanding of the requirements. G-CTL is a visual notation for the CTL, a variant of the temporal logic.

In addition, we presented a set of rule patterns based on the experience of industrial projects. This set reflects the implicit knowledge involved in order to automate the verification. The first five rules patterns target the structure of a process model and the next five rules target the accomplishment.

The tool BAM integrates the business process modeling and rule specification. Moreover, it provides the automatic verification of the models that saves time and costs. All relevant rules can be checked iteratively while creating new versions of a business process ("regression testing"). This rule selection is done on the basis of our approach of scope definition. Each rule can be augmented with corresponding process types and optional section restrictions.

The application of the approach is illustrated by a real-life example that has been taken from a project. This demonstrates the effectiveness for verifying the operational conformity of business process models using the described approach based on elements of a business architecture.

Further work will focus on the application of the approach in other domains of the financial sector. It is planned to extend BAM by offering templates to simplify rule definitions while enabling the specification of arbitrary (complex) rules at the same time. Moreover, it has to be evaluated how the definition of the operational architecture, which has to be done outside of BAM yet, could be integrated in BAM. In addition to that it has to be analyzed how the user can be better supported in case of errors in the model.

References

1. Barkow, R.: Grundlagen von EAM. In Keuntje, J.H., Barkow, R., eds.: Enterprise Architecture Management in der Praxis: Wandel, Komplexität und IT-Kosten im Unternehmen beherrschen, Düsseldorf, Symposion Publishing GmbH (2011) 15–47
2. BITKOM: Enterprise Architecture Management - neue Disziplin für die ganzheitliche Unternehmensentwicklung. `http://www.bitkom.org/files/documents/EAM_Enterprise_Architecture_Management_-_BITKOM_Leitfaden.pdf` (2011)
3. Crown/Ministry of Defence (UK): MODAF Glossary v1.2. `http://www.mod.uk/NR/rdonlyres/D4CEF7F5-B008-4734-9D05-7DD746B0F166/0/20090304_MODAF01_2Glossary_V1_0__1.pdf` (July 2008)
4. AK-VAA: VAA Final Edition. Managementsummary. Version 2.1 prozedural, Version 2.0 objektorientiert. `http://www.gdv-online.de/vaa/vaafe_html/dokument/asummary.pdf` (2001)
5. Kleinert, H., van Megen, H., Kohl, T.: Integration von Prozess- und IT-Architekturmanagement. In Gensch, C., Moormann, J., Wehn, R., eds.: Prozessmanagement in der Assekuranz. Frankfurt-School-Verlag, Frankfurt am Main (2011) 221–234
6. Clarke, E.: The Birth of Model Checking. In Grumberg, O., Veith, H., eds.: 25 Years of Model Checking. Volume 5000 of LNCS. Springer, Heidelberg (2008) 1–26
7. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. **8** (April 1986) 244–263
8. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (2000)
9. Feja, S., Fötsch, D.: Model Checking with Graphical Validation Rules. 15th IEEE Intern. Conf. on the Engineering of Computer-Based Systems (ECBS 2008), Belfast (2008) 117–125
10. Clarke, E.M., Emerson, E.A.: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In: Logic of Programs, Workshop, London, Springer-Verlag (1982) 52–71
11. Pulvermüller, E., Feja, S., Speck, A.: Developer-friendly verification of process-based systems. Knowledge-Based Systems **23**(7) (2010) 667 – 676 Special issue on "Intelligent Formal Techniques for Software Design: IFTSD".
12. Feja, S., Witt, S., Speck, A.: BAM: A Requirements Validation and Verification Framework for Business Process Models. 11th Intern. Conf. On Quality Software (2011) 186–191
13. OMG: Business Process Model and Notation (BPMN) Version 2.0, formal/2011-01-03. `http://www.omg.org/spec/BPMN/2.0/PDF` (January 2011)
14. Wagner, F., ed.: Gabler Versicherungslexikon. Gabler, Wiesbaden (2011)
15. Emerson, E.A., Clarke, E.M.: Characterizing Correctness Properties of Parallel Programs Using Fixpoints. In: ICALP 1980, Automata, Languages and Programming, 7th Colloquium, Springer LNCS 85 (1980) 169–181
16. McMillan, K.L.: Symbolic Model Checking. Kluwer Academic Publishers (1993)
17. van der Aalst, W.M.P.: Formalization and Verification of Event-driven Process Chains. Information and Software Technology **41**(10) (1999) 639–650
18. Anderson, B.B., Hansen, J.V., Lowry, P.B., Summers, S.L.: Model checking for design and assurance of e-Business processes. Decision Support Systems **39**(3) (2005) 333–344

19. Decker, G., Mendling, J.: Instantiation Semantics for Process Models. In: Business Process Management, 6th International Conference, (BPM 2008). Volume 5240 of Lecture Notes in Computer Science., Springer (2008) 164–179
20. van der Aalst, W.: Process Discovery: Capturing the Invisible. Computational Intelligence Magazine, IEEE **5**(1) (feb. 2010) 28 –41
21. Gerke, K., Cardoso, J., Claus, A.: Measuring the Compliance of Processes with Reference Models. In: Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I. OTM '09, Berlin, Heidelberg, Springer-Verlag (2009) 76–93
22. Governatori, G., Hoffmann, J., Sadiq, S., Weber, I.: Detecting Regulatory Compliance for Business Process Models through Semantic Annotations. In Ardagna, D., Mecella, M., Yang, J., Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M.J., Szyperski, C., eds.: Business Process Management Workshops. Volume 17 of LNBIP. Springer, Heidelberg (2009) 5–17
23. Becker, J., Bergener, P., Delfmann, P., Eggert, M., Weiß, B.: Supporting Business Process Compliance in Financial Institutions - A Model-Driven Approach. In: 10th Intern. Conf. on Wirtschaftsinformatik (WI 2011), Zürich (2011) 355–364
24. Awad, A., Weidlich, M., Weske, M.: Visually specifying compliance rules and explaining their violations for business processes. JVLC **22**(1) (2011) 30 – 55 Special Issue on Visual Languages and Logic.
25. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Proc 22nd intern. conf. on Advanced information systems engineering. CAiSE'10, Heidelberg, Springer (2010) 9–23
26. Vasilecas, O., Smaizys, A.: Business Rule Model Integration into the Model of Transformation Driven Software Development. In: Advances in Databases and Information Systems, 13th East European Conference, ADBIS 2009. Volume 5968 of Lecture Notes in Computer Science., Springer (2009) 153–160
27. Medeiros, A.D., Karla, A., Aalst, V.D.: Semantic Process Mining Tools: Core Building Blocks. In: In 16th European Conference on Information Systems. (2008)
28. Opdahl, A.L., Berio, G., Harzallah, M., Matulevicius, R.: An ontology for enterprise and information systems modelling. Applied Ontology **7**(1) (2012) 49–92
29. Russell, N., ter Hofstede, A.H., van der Aalst, W.M., Mulyar, N.: Workflow Control-Flow Patterns: A Revised View; BPM Center Report BPM-06-22, BPMcenter.org. `http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf` (2006)
30. Russell, N., ter Hofstede, A.H., Edmond, D., van der Aalst, W.M.: Workflow Data Patterns; QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane. `http://www.workflowpatterns.com/documentation/documents/data_patterns%20BETA%20TR.pdf` (2004)
31. Sandkuhl, K.: Validation and Use of Information Demand Patterns in Higher Education. In: Business Information Systems Workshops - BIS 2010 International Workshops. Volume 57 of Lecture Notes in Business Information Processing., Springer (2010) 204–213
32. Förster, A., Engels, G., Schattkowsky, T., Van Der Straeten, R.: Verification of Business Process Quality Constraints Based on Visual Process Patterns. In: Proceedings of the First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering. TASE '07, Washington, DC, USA, IEEE Computer Society (2007) 197–208