An Approach to Combine Data-Related and Control-Flow-Related Compliance Rules

Daniel Schleicher*, Stefan Grohe, Frank Leymann*, Patrick Schneider§, David Schumm*, Tamara Wolf

*Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
lastname@iaas.uni-stuttgart.de

§Fraunhofer Institute for Industrial Engineering IAO, Germany
Patrick.Schneider@iao.fraunhofer.de

***Abstract*—Compliance of IT-enabled business processes is a research area gaining more and more attraction for enterprises today. Many enterprises are on the gap of installing workflow systems within their premises. During this process they need to make sure that several regulations, coming from governments or enterprise-internal institutions, are obeyed. We argue that the compliance regulations, enterprises are faced with today, can be built using a number of atomic compliance rules. Until now only control-flow-related atomic compliance rules have been identified in literature. In this paper we extend this list with several data-related atomic compliance rules. We further show how control-flow-related compliance rules and data-related compliance rules can be combined. A fundamental finding that we made in our work with industrial use case partners from EU projects, as well as projects with customers, is that for the specification of control-flow-related compliance rules data issues must also be considered.**

The main contribution of this paper is a collection of combined compliance rules implementing complex compliance requirements which consist of atomic control-flow related and data-related compliance rules.

***Keywords*-Compliance, business process, constraint, pattern**

I. Introduction

The divide and conquer approach is one of the main techniques for humans to create solutions for problems that are too big to be solved as a whole. This is also true for the act of designing business processes. Human business process designers should be able to fully concentrate on the main problem which they want to solve with a new business process.

Today compliance of business processes is becoming more and more important for enterprises. Business processes are subject to be compliant with requirements coming from various sources, like laws or enterprise-internal regulations. Thus, it is crucial to separate the act of designing a business process and implementing a business goal from the act of designing a *compliant* business process. Tools should support human business process designers when they are making a modification to a business process. When this modification would violate a certain compliance rule the tool should notify the human business process developer.

In the following we use the term *compliance rule* to describe formally defined compliance requirements. Thus, the term *compliance requirement* is used to describe abstract requirements coming from laws or enterprise-internal regulations.

In this paper we focus on the two main kinds of compliance requirements for business processes, data-related requirements and control-flow requirements. There are also some compliance rules which need the concept of a role to be defined properly. The role that is able to execute a certain activity can be specified within a business process model. It thus is part of a data-related requirement. However, there are few works [1], [2] in literature which present a collection of different atomic compliance rules which are of interest in business process design. Thus, the paper has two main contributions.

The first contribution is the extension of the collection of compliance requirements proposed by Turetken et al. [1] which are related to business process management. In our work in the MASTER EU project[1] we encountered several compliance rules extending the list presented in [1]. During our investigations we experienced that data-related compliance requirements must be considered in order to create completely specified compliance rules. Therefore, we added the new dimension of data-related compliance rules to this list.

The second contribution is the combination of atomic control-flow related compliance rules and data-flow related compliance rules. Here, we extend patterns proposed by Turetken et al. and add the new dimension of data-related compliance rules. These patterns can be used to incorpo-

[1]http://www.master-fp7.eu

rate complex compliance rules in business processes. We introduce the concept of a *rule tree* which supports the combination of different kinds of atomic compliance rules. These atomic compliance rules can be specified using arbitrary languages. In order to verify such combined compliance rules we developed an algorithm. The prototype uses plugins to verify different parts of a combined compliance rule. Each plugin can handle a certain formal language that may be used to specify a combined compliance rule.

Our findings are mainly based on two EU funded IT projects dealing with compliance of business processes, COMPAS[2] and MASTER. In COMPAS we designed an architectural framework to support business process compliance. In MASTER we designed a solution to monitor, audit, and enforce the compliant execution of business processes. In MASTER two real-world case studies have been used during the project to evaluate the results of the research. The first case study is placed in the health care sector. Here, an Italian hospital provided information on how a drug dispensation should be performed. There are strict compliance requirements regarding the administration of a drug or a drug trial study [3]. The second use case partner was a Spanish insurance company. Here, a process is described that starts when an agency of that insurance company requests information regarding a client of that agency [4], [5]. We identified compliance requirements in interviews with the partners and analysed the business processes of the use case partners. The findings have been published in [6].

In our investigations we encountered many kinds of compliance requirements like non-functional compliance requirements or security related compliance requirements. However, in this paper we focus on the most central compliance requirements that mainly deal with the structure of the business process model. Due to our research direction, the collection of atomic compliance rules and combined compliance rules we identified is limited by the kinds of business processes we worked with. It is also limited by the kinds of problems we encountered during our research and our work with industry partners. In this paper we focus on design time verification of control-flow and data-flow related compliance problems in the business process domain. Thus, we do not cover the following compliance problems. We do not answer the questions how secure or robust, for example, an implementation of a business process may be. We do not deal with quality of services problems, like response times of a business process. We do not consider the resources a business process may use. Resources can for example be humans or machines taking part in a business process.

With these limitations, we observed that there are two kinds of compliance rules for business processes: control-flow compliance rules and data-flow compliance rules.

In this paper we use the business process modelling and notation language (BPMN) in the version 2.0 [7] to illustrate the used processes. BPMN is a well tooled and widely adopted notation in the field of business process management. In BPMN data-objects are used to store the data used within a business process. Data associations connect these data-objects to the inputs and outputs of the activities which use this data.

[2]http://www.compas-ict.eu

The paper is structured as follows. Section II shows a running example which is used throughout the paper to illustrate the different aspects of the approach. Section III provides an introduction to the compliance patterns approach presented by Turetken et al. It also shows related work. Control-flow related atomic compliance rules are presented in Section IV. Data-flow related compliance rules are presented in Section V. Section VI shows an example of a combination of data-flow compliance rules and control-flow compliance rules. A prototype implementing the concepts to combine data-flow and control-flow compliance rules is presented in Section VII. The paper concludes with Section VIII.

II. Running Example

Figure 1 shows a business process written in BPMN. This business process implements the steps that have to be taken to process a credit request in a bank. After the credit request has been received by the bank a manager has to approve it when the requested amount is above 10.000$. In either of these cases an account manager has to approve the credit request afterwards. The last two steps of the credit approval business process are executed in parallel. The message board of directors is informed about the new credit and a message is sent to the customer detailing if the credit request has been approved or not. For reasons of simplicity we only show two data-objects in Figure 1. The business process model shows two compliance domains (dashed ellipses). A compliance domain is a concept we developed to annotate business process models with compliance rules. Details about this concept are presented next.

Compliance domains [8] are used to annotate business processes with data-related compliance rules. They also represent physical runtime infrastructures where different parts of a business process are executed on. A business process can be split apart using compliance domains and run on different physical infrastructures like cloud environments or a data-centre of an enterprise [9]. All activities contained within a compliance domain can only use services which run within the physical runtime infrastructure of the compliance domain where the invoking activity resides.

The main purpose of compliance domains is to restrict the data-flow within a business process model. With this means, it is possible to forbid the use of certain data for certain activities within a business process model, for instance.

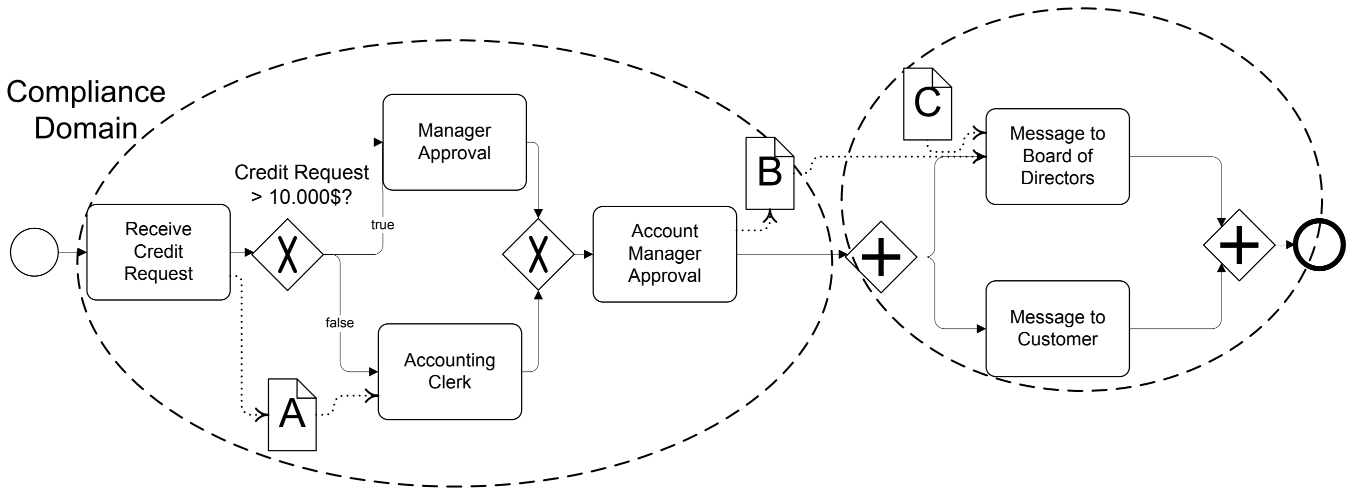


Figure 1. Example BPMN process

III. Research Foundations

In this section we introduce the notion of atomic compliance rules and present a number of them mentioned in literature. We further show related work in the field of compliance rules for business processes with focus on data-flow.

The work presented in [10] shows several data-related compliance problems for business processes. We used this paper as a source and picked the data-related compliance problems which can be automatically verified.

There are few works dealing with data-related compliance processes in the area of business process design. In [11] the authors extend BPMN-Q, a query language that can be used to query for business process models which fulfil a certain set of compliance rules. The extension to this language makes it data-aware. With this extension it is now possible to incorporate variable states in the queries. The argumentation in the paper is that, depending on the state of a variable, the process execution may continue in another direction. Thus, the extension presented in this paper deals with the control-flow of a business process in the first place. The data-related compliance rules we present in this paper can be seen as a separate kind of compliance rule. They are not influencing the control-flow of a business process in the first place.

Publication [12] deals with control objectives to be applied to business processes. Here, also data-related control objectives are mentioned. However, no formal representation of such a data-related control objective is given.

In [13] Knuplesch et al. show how data-aware compliance rules can be modelled using a simple notation. They also integrate their approach on defining data-aware compliance rules with LTL formulas. We basically use a similar approach to define data-aware compliance rules. In Section VI we show how they can be integrated in an LTL formula where combinations of atomic control-flow-related compliance rules and atomic data-related compliance rules are used. However, the focus of the approach presented in [13] lies on the automatic verification of compliance rules. The paper presents an approach on how states of variables of a business process model can be used in the automatic verification of the business process model. The states of variables are directly related to the control-flow of a business process because at a control-flow fork, variables are used to decide in which direction the business process execution would proceed.

In this paper we go further and present a number of data-related compliance rules which do not impact the control-flow of a business process in the first place.

The interrelation of reusable process artifacts implementing compliance requirements on one side, and the compliance patterns approach by Turetken et al. has been explored in [14].

Related work in the direction of rule-patterns to be used to develop compliant business processes has been done by Turetken et al. [1]. Here, the authors describe how control-flow related compliance rules can be expressed as patterns to be used by non-technical business process developers. Further, these patterns are used in a generic compliance conceptual model, also presented in the paper. We add the missing dimension of data-related compliance rules to this approach.

Besides [1] there is another paper presenting a taxonomy of compliance rules in the area of business process design [2]. In this work, atomic control-flow related compliance rules are presented. We use these two works as the foundation.

IV. Atomic Control-flow Compliance Rules Derived from Literature

In the following we show a number of atomic control-flow-related compliance rules that we found in literature. We use them in the remainder of this paper. We use logical expressions written in Linear Temporal Logic (LTL) in the head lines of the properties to indicate how they can be expressed using a formal language. In short, LTL is used to define formulae on future paths. It extends first order logic by introducing temporal operators. There is a number of temporal operators defined in LTL. In this paper we only use the *G* operator. G means always, globally. If the G operator is placed in front of a formula A this means A should always be true. The variables in these short logical expressions are process constructs like activities or process fragments. In Table I we show for every atomic compliance rule if it already has been mentioned in literature. We mention this also in the description of every atomic compliance rule.

The atomic compliance rules we present in the following are formulated in a positive way. The negation of every following compliance rule is also an atomic compliance rule.

A. Presence of Activities: $G(A)$

The term *presence of activities* denotes that a compliance rule is valid when a certain set of activities is present. In other words, activities implementing certain functionality, are used in a process model. This atomic compliance rule has been discussed in [1].

Example Use Case: In the credit approval process of our example shown in Section II there should be an activity implementing the credit approval by an account manager.

B. Execution Order of Activities: $G(A \Rightarrow B)$

With this compliance rule it is ensured that a predefined set of activities is executed in a predefined order. In a separation of duties scenario there are normally two check activities which need to be executed in a predefined order to verify if for example a document is valid.

Example Use Case: In our running example (see Section II), the activity *account manager approval* should always be executed before the activity *message to customer* is executed.

Next is one atomic control-flow compliance rule we did not find in literature, yet.

C. Parallel Activities: $G(A \mid\mid B)$

Figure 1 also shows an example for parallel activities. The activities *message to board of directors* and *message to customer* are running in parallel. This is a new atomic compliance rule, we did not find in literature, yet.

Example Use Case: In the running example (see Section II), the activities *message to board of directors* and *message to customer* should be executed in parallel to save the company time and money. Table I shows a list of all control-flow related compliance rules mentioned before.

Table I
LISTING OF CONTROL-FLOW RELATED COMPLIANCE RULES

Pattern name	**New**	**Formal Term**
Presence of activities		$G(a)$
Execution order of activities		$A \Rightarrow B$
Parallel activities	X	$A \mid\mid B$

V. ATOMIC DATA-FLOW RELATED COMPLIANCE RULES

In this section we present a new kind of atomic compliance requirement, namely data requirements. We abbreviate activities in the form of A1 … An and data-objects in the form of DO1 … DOn. We also do not use LTL to define the atomic data-related compliance rules in this section, because with LTL it is possible to specify logical terms on the future of an execution path in a business process. This property cannot be used with data-related compliance rules. Instead, we use a syntax that is related to Java. We use a dot notation to access the contents of data-objects and operators like $==$ to test for equality, for example. The atomic compliance rules we present in the following are formulated in a positive way. The negation of every following compliance rule is also an atomic compliance rule.

A. Type of Data-Object: $DO.type == T$

With this atomic compliance rule a data-object can be constrained to objects of a certain data-type.

Example Use Case: The type of an input data-object for an activity that expects an integer as input should be integer.

B. Range of Data-Object: $DO.range == R$

This compliance requirement describes a constraint on the data that may be present during the execution of a business process. It can be applied to a data-object in a business process to constrain its data type, for example.

Example Use Case: The data-object shown in Figure 1 may be restricted to only contain data of type *credit request*, which may be an XML complex type.

This kind of compliance rule can also be used to describe forbidden states that a business process must never reach at run time. One example to illustrate this is the exclusion of semantic failures at run time of a business process. The transaction amount of a credit request must never be negative.

C. Data Location: $DO.location == L$

There are cases when the location of data within a business process is important. To explain that we will define what location of data means within the context of a business process. Afterwards we show an example detailing where it is important to know the location of data within a business process.

The *location of data* is bound to the location of the activities which use this data for execution. As we saw before compliance domains can be used to split a business process model. Every compliance domain then can be executed on a different physical run-time environment like a private data-centre or a public cloud. Thus, the location of a data-object is determined by the location of the activity which uses the data from this data-object. The location of an activity is determined by the compliance domain it resides in. The output of an activity does also matter because the data has to pass the activity in order to get to the output data-object. Another aspect of data location is visibility. In BPMN, one can define areas of visibility. Sub processes are such areas for example. BPMN constructs which reside within such an area of visibility are only visible from other constructs residing in the same area of visibility. To be more concrete, data-objects which reside within a sub process are not visible for BPMN constructs residing outside of the sub process. Thus, such data-object cannot be used by activities residing outside of a sub process.

Example Use Case: The physical location *L* of the data-object shown in Figure 1 may be restricted by the surrounding compliance domain. This compliance domain may represent a data-centre of an enterprise.

To show the importance of this data location compliance requirement we assume a business process which is designed as a whole by a human business process designer and then split and executed in different physical locations. These physical locations can be cloud environments or data centres. It is crucial for enterprises today to know on which physical locations their data is stored because of different sometimes colliding laws.

It may be that a European enterprise does not want that any of its data is physically stored in the USA, because of the different legal situation in the USA and Europe concerning data management. A good real-world example could be the conflicting "Patriot Act" and the German Data Security and Privacy legislation (Bundesdatenschutzgesetz) [15].

D. Restricted Data Input of Activity: $A1.input == DO1.customerId$

Another compliance requirement which is related to the requirement described in Section V-C is *restricted data input*. In contrast to the data location compliance requirement the focus of *restricted data input* lies on the activities which use data in order to execute their tasks.

In some occasions it is important that certain data may not be passed to a certain activity. Or it is important that certain parts of a data-object may or may not be passed to an activity. The formal term for this atomic compliance rule states for example that the input data for activity A1 must come from data-object DO1. Additional to that it states only the customerId stored in data-object DO1 can be used as input for A1.

Example Use Case: There could be a data-related compliance rule stating that the activity *accounting clerk* should only use the data-object A shown in Figure 1 as an input.

E. Restricted Data Output of Activity: $A1.output == DO1$

Analogue to the atomic compliance rule shown in Section V-D the output of an activity can be restricted.

Table II
LISTING OF DATA-FLOW RELATED COMPLIANCE RULES

Compliance Rule	**Description**
$DO.type == T$	Type of data-object
$DO.range == R$	Range of data-object
$DO.loation == L$	Data location
$A.datainput == D$	Restricted data input of activity
$DO.writes <= 1$	Avoid Race Conditions
$A1.input == A2.input$	Use of same data
$A.assignee == P1$	Assignment of a Person to an Activity

F. Avoid Race Conditions: $DO.writes <= 1$

For this data-related compliance requirement we define that a data-object DO knows how many writing data-associations are writing to it. This number of writing data-associations needs to be queried using the dot-notation like in the example in the headline of this atomic compliance rule.

A concurrent write of two activities on the same data-object may lead to unpredictable behaviour of business process models. In the following we explain that with an example. Assuming activities A1 and A2 are writing their results to the same data-object DO. We further assume, A1 writes the result before A2 to DO. Now we assume that another activity A3 is assuming it reads the result of A1 from DO, this may well not be the case anymore because A2 may have overwritten the results of A1.

Example Use Case: Examples are process models where two or more activities update one data-object concurrently.

G. Use of Same Data: $A1.input == A2.input$

This compliance rule constrains two or more activities in a business process to have exact the same input data.

Example Use Case: All activities in our running example business process presented in Figure 1 need to work with exactly the same credit request document.

H. Assignment of a Person to an Activity: $A.assignee == P1$

This compliance rule is also a data-related compliance rule because the assigned person for activity A is stored as meta-data of this activity. In the logical expression for this atomic data-related compliance rule P1 denotes a specific person. The assignment of persons to activities in a business process has been discussed in literature before. We list it here as an atomic compliance requirement.

Example Use Case: In a business process running in an hospital it is imaginable that task A and task B must be carried out by the same doctor. Thus, the same person must be assigned to tasks A and B.

Table II shows the atomic data-related compliance rules which have not yet been discussed in literature before. As stated above we discovered these compliance rules with our partners until now. There may be further compliance rules to be discovered in the future.

VI. Combination of Data-Related and Control-Flow-Related Atomic Compliance Rules

In this section we describe a complex compliance rule that we encountered during our research. For this we use the atomic compliance rules presented in Section IV to V. Control-flow related compliance rules imposed on IT-driven business processes depend in many cases on data-related compliance requirements and vice versa. This means, for example, that certain data must be present at a specific execution point in the business process so that the execution can be continued. In this section we detail the connection between the most common control-flow-related compliance rules and their implications on data-related compliance rules. We also show data-related compliance rules and possibilities of combination with control-flow compliance rules. The aim of this collection of complex compliance rules is to help human business process developers to easily select compliance rules on a more abstract level. When such a compliance rule is applied to a business process model the constituent atomic compliance rules are applied to the process model. These atomic compliance rules together implement the complex compliance rule.

A. Rule Tree: A Means to Compose Atomic Compliance Rules

Rule trees can be seen as a simple graphical language. Each rule tree represents a certain complex compliance rule which is built using atomic ones. The purpose of rule trees is to present in a graphical way which atomic compliance rules have been combined in order to built a complex compliance rule. This enables people with less knowledge in the fields of formal languages and compliance to specify and understand complex compliance rules.

In our investigations we saw that compliance rules for different purposes, like the expression of data-requirements, or control-flow requirements, are often written in different formal languages. Therefore, the atomic compliance rules used in a rule tree can be written in arbitrary formal languages.

The concept of a *rule tree* is based on a diploma thesis [16]. It also uses the work of Turetken et al. [1]. Turetken et al. show that compliance rules written in linear temporal logic can be combined using named patterns which can be reused also by non technical people. We extend this approach in two ways. We define a graphical syntax to show how atomic compliance rules are put together to form a more complex compliance rule, and we removed the constraint that only linear temporal logic can be used to describe complex compliance rules.

Figure 2 shows a rule tree. Here several control-flow compliance rules (e.g. activityOrder) and one data-flow compliance rule (use of same data) are combined to a separation of duties compliance rule. Complex compliance rules, implemented by a rule tree, can be applied to a business process model in the following way. The atomic compliance rules contained in a rule tree carry placeholders for activity names. These placeholders must be replaced by actual names of constructs within the business process model. These constructs may be single activities or complex business process structures.

Rule trees are built using the following schema. All inner nodes of a rule tree are formal operators. Formal operators that are allowed are *AND*, *OR*, and *NOT*. Leaves of a rule tree can not be operators. Leaves must contain a formal definition of a compliance rule.

Algorithm 1 Verify rule tree

```
 1: function VERIFY(BinaryTree ruleTree)
 2:     boolean result = false;
 3:     if not ruleTree.isOperator() then
 4:         return ruleTree.getFormula().verify();
 5:     else
 6:         if ruleTree.isNot() then
 7:             return not verify(ruleTree.getChild());
 8:         end if
 9:         result = verify(ruleTree.getLeft());
10:         if ruleTree.isAnd() then
11:             return result and verify(ruleTree.getRight());
12:         end if
13:         if ruleTree.isOr() then
14:             return result or verify(ruleTree.getRight());
15:         end if
16:     end if
17: end function
```

Algorithm 1 shows how rule trees are verified. The underlying concept is a pre-order traversal of the tree. A tree is pre-order traversed when the root of a sub-tree is processed before the left branch and the right branch of the sub-tree. For reasons and space and simplicity we left out any null-pointer checks in algorithm 1. The *verify*-function in algorithm 1 starts by processing the root node of the current sub-tree which is a parameter for that function. In line three it first determines if the root node of the current sub-tree is an operator. If it is no operator the algorithm must have reached a leaf of the overall rule tree. The formal rule (formula) that is attached to that leaf can be verified (see line 4). If the current root of the sub-tree is an operator the algorithm determines which one of the three possible operators is represented by the current root node.

If the current operator is a *not*-operator (see line 6) the only child node is verified by invoking the verify function with the child node of that not-operator. Since the not-operator is an unary operator, this node can only have one child. If the current operator is an *or*-operator or it is an *and*-operator the algorithm is processing these binary operators in the following way. At first the left side of the current sub-tree is verified by invoking the verify function with the left child

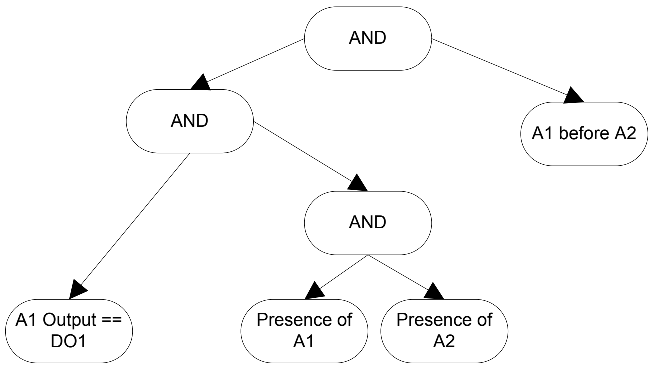


Figure 2. Combination of atomic compliance rules to implement a restricted data movement compliance rule

of the current root node. Afterwards this result is combined with the verification result of the right side of the sub-tree using either a logical and operation or a logical or operation.

B. Example: Restricted Data Movement

We came across this compliance rule on one of our previous works [8]. In this paper certain areas of a business process model are marked using compliance domains. Data that is passing the borders of these areas need to be compliant with a certain set of compliance rules. One example for such a compliance rule would be that if personal data is transferred it must be anonymised.

Figure 1 shows two compliance domains (dashed line). These compliance domains can contain an arbitrary number of activities. Compliance domains are useful in enterprise business process environments. For a European enterprise it may not be advisable to store or process operation-critical data in the US. This leads to internal regulations stating that no operation critical data must leave the premises of a certain data centre.

The compliance rules being annotated to compliance domains restrict the data set that can be processed or stored within a certain compliance domain. Thus, in the modelling phase of a business process tools can detect if certain data is sent to a compliance domain that would violate one or more data-related compliance rules.

Equation 1 shows the atomic compliance rules that have to be combined in order to produce a compliance rule for the compliance requirement of restricted data movement. In this combined compliance rule the $\wedge$ operator is used to combine eight atomic compliance rules. In this example for restricted data movement we want to restrict the data movement between the two compliance domains shown in Figure 1. We take a closer look at activities *account manager approval* and *Message to board of directors* for this example. We assume the right compliance domain can only handle non-sensitive data. This means the activities contained within the right compliance domain may be executed in a public cloud environment. Thus, the restriction on the data movement here is that only the identifier of the person requesting a credit may be forwarded from the left compliance domain to the right one. This concretely means that the input for activity *message to board of customers* consists of two parts. The identifier of the person coming from data-object B and some other details coming from data-object C. In the following we describe these atomic compliance rules in detail.

In the first part of Equation 1 it is assured that activities A1 and A2 are present and that activity A1 is always executed before activity A2. Then, the locations of the used data-objects (DO) in the business process model are set to compliance domains CD1 and CD2. Afterwards it is set that the output data of activity A1 is stored in DO1 and that activity A2 does not get its input data from DO1. This is the main part of the equation stating that data should not be moved directly between A1 and A2. In this part we make sure that the output data of activity A1 cannot be the input data for activity A2.

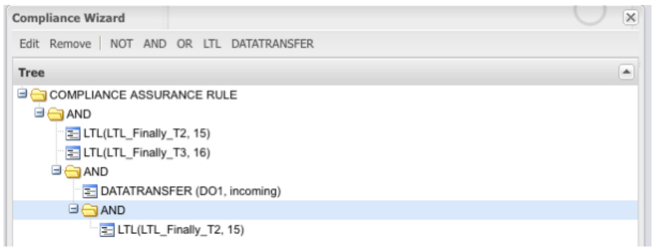
$$\begin{aligned} & G(A1) \wedge G(A2) \wedge G(A1 \rightarrow A2) \\ & \wedge DO1.location == CD1 \wedge DO2.location == CD2 \\ & \wedge A1.output == DO1 \wedge A2.input == DO1.requestId \\ & \wedge A2.input == DO2 \quad (1) \end{aligned}$$

For space reasons, the rule tree shown in Figure 2 shows a fraction of the atomic compliance rules presented in Equation 1. To validate this rule tree Algorithm 1 is used. It traverses the tree in pre-order until the algorithm reaches a leaf. In the case of the rule tree shown in Figure 2 it is the left most leaf of the rule tree stating that activity A1's output is written to data-object DO1. In order to verify this atomic data-related compliance rule a plugin is chosen that can verify data-related compliance rules. After the first leaf is validated the algorithm moves on the middle and-operator. It sees that this tree node is an operator and moves on to the leaves of this operator. It verifies them using a plugin that can verify atomic control-flow-related compliance rules. Afterwards the left most and-operator can be verified by the algorithm. The algorithm moves to the top and-operator and continues with the right most leaf. The algorithm provides the verification result when all operators in the rule tree have been processed.

VII. Prototype

Using the web-based BPMN editor *Oryx*, we have added support for control-flow and data-flow related compliance rules in previous works.

For this paper we implemented the *rule tree* concept and added support to graphically design compliance rules in linear temporal logic (LTL). Figure 3 shows the *compliance wizard dialogue* containing a graphical representation of a rule tree. In this rule tree several LTL compliance rules and one data-aware (data-transfer) compliance rule are combined



COMPLIANCE ASSURANCE RULE
AND
LTL(LTL_Finally_T2, 15)
LTL(LTL_Finally_T3, 16)
AND
DATATRANSFER (DO1, incoming)
AND
LTL(LTL_Finally_T2, 15)

Figure 3. Prototype: Compliance wizard containing rule tree

using *and* operators. The rule tree can be graphically built using control-flow related or data-flow related compliance rules that are stored in a compliance rule repository. After the rule tree for a specific BPMN business process model has been built, it is automatically stored in the repository.

Compliance rules can be written using a graphical syntax so that together with the rule tree concept, it is easier for non-technical people to define control-flow related compliance rules. The rule tree can be applied to an existing business process model by attaching the rule to a compliance domain, for example. Currently the activity names used in the attached compliance rule must match the activity names of the activities residing in a compliance domain. In the future we want to provide a wizard where it is possible to map activity names in compliance rules to process constructs in a step-by-step manner.

VIII. Conclusion

In this paper we added a new kind of compliance rules to the list of atomic control-flow related compliance rules described in literature. We further showed that this new data-related kind of compliance rules must be combined with existing control-flow-related compliance rules in order to fully specify complex compliance rules implementing a separation of duties compliance requirement, for example. To support the understanding of combined atomic compliance rules we introduced the concept of a rule tree.

In the future we want to prepare a concept and prototype so that a complex compliance rule can be selected and applied using a wizard to an existing business process model. During this application process the variables in the compliance rule to be applied must be mapped to constructs in the current business process model. We further want to deal with conflicting compliance rules. Here we want to extend the existing approach presented in [17] to handle combined compliance rules.

References

[1] O. Turetken, A. Elgammal, W.-J. van den Heuvel, and M. Papazoglou, "Enforcing Compliance on Business Processes Through the use of Patterns," in *European Conference on Information Systems (ECIS 2011)*. Elsevier, 2011.

[2] A. Elgammal, O. Turetken, W. v. d. Heuvel, and M. Papazoglou, "Root-cause analysis of design-time compliance violations on the basis of property patterns," Tilburg University, Tech. Rep., 2010.

[3] EMEA, *Guideline for Good Clinical Practice*, European Medicines Agency, July 2002.

[4] "MASTER Scenarios," MASTER EU Project, Deliverable D1.2.1, 2009.

[5] "Pilot Case Studies Instantiation," MASTER EU Project, Deliverable D1.2.3, 2011.

[6] "Stakeholder Requirements Analysis," MASTER EU Project, Deliverable D1.1.1, 2009.

[7] Object Management Group (OMG), "Business Process Model and Notation (BPMN) Version 2.0," Tech. Rep., 2009.

[8] D. Schleicher, C. Fehling, S. Grohe, F. Leymann, A. Nowak, P. Schneider, and D. Schumm, "Compliance Domains: A Means to Model Data-Restrictions in Cloud Environments," in *Enterprise Distributed Object Computing Conference*, 2011.

[9] R. Khalaf, "Supporting business process fragmentation while maintaining operational semantics : a BPEL perspective," Doctoral Thesis, University of Stuttgart, Germany, 2008.

[10] C. Cabanillas, M. Resinas, and A. Ruiz-Cortes, *On the identification of data-related compliance problems in business processes*, 2010, pp. 89–102.

[11] A. Awad, M. Weidlich, and M. Weske, "Specification, verification and explanation of violation for data aware compliance rules," in *International Joint Conference on Service-Oriented Computing*, 2009.

[12] S. W. Sadiq, G. Governatori, and K. Namiri, "Modeling control objectives for business process compliance," in *BPM*, 2007.

[13] D. Knuplesch, L. T. Ly, S. Rinderle-Ma, H. Pfeifer, and P. Dadam, "On enabling data-aware compliance checking of business process models," in *Proceedings of the 29th international conference on Conceptual modeling*, 2010.

[14] D. Schumm, O. Turetken, N. Kokash, A. Elgammal, F. Leymann, and W. v. d. Heuvel, "Business process compliance through reusable units of compliant processes," Tilburg University, Open Access publications from Tilburg University, 2010.

[15] S. Simitis, *Bundesdatenschutzgesetz*. Frankfurt: Nomos, 2006.

[16] S. Grohe, "Visualisierung und Implementierung von compliance Scopes," Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, Mai 2011.

[17] D. Schleicher, T. Anstett, F. Leymann, and D. Schumm, "Compliant Business Process Design Using Refinement Layers," in *OTM 2010 Conferences*, T. D. et al. R. Meersman, Ed. Springer Verlag, Oktober 2010.