Business Process Data Compliance

Mustafa Hashmi[1,2], Guido Governatori[1,2] and Moe Thandar Wynn[1,2]

[1] NICTA, Queensland Research Laboratory, St. Lucia Australia
{mustafa.hashmi,guido.governatori}@nicta.com.au
[2] Queensland University of Technology (QUT) Brisbane, Australia
m.wynn@qut.edu.au

Abstract. Most approaches to business process compliance are restricted to the analysis of the structure of processes. It has been argued that full regulatory compliance requires information on not only the structure of processes but also on what the tasks in a process do. To this end Governatori and Saqid [2007] proposed to extend business processes with semantic annotations. We propose a methodology to automatically extract one kind of such annotations; in particular the annotations related to the data schema and templates linked to the various tasks in a business process.

Keywords: Business process, business process compliance, database schema, compliance by design

1 Introduction

Recently much interest has been seen in the business process management community on business process compliance due to the introduction of new regulatory laws such as Sarbanes-Oxley, BASEL II, and HIPPA to name a few. These laws impose severe penalties on violations. Hence enterprises are heavily investing to comply with internal or external policies thus the compliance software and services industry is booming rapidly. A recent survey by Deloitte Australia [1] reveals that, in Australia alone, estimated spending on compliance related activities in the public sector is around 4% of total IT spending (approaching the annual cost of AUD$1-billion), and compliance costs are expected to rise in coming years. No matter what they have to do, enterprises are obliged to streamline their daily business operations to the regulatory laws for transparency and better operations.

Enterprises develop process models to document and automate their operational activities. These process models provide an enterprise with a high-level view on how to achieve their business objectives and implement regulatory policies governing these processes. Hence process models can be used to verify the effectiveness of regulatory laws and/or policy controls. Furthermore, these models can also provide a view on the flow of data and relationships among the activities in the process, thus making a process model a natural venue to implement compliance related controls. Essentially compliance is a relationship between two distinct spaces with different objectives: *process modeling specifications space* and *business rules specifications space*. The business modeling specification space is procedural in nature, detailing how a business activity should take place. In contrast, business rules specifications are descriptive, dictating what need to be done to remain compliant [2].

Achieving balance between these two different worlds is not straightforward as a number of efforts have been reported in business process management literature [3–5]. However, predominantly much of these efforts have been limited to the development of descriptive approaches for BPM to achieve flexibility in business process execution [6] or restricted their attention to the analysis of the structure of business processes only [7, 8]. As compliance requirements come from different sources, it has been argued that to achieve full compliance it is inevitable to have complete information not only on the structure of processes, but also on what the tasks in a process do. To this end [2] proposed to enrich business processes with semantic annotations. Enhancing processes with these annotations allows process designers to implement, and see the control objectives within the process modeling space.

The idea to semantically annotate business processes is based on the notion of control tags. Control tags provide better understanding of the interaction between business process modeling specifications and business rule specifications. From a business process model perspective, there are four types of control tags: *control-flow*, *data*, *resources*, and *time* control tags [2]. These control tags consist of the state and operations of propositions about the conditions that are to be checked on a task; and are typed linked. In addition to that, control tags are not based on specific ontology, and may have associated constraints or policies. We build our work on the idea of these control tags with primary focus on the data control tags which identify the data retention and lineage requirements. For compliance checking purpose, the data control tags can be designed through parsing of Formal Contract Language[3](FCL) expressions, representing business rules. However, the problem is that how do we get the data for the data control tags; and where the data will come from. In addition to that, another question is how we can enforce the data constraints when annotating a process model with data tags. In this paper we are interested with the first two questions only: how to get data for data control tags; and from where. To address this problem, we proposed a query-based methodology to extract the data for control tags to annotate process models.

Business rules can be used for a variety of purposes. One particular application of business rules is to capture constraints on the data used in/by an application. Business rules provide a declarative approach to model such constraints and typically, they do not force specific technology and implementations.

A business process is a self-contained description of the activities to be done to achieve particular business objectives, the order in which the activities have to be done, the data the process operates on, and the resources required by the process. In this paper we restrict our focus on the data aspect. Figure 1 shows the links between the data and a task of a process model. Typically, there are three possible ways to provide data to a task in a process: (i) *the tasks receives the data from a previous task*, (ii) *the task is given data from a user*, and (iii) *the task reads data from a database linked to the process*. We can reverse the direction for the data produced by a task. In general for the data interactions we assume that: (a) *users interact with a process using forms and get reports back following some specific templates*, and (b) *data passes from one task to*

[3] *FCL: a formalism to express normative specifications.*

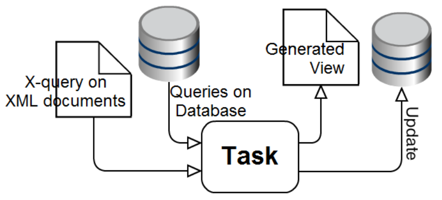


Fig. 1. Links between Data and Task in Process Model

the other using messages (according to specified templates or schemas). Thus data can be obtained by querying a database or parsing (XML) documents. The data produced, again is obtained by queries on databases (including the generation of views).

A typical scenario for the methodology we are going to discuss in the rest of paper is that where we have document-centric business processes, and where there is an organization that provides requirements for what data has to be in the documents, and how the documents are handled. The issue is to provide compliance certification for a process implementing the business rules specifications. A prototypical example is that of electronic lodgment of applications (for which we provide a simplified scenario based on a real life case, in Section 2).

Since we focus on compliance (i.e., design-time verification of the alignment of two sets of specification), it is not possible to have the actual data of instances of a process. Accordingly, the data control tags are not about the data of an instance case of a process, but on the schema of the databases and the parameters of document templates. Thus the research question of the paper is: *how to extract relevant information from the schema of the databases linked to a process*.

The organization of the paper as follow: in Section 2, we introduce a motivating scenario to set the stage to present our methodology. A short discussion on business process compliance follows in Section 3. The basics of formal contract language (FCL) will be presented in Section 4 after a short discussion on modeling normative requirements. In Section 5, we show how FCL can be used to model the business rules of our motivating example. Section 6 will outline our proposed compliance by design schema extraction methodology, followed by a review of latest research in the problem domain in Section 7. Concluding remarks and an outlook on future work will be presented in Section 8.

2 Scenario: Lodgment Verification Process

To present our proposed methodology consider a hypothetically simple lodgment case verification process aiming to verify whether the lodgment case comply with all designated rules, and whether it is in an acceptable form for further processing (cf. Figure 2). The data verification process can generate the following response:

- *an indication that the elodgment case meets all requirements for the lodgment*, or

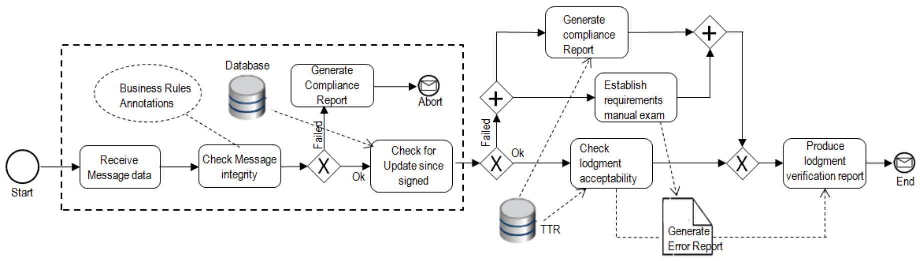

Start
Receive Message data
Business Rules Annotations
Database
Check Message integrity
Failed
Generate Compliance Report
Abort
Ok
Check for Update since signed
Failed
Ok
Generate compliance Report
Establish requirements manual exam
Check lodgment acceptability
TTR
Generate Error Report
Produce lodgment verification report
End

Fig. 2. Lodgment Case Process

- *a list of business rules with which that elodgment case does not comply, including registration requirements, and/or where required*,
- *a list of the manual examination processes that need to be performed on that elodgment case following the lodgment*.

The process starts with a verification request message from the subscriber containing message data items included in the message header such as *electronic lodgment notice workspace ID*, *request message type*, *system request ID* (from where message originates), *operator ID*, *lodgment case ID* just to name a few. The application consists of four documents:

- Electronic Lodgment Case (*eLC*),
- eConsent Information Report (*eCIR*),
- eLodgment Information Report (*eLIR*), and
- eNotice of Sale Information Report (*eNoSIR*).

In addition to basic data in the verification request message data, additional information pertaining lodgment case must be present in these documents. The main objective of the verification process is to verify that these documents are present in an application, and that they contain the information required to verify the suitability of the lodgment case. Furthermore, the required data must be in the required format.

There are four documents associated with the request message data of the lodgment case process. The request message must contain data items from these documents. The data requirements are expressed by the following business rules.

Business Rules

BR1 Each eConsent Information Report must specify exactly one ELN (Electronic Lodgment Notice) workspace ID.

BR2 The ELN workspace ID specified in each eConsent Information Report must be the same as the ELN workspace ID specified in the eLodgment Information Report in the eLodgment Case that includes that eConsent Information Report.

BR3 Each eConsent Information Report must specify exactly one ELN eLodgment Case ID.

BR4 The ELN eLodgment Case ID specified in each eConsent Information Report must be the same as the ELN eLodgment Case ID specified in the eLodgment Information Report ID in the eLodgment Information Report, in the eLodgment Case that includes that eConsent Information Report.

BR5 Each eNoS Information Report must specify exactly one ELN eLodgment Case ID.

BR6 The ELN elodgment Case ID specified in each eNoS Information Report must be the same as the ELN eLodgment Case ID specified in the eLodgment Information Report included in the eLodgment Case that includes the eNoS Information Report.

In the next sections we introduce the methodology for business process compliance and the formalism we are going to use to formalize the above business rules.

3 Business Process Compliance

The main objective of business process compliance is to ensure that businesses perform their operations in accordance with regulatory laws and/or internal policies. Business rules on one hand, and business process modeling on the other, are two separate worlds with different objectives. The business rules specifications (a.k.a *normative specifications*) dictate what business has to do, in contrast, process modeling specifications describe how a business activity is performed. To properly verify that a business process is fully compliant with designated normative specifications, it is compulsory to provide a conceptually rich representation of both normative specifications and business process modeling specifications. This defines what obligations and permissions a business process is subject to. To capture the real intention of the business rules and for effective compliance checking, a formally rich representation of normative specifications is mandatory and challenging. We follow the methodology proposed by Governatori and Sadiq [9, 2, 10] for business process compliance. The key aspects of the methodology are: (1) *to enrich business process models with semantic annotations*, (2) *to extract control objectives from business rules*, and (3) *to formalize the control objectives in an appropriate logical formalism*.

Annotations can be at the level of a business process or at the level of tasks, where each task can have its own annotations. The annotations for a task, essentially, provide additional information about what the task does (effects to the task), the resources involved in a task, the data associated or produced by a task. The effects of the tasks are accumulated over the tasks in an execution trace of the process using an update semantics [11–13], and compliance is checked based on the algorithm proposed in [13, 14]. Figure 3 depicts the architecture of business process compliance. The proposed methodology goes well beyond simple structural compliance (i.e. checking the structure of a business process). The cost for this is to have semantic annotations and properly modeling normative requirements. Most of the semantic annotations must be given by domain experts. However, in the rest of the paper we are going to show how annotations for data constraints on document-centric business processes can be extracted automatically from the schemas/templates associated to the process. In the next section we outline

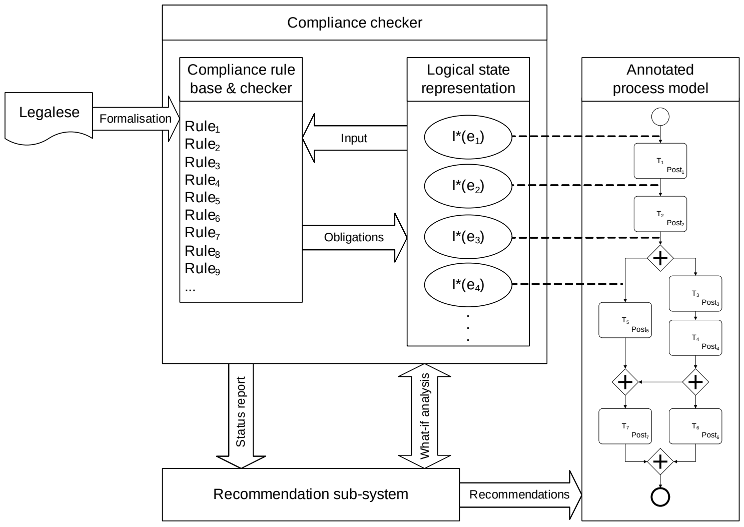


Fig. 3. Business Process Compliance Architecture: Adopted from [12]

how the formalism (FCL) proposed in [9] can be used for properly modeling normative requirements.

4 Modeling Normative Requirements

Deontic logic studies formal properties of normative specifications in terms of the so called normative positions (i.e. obligations, permissions and prohibitions). Deontic logic provides machinery to investigate relationships among different normative positions. A detailed discussion on deontic logic and Standard Defeasible Logic (SDL) is beyond the scope of this paper, the reader is directed to [9] for further reading. The problem with deontic logic is that, it is not capable of dealing with violations and the obligations arising from violated obligations [15]: in some situations business rules may specify conditions about when other conditions in business rules books have been violated (i.e. some clauses of the rules have been violated). Deontic logic does not provide a faithful representation of such situations. Governatori [16] proposed Formal Contract Language (FCL), a formalism to analyze business contracts and address the deficiencies of deontic logic by providing rich representation of contract violations. We used FCL to provide formal representation of lodgment verification process. FCL is a rich combination of an efficient non-monotonic formalism (defeasible logic (cf. [17, 18]), and deontic logic of violations (Governatori and Rotolo [19]) which enables us to present exceptions as well as ability to capture violations. Moreover, FCL provides for a conceptually rich formalization of norms for compliance checking of a process where partial information and possibly conflicting provisions are present.

FCL consists of two sets of atomic symbols: a numerable set of propositional letters $a, b, c \ldots$ that represent the state variables and the tasks of a process. Formulas of the logic are built using the deontic operators O (for obligations), P (for permission), negation $\neg$ and the non-structural connective $\otimes$ (for the contrary to duty operator). An FCL formula is defined in a two-step process under the following formation rules:

- every propositional letter is a literal;
- the negation ($\neg$) of a literal is a literal;
- if X is deontic operator and l is a literal then Xl and $\neg Xl$ are deontic literals.

In addition we introduce the notion of $\otimes$-expressions.

- every literal is an $\otimes$-expression;
- if $l_1, \ldots, l_n$ are literals, then $l_1 \otimes \cdots \otimes l_n$ is an $\otimes$-expression

In FCL each business rule statement or any condition applying on a process is represented by a rule, where a rule is an expression

$$r : A_1, \ldots, A_n \Longrightarrow_O C$$

where r is the ID or name of a business rule statement, $A_1, \ldots, A_n$ is the *antecedent* of rule and C is conclusion of the rule. Each A_i is either a literal or deontic literal and C is an $\otimes$-expression. The meaning of the above expression is that normative position (e.g., obligations) represented by the conclusion of the rules is in force provided all premises of the rule hold. By using the $\otimes$ connective, we can combine the primary as well as contrary to duty obligations to form a unique rule e.g. $A \otimes B \otimes C$. The meaning of such expressions is very simple: A is the primary obligation, but if A is violated or not done, then B becomes the obligation as a replacement of A. Thus B becomes a reparation of the violation of A which means that A does not hold but the negation of A i.e. $\neg A$ holds. In addition, in case if B also fails, then now it is required to fulfill the obligation of C. Suppose we have the rules

$$r_1 : a \Longrightarrow_O b \qquad\qquad r_2 : c, Ob \Longrightarrow_O \neg d \otimes e$$

and we have that a and c and d hold. From the first rule we obtain the b is obligatory (Ob), and then we can apply r_2. This rule produces $O\neg d$ (d is forbidden or $\neg d$ is obligatory). Rule r_2 also states that the violation of the prohibition of d is compensated by e. Thus, since we have d, we violated the prohibition, and now we have the obligation to compensate it, that is Oe. See [16] for full details of FCL.

5 Modeling Control Objective and Business Rules

Based on the compliance methodology proposed by [2], we generate control objectives corresponding to the business rules given above. For each control objective we identify the relevant document, data items and constraints. We present how FCL intuitively captures the meanings of business rules and provides a faithful representation of normative specifications for lodgment verification process scenario presented in Section 2.

The first step is to introduce the logical predicates needed for the representation of the control objectives and business rules. Here we assume that the relevant data is stored in a database. We will have two types of predicates. The predicates in the first class essentially correspond to the attributes in the database. Thus we have predicates representing the tables and the attributes in the database. In the second class we have the predicate $contains(x,y)$. The meaning of it is 'document x contains information/data value y'.

In the first class we have the predicates (with their meaning)

- $eLC(x)$: x is an Electronic Lodgment Case;
- $eCIR(x)$: x is an eConsent Information Report;
- $eLIR(x)$: x is an eLodgment Information Report;
- $eNoSIR(x)$: x is a Notice of Sale information Report;
- $ELNws(x)$: x is a ELN workspace.

We are now ready to provide the control objectives and the formalization of the rules.

Business Rules: BR1 and BR2
Document Type: eConsent Information Report
Data Item: ELN Workspace ID
Constraints on Data Item:

1. Exactly one must be present
2. Must be the same as the ELN workspace ID in the eLodgment information report in the same elodgment case

Mapping:
$r_{1,1}: ELNws(x), eCIR(y) \Longrightarrow_O contains(y,x)$
$r_{1,2}: ELNws(x), eCIR(y), ELNws(z), x \neq z, contains(y,x) \Longrightarrow_O \neg contains(y,z)$
$r_2: \; ELNws(x), eCIR(y), eLIR(z), eLC(u), contains(y,x), contains(u,y),$
$\quad contains(u,z) \Longrightarrow_O contains(z,x)$

The meaning of this formal representation is that, the predicate *ELNws* must be present exactly once in the *eConsent Information Report eCIR*. Rule $r_{1,1}$ specifies that if x is the ID of an ELN workspace, and y is the ID of eConsent Information Report, then it is obligatory that the value of the workspace ID appears in the eConsent Information Report. Rule $r_{1,2}$ states that if x and z are different workspaces (workspace IDs) and one of them is present in the eConsent Information Report identified by y, then it is forbidden for the other workspace ID to appear in y.

Rule $r2$ first identifies the type of several documents (e.g. eConsent Information Report, eLodgment Information Report, eLodgment Case), and the ID for the ELN workspace. In addition, if the eConsent Information Report contains a reference to a ELN wokspace ID ($contains(y,x)$), and the eConsent Information Report is part of an eLodgment Case ($contains(u,y)$), and there is an eLodgment Information Report that is part of the same application ($contains(u,z)$), then the eLodgment Information Report must contain a reference to the same ELN workspace $contains(z,x)$.

Business Rules: BR3 and BR4
Document Type: eConsent Information Report
Data Item: ELN eLodgment Case ID
Constraints on Data Item:

1. Exactly one must be present
2. Must be the same as the ELN elodgment case ID in the elodgment information report in this elodgment case

Business Rules: BR5 and BR6
Document Type: eNoS Information Report
Data Item:ELN eLodgment Case ID
Constraints on Data Item:

1. Exactly one must be present
2. Must match the ELN elodgment Case ID in the elodgment information report

The formal rules for the control objectives for Business Rules BR3–BR6 have the same formal representation of the rules for BR1 and BR2.

These control objectives and the resulting FCL rules will provide us guidance on (1) what elements of a lodgment document are relevant for compliance, and consequently which tables and attributes must be extracted from the database schema, and (2) how to formally model the business rules.

6 The Schema Extraction: Compliance Methodology

In this section we provide the account of our proposed methodology which explicitly shows how we can extract the schema for the required data from the databases linked to a process. Figure 4 gives an overview of the overall methodology. As was mentioned in Section 1, process models can be enriched with data in the form of data control tags as required by the process to complete a specific task for compliance verification, and the question was raised where these data annotations will come from. We propose to essentially extract these annotations by querying the database created from the analysis of business rules statements. We use abstract business rules with no information on the processes and identified all pertaining entities involved in the data verification process by means of Entity Relationship (ER) diagram as shown in Figure 5. In the data verification process, each lodgment case has several associated documents with defined attributes. These documents may have several identical and distinct attributes, we have not listed all the attributes in the ER diagram and just give a nominal representation of the data items. From the ER model, the database schema for the lodgment case has been extracted comprising several database tables corresponding to each associated document such as *e_consentinformationreport*, *e_registryinstrument*, *enos_informationreport* etc., and tables for request and response messages. The lodgment case table contains information about the lodgment case for which the data verification request message is sent. In the created database, there are a number of system tables that are automatically created containing information about the database such as columns and key constraints tables. Figure 6 shows the schema for lodgment case database consisting of base tables for each of the associated documents and their attributes, data types, and primary keys. We can query the database to extract the predicates (attributes) of our business rules.

The query to extract the predicates *ELNws*, *eCIR*, *eLIR*, *eNoSIR* and *eLC* is[4]

[4] For space and readability reasons, we use abbreviation for the predicates, and full names for the database. It would have been possible to use the same names, or to establish a one-to-one

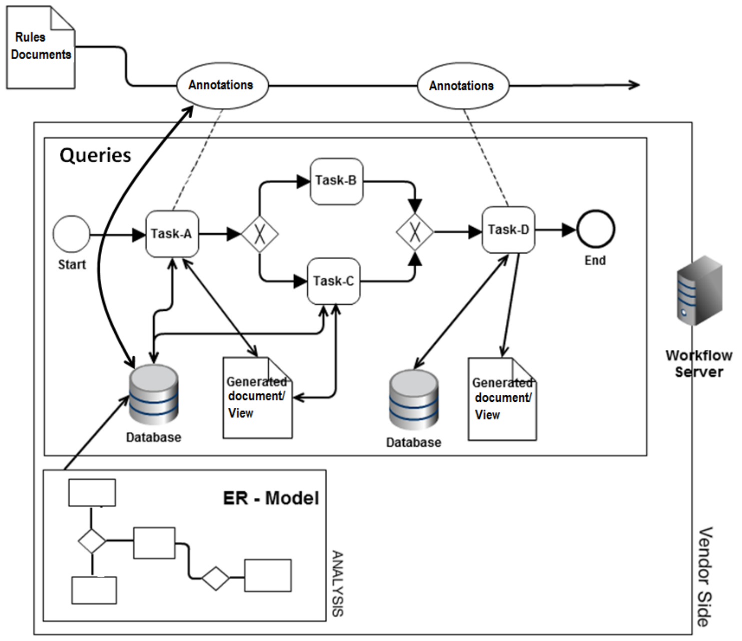


Fig. 4. The Database Schema Extraction Methodology

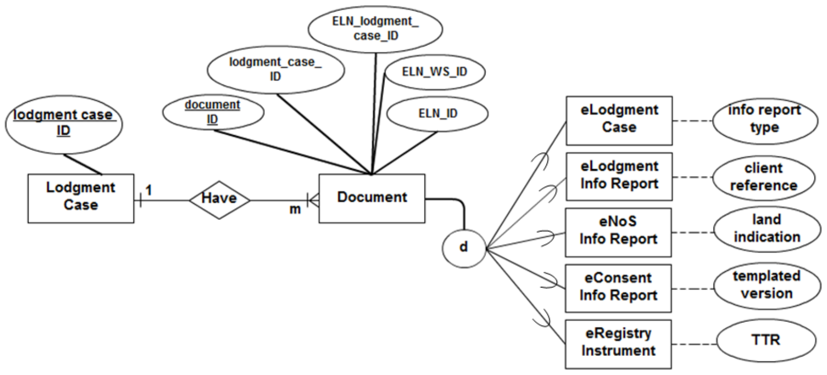
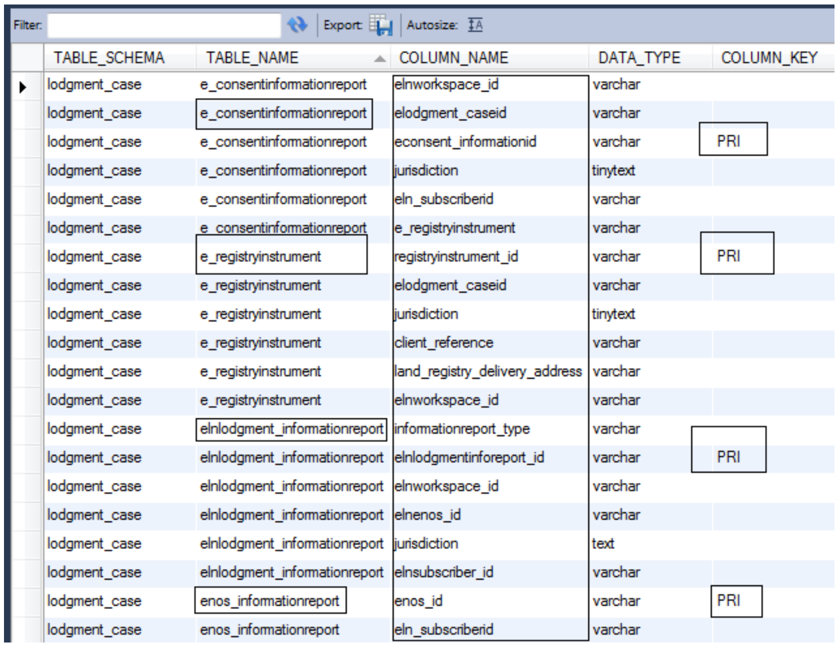


Fig. 5. ER-Diagram for Data Verification



TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_KEY
lodgment_case	e_consentinformationreport	elnworkspace_id	varchar	
lodgment_case	e_consentinformationreport	elodgment_caseid	varchar	
lodgment_case	e_consentinformationreport	econsent_informationid	varchar	PRI
lodgment_case	e_consentinformationreport	jurisdiction	tinytext	
lodgment_case	e_consentinformationreport	eln_subscriberid	varchar	
lodgment_case	e_consentinformationreport	e_registryinstrument	varchar	
lodgment_case	e_registryinstrument	registryinstrument_id	varchar	PRI
lodgment_case	e_registryinstrument	elodgment_caseid	varchar	
lodgment_case	e_registryinstrument	jurisdiction	tinytext	
lodgment_case	e_registryinstrument	client_reference	varchar	
lodgment_case	e_registryinstrument	land_registry_delivery_address	varchar	
lodgment_case	e_registryinstrument	elnworkspace_id	varchar	
lodgment_case	elnlodgment_informationreport	informationreport_type	varchar	
lodgment_case	elnlodgment_informationreport	elnlodgmentinforeport_id	varchar	PRI
lodgment_case	elnlodgment_informationreport	elnworkspace_id	varchar	
lodgment_case	elnlodgment_informationreport	elnenos_id	varchar	
lodgment_case	elnlodgment_informationreport	jurisdiction	text	
lodgment_case	elnlodgment_informationreport	elnsubscriber_id	varchar	
lodgment_case	enos_informationreport	enos_id	varchar	PRI
lodgment_case	enos_informationreport	eln_subscriberid	varchar	

Fig. 6. Column Schema for Lodgment Case Database

```
SELECT TABLE_NAME, COLUMN_NAME
FROM SYSTEM_INFORMATION
WHERE COLUMN_KEY = 'PRI'
```

After that we have to take the `TABLE_NAME` as the name of the predicate and the `COLUMN_NAME` as the argument of the predicate. For example, one of such predicates would be $enos\_informationreport(enos\_id)$, or if we use the mapping with the abbreviations $eNoSIR(enos\_id)$. The meaning for ground instances of these predicates is, "column_name"is the primary key of "table"; thus eNoSIR($enos\_id$) means $enos\_id$ is the identifier of an eNoSIR document. This kind of predicates, where the variables (arguments) of the predicates have names, are helpful in other respects. For example it could be used to check conformance (compliance at run time, or that the data in an instance of a process is correct). We can instantiate a predicate using the following schema for a query:

```
SELECT $COLUMN_NAME
FROM $TABLE_NAME
```

So to get the extension of the predicate *eNoSIR*, we run the query

mapping between the elements in the data dictionary of the rules and the elements in the data dictionary of the datebases linked to a process.

```
SELECT ENOS_ID
FROM ENOS_INFORMATIONREPORT
```

The abstract extension of the predicate *contains* can be computed by the following query

```
SELECT X.COLUMN_NAME, Y.COLUMN_NAME
FROM SYSTEM_INFORMATION as X, SYSTEM_INFORMATION as Y
WHERE X.TABLE_NAME = Y.TABLE_NAME
AND X.COLUMN_KEY = 'PRI'
```

The query is a simple self-join of the system information table, using `TABLE_NAME` as the join attribute, and it returns pairs of column names for columns in the same table, where the first is the primary key of the table. Thus for example using the data in Figure 6 we have the pair *registryinstrument_id* and *elodgmentcase_id*. Every row of the table *e_registryinstrument* has the ID of an *e_registryinsturment* and the ID of the *elodgmentcase*. In other words each row represents a document of a given type, the information in it, and the primary key represents the document. Thus *registryinstruemnt_id* and *elodgmentcase_id* means that the information report about an e_registryinstrument contains a reference to the elodgmentcase of which it is part of. Notice that the first query and the last query are domain independent. All we need is a system table with the information on the schema of the database used by a process.

Based on the idea presented above, checking data compliance (of a database schema against a set of business rules defining the data constrains) can be simply performed by running the above query on the database linked to a particular task to get the (data) annotations for that task. After that we can use a two step compliance checking algorithm proposed by [13, 14] which, in the first step, examines each task in the process against all relevant obligations; and generates a status report on active reparation chains. Then, in the second step, it determines if a process is compliant with all regulations or not.

7 Related Work

In recent past a number of approaches focusing on checking compliance on business process models have been reported in literature [3, 20–23]. As we discussed previously about the requirement of a preventive approach *compliance by design* for business process compliance. This literature can be divided into two distinct categories: compliance by design and post design compliance checking. In the first approach new business process models are fed with business rules as input whereas a process model is checked against compliance requirements when a process has completed the design phase.

Lu et. al [24] objectively showed how to enforce compliance requirements to avoid the chance of potential rules violations. Similar works reported by [25–27], although provide good solution to achieve design-time compliance yet compliance checking will be required if changes are made to the process model, and new business rules are introduced. In addition to that, the emphasis of these approaches remained on the structural compliance of a process model, and the data aspect has largely been ignored. Goedertier and Vantienen [25] achieved design-time business process compliance using rule sets with permissions and obligations and proposed PENELOPE, a declarative language to specify compliance rules. PENELOPE generates a state space and a BPMN model from these rules which is compliant by design. This approach concentrates on acyclic processes only, and the data and data constraints aspect in the business rules is not present. An artifact-centric business process modeling approach has been recently proposed in [28], exhibiting how artifact-centric business processes can be canonically extended to take also compliance rules into account. As these business rules can express constraints on the execution of actions, it is claimed that the data information can also be taken into account but it is not clear whether the model will be semantically annotated with the data, and how data constraints will be modeled. If in case business process model is semantically annotated then where this data will come from.

In the post process model design compliance checking Awad et.al. [29] discussed a temporal logic query based approach for specification, verification and explanation of violations of data-aware compliance rules. The approach employs extended BPMN-Q to realize the business rules including the data aspects to increase the expressiveness of their previously proposed language in [30]. As the authors used *past linear temporal logic* (PLTL) to formalize the business rule, the problem with temporal logic is that it provides structural compliance only, and does not distinguish different normative positions. There is no indication how these normative positions and data associated with these normative positions can be represented. Moreover, this proposed approach comes under the post design compliance checking. To measure the compliance distance between the process model and a rule an automated approach was introduced in [31]. The degree of compliance is checked on a scale from 0 to 1 but the data aspect has not been covered.

Our work reported in this paper falls in latter category to achieve compliance by design. To our end, we believe until recently no work has been reported that specifically extract data schema from the business rules to semantically annotate a process model for compliance checking purpose.

8 Conclusions and Future Work

In this paper we proposed a methodology to automatically extract annotations related to the data schema, and templates linked to the tasks in a process. This exhibits how we can extract the data schema from the database generated from the business rules, including the primary keys of the associated documents of the lodgment verification process presented in Section 2. We see the contribution of this work in different ways: *first* our methodology will provide a better understanding of data annotations from schema related to tasks a the process. *Second:* the BPM process model, at hand, can be extended with the extracted annotations in the form of data tags as proposed in [2]. In addition to that, the extracted schema will also help to model data constraints on the process model for better compliance checking. *Third:* this methodology provides an answer to the question where do we get the data annotations from, if we want to extend a process model with these annotations, and model constraints on a process model.

Currently we have used abstract data, and a hypothetically created process example to present our methodology to show how we can extract annotations from abstract data to extend a business process model. As this is a preliminary work and has not been implemented yet, we are not aware of any complexity that might arise when extending a process model. On the same note, we are also unable to report, at this stage, what will be the behavior of process model populated with the extracted data schema. Hence we believe this proposed preliminary work requires a large scale industry evaluation and validation for further insights and generalization. In addition to that, due to the varying nature of business process tasks, amount of data used, and data redundancies are prevalent in the business rules statements, extracting a normalized data schema will certainly be a challenge. In our example case scenario, we experienced many redundant data items appearing several times in the business rules, we just used this redundant data to extract preliminary schema. The positive aspect of using FCL is that, it provides normalization functionalities to remove any redundancies in the business rules statements, we believe that this matter is solvable and of further interest. Lastly, but no least, as business rules tend to change frequently, a business analyst can come up with new predicates. This requires further understanding how these changes can be accommodated in the existing database.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Center of Excellence program.

References

1. Australia, A.I.G.D.: The Australian Industry Group National CEO Survey: Business Regulations (September 2011)
2. Sadiq, S., Governatori, G., Namiri, K.: Modeling Control Objectives for Business Process Compliance. In: BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
3. Liu, Y., Müller, S., Xu, K.: A Static Compliance-Checking Framework for Business Process Models. IBM Systems Journal 46, 335–361 (2007)
4. Schmidt, R., Bartsch, C., Oberhauser, R.: Ontology-based Representation of Compliance Requirements for Service Processes. In: SBPM 2007. CEUR, vol. 251, pp. 28–39 (2007)
5. El Kharbili, M., Stein, S., Markovic, I., Pulvermüller, E.: Towards a Framework for Semantic Business Process Compliance Management. GRCIS08 Workshop at CAiSE08. CEUR, vol. 339, pp. 1–15 (2007)
6. Hagerty, J.: Sox spending for 2006. ARM Research, Boston USA (November 2006) JH06.
7. Ly, L.T., Knuplesch, D., Rinderle-Ma, S., Goeser, K., Reichert, M., Dadam, P.: Seaflows toolset — compliance verification made easy. In: CAiSE Forum 2010. LNBIP, vol. 72, pp. 76-91. Springer, Heidelberg (2010)
8. Ly, L.T., Rinderle, S., Dadam, P.: Semantic Correctness in Adaptive Process Management Systems. In: BPM 2006. LNCS, vol. 4102, pp. 193–208. Springer, Heidelberg (2006)
9. Governatori, G., Sadiq, S.: The Journey to Business Process Compliance. In: Handbook of Research on Business Process Management. pp. 426–454. IGI Global (2009)
10. Sadiq, S., Governatori, G.: A Methodological Framework for Aligning Business Processes and Regulatory Compliance. In: Handbook of business process management: 2. Strategic alignment, governance, people and culture. pp. 159–176. Springer, Berlin (2010)

11. Ghose, A., Koliadis, G.: Auditing Business Process Compliance. In: ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
12. Governatori, G., Hoffmann, J., Sadiq, S.W., Weber, I.: Detecting Regulatory Compliance for Business Process Models Through Semantic Annotations. In: Business Process Management Workshops'08. LNBIP, vol. 17, pp. 5–17. Springer, Heidelberg (2008)
13. Governatori, G., Rotolo, A.: An Algorithm for Business Process Compliance. In: Legal Knowledge and Information Systems. pp. 186–191. IOS Press (2008)
14. Governatori, G., Rotolo, A.: A Conceptually Rich Model of Business Process Compliance. In: APCCM 2010. CRPIT, vol. 110. pp. 3–12. Australian Computer Society (2010)
15. Carmo, J., Jones, J.: Deontic Logic and Contrary to duties. In: Handbook of Philosophical Logic: 2nd Edition. Dordrech: Kulwer (2002) 265–343
16. Governatori, G.: Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems 14, 181–216 (2005)
17. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2, 255–287 (2001)
18. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Embedding defeasible logic into logic programming. Theory and Practice of Logic Programming 6, 703–735 (2006)
19. Governatori, G., Rotolo, A.: Logic of Violations:A Gentzen System for Reasoning with Contrary-To-Duty Obligation. Australasian Journal of Logic 4, 193–215 (2006)
20. Bonazzi, R., Pigneur, Y.: Compliance Management in Multi-actor Contexts. In: GRCIS 2009. CEUR, vol. 459, article 7 (2009)
21. COMPAS: Compliance Driven Models, Languages, and Architectures for Services. 7th Framework Programme Information and Communication Technologies (2008)
22. el Kharbili, M., Stein, S.: Policy-Based Semantic Compliance Checking for Business Process Management. In: MobIS Workshops 2008. CEUR, vol. 420, pp. 178–192 (2008)
23. Ly, L., Rinderle-Ma, S., Göser, K., Dadam, P.: On Enabling Integrated Process Compliance with Semantic Constraints in Process Management Systems. Information Systems Frontiers 14, 195-219 (2012)
24. Lu, R., Sadiq, S., Governatori, G.: Compliance aware business process design. In: Business Processes Management Workshops 2007. LNCS, vol. 4928, pp. 120–131. Springer, Heidelberg (2007)
25. Goedertier, S., Vanthienen, J.: Designing Compliant Business Processes with Obligations and Permissions. In: Business Process Management Workshops 2006. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
26. Governatori, G., Milosevic, Z., Sadiq, S.: Translating business contract into compliant business processes. In: 10th IEEE International Enterprise Distributed Object Computing Conference. pp. 221–232. IEEE Press (2006)
27. Goedertier, S., Vanthienen, J.: Compliant and flexible business processes with business rules. In: BPMDS 2006. CEUR, vol. 236. (2006)
28. Lohmann, N.: Compliance by design for artifact-centric business processes. In: BPM'11, LNCS, vol. 6856, pp. 99–115. Springer, Heidelberg (2011)
29. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: ICSOC 2009. LNCS, vol. 5900, pp. 500–515. Springer, Heidelberg (2009)
30. Ahmed, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: BPM'08. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
31. Lu, R., Sadiq, S., Governatori, G.: Measurement of compliance distance in business processes. Information Systems Management 25, 344–355 (2008)