Compliance Aware Business Process Design

Ruopeng Lu, Shazia Sadiq, and Guido Governatori

School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane, Australia
{ruopeng, shazia, guido}@itee.uq.edu.au

Abstract. Historically, business process design has been driven by business objectives, specifically process improvement. However this cannot come at the price of control objectives which stem from various legislative, standard and business partnership sources. Ensuring the compliance to regulations and industrial standards is an increasingly important issue in the design of business processes. In this paper, we advocate that control objectives should be addressed at an early stage, i.e., design time, so as to minimize the problems of runtime compliance checking and consequent violations and penalties. To this aim, we propose supporting mechanisms for business process designers. This paper specifically presents a support method which allows the process designer to quantitatively measure the compliance degree of a given process model against a set of control objectives. This will allow process designers to comparatively assess the compliance degree of their design as well as be better informed on the cost of non-compliance.

Keywords: Business Process Design, Process Compliance Control, Business Process Modeling.

1 Background and Motivation

Compliance essentially means ensuring that business processes, operations and practice are in accordance with a prescribed and/or agreed set of norms. Compliance is increasingly gaining importance as well as raising the pressure for organizations in practically all industry sectors. Although this is not a new issue, but recent events, particularly high profile corporate scandals, as well as new regulations such as the Sarbanes-Oxley act have raised a new set of challenges for businesses.

Compliance is predominantly viewed as a burden, although there are indications that businesses have started to see the regulations as an opportunity to improve their business processes and operations. Industry reports [7] indicate that up to 80% of companies said they expected to reap business benefits from improving their compliance regimens.

Currently there are two main approaches towards achieving compliance. First is *retrospective reporting*, wherein traditional audits are conducted for "after-the-fact" detection, often through manual checks by expensive consultants. A second and more recent approach is to provide some level of automation through *automated detection*. The bulk of existing software solutions for compliance follows this approach. The proposed solutions hook into variety of enterprise system components (*e.g.* SAP HR, LDAP Directory, Groupware etc.) and generate audit reports against hard-coded checks performed on the requisite system. These solutions often specialize in certain class of checks, for example the widely supported checks that relate to Segregation of Duty violations in role management systems. A major issue with the two discussed approaches is the lack of sustainability. Even with automated detection facility, the hard coded check repositories can quickly grow out of control making it extremely difficult to evolve and maintain them for changing legislatures and compliance requirements. The complexity of the situation is exasperated by the presence of dynamically changing collaborative processes shared with business partners. The diversity, scale and complexity of compliance requirements warrant a highly systematic and well-grounded approach.

We believe that a sustainable approach for achieving compliance should fundamentally have a preventative focus, thus achieving *compliance by design*. Incorporating compliance issues within business process design methodology can assist process designers in tackling this complex issue using known successful strategies. However, at the same time, there is evidence that dealing with compliance may be a rather distinct activity from business process management within organizational structures.

This paper presents a particular method to study the relationship between compliance requirements modeled as controls, and process requirements modeled as business process models. Specifically we will present a quantitative measure of compliance for a given process model against a set of control objectives. The associated methods will allow process designers to comparatively assess the compliance degree of their design as well as be better informed on the cost of non-compliance.

Related work can be found in the research of [1, 2, 3, 10]. Space does not allow further elaboration of these works, but a distinctive feature of our work is that most related works present solutions for runtime monitoring, where as we focus on design time support.

The remaining paper is structured as follows. Section 2 presents the underlying methodology for *compliance aware* business process design. In section 3, we present the technique to quantitatively measure the degree of compliance during business process design. We conclude this paper in section 4.

2 Compliance by Design Methodology

Regulations and other compliance directives are complex, vague and require interpretation. Business will typically deal with a number of regulations/standards at one time. Often in legalese, these mandates need to be translated by experts. Tackling this issue warrants a systematic methodology [9].

Firstly, there is a need to provide a structured means of managing the various (expert) interpretations within regional, industry sector and organizational contexts. As a first step, a facility for *control directory management* (e.g. SAP GRC Repository) needs to be realized by repositories of control objectives (and associated parameters) against the major regulations.

Interpretation of regulations from legal /financial experts comes in the form of textual descriptions (see the examples in Section 2.1). *Establishing an agreement* on terms and usage between these descriptions and the business processes and constituent activities/transactions is a difficult but essential aspect of the overall methodology. However, it is evident that several controls may be applicable on a given business task, and one control may impact on multiple tasks as well.

A fundamental question in this regard is the appropriate formalism to undertake the task of *representing controls objectives* in a precise and unambiguous manner. Our observation is that a compliance requirement (or its translation into a control objective and subsequently internal controls) can be reduced to the identification of what obligations an enterprise has to fulfill to be deemed as compliant.

The motivation to model control objectives is multifaceted: Firstly, a generic requirements modeling framework for compliance by design will provide a substantial improvement over current after-the-fact detection approaches. Secondly, it will allow for an analysis of compliance rules thus providing the ability to discover hidden dependencies, and view in holistic context, while maintaining a comprehensible working space. Thirdly, a precise and unambiguous (formal) specification will facilitate the systematic enrichment of business processes with control objectives.

Subsequent to the modeling of control objectives, there is a need to provide the ability to *enhance enterprise models* (business processes) with compliance requirements. This may constitute visualization schemes [9], which facilitates a better understanding of the interaction between the two specifications for both stakeholders (process owners as well as compliance officers).

However, the visualization is only a first step. The new checks introduced within the process model, can in turn be used to analyse the model for measures such as *compliance degree* that can provide a quantification of the effort required to achieve a compliant process model. Eventually, process models may need to be modified to include the compliance requirements.

In this paper, we are focused on this last aspect, that is to assist process designers in creating compliant business processes. The presence of the previous phases of the methodology is assumed. As such, the goal of this so-called *compliance aware business process design* is to design the process while keeping track of relevant control objectives and ensuring that high risk controls are not ignored or violated.

In the rest of the paper, we first discuss the approach to model the controls objectives and present an appropriate language for their representation, followed by a simple formalization for the business process model. We then introduce the technique to map the controls objectives and the process model into a canonical form, such that the degree of compliance in the process model can be compared with regard to the controls objectives. The subsequent discussion is based on a sample procurement process (*cf.* Figure 1).

The procurement process may be subject to a number of control objectives from various restrictions such as regulations, industrial standards and partner obligations etc. The control objectives will typically have a corresponding risk statement, and a translation to an internal control indicating effective implementation of the control objective.

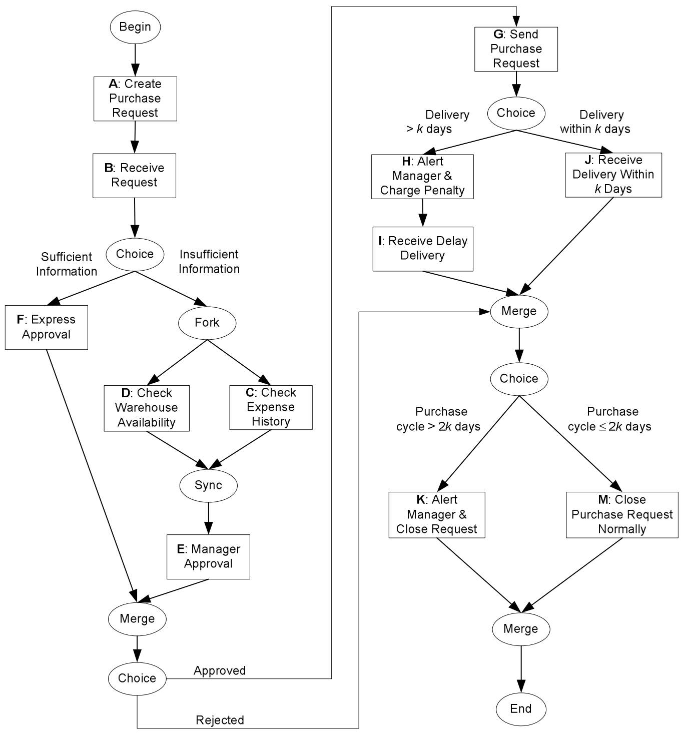

Fig. 1. Example procurement process

Table 1. Control objectives of the procurement process

Control Objective	Risk	Internal Control
Process efficiency	Process delays due to repeated or additional activities.	Purchase request with necessary information should be fast-tracked without management level approval.
Ensure adequate supply of materials	Production delays due to lack of resources/ materials	Supplier can be charged a penalty if goods not received within *k* days of receipt of goods shipment notice.
Timely and efficient procurement process	Production delays due to lack of resources/ materials	Purchase requests not closed (declined or converted to Purchase Orders) within 2*k* days should raise an alert to purchasing manager.

Table 1 provides examples of such control objectives for the procurement process. Typically, these internal controls cover multiple aspects of business process, including:

– Model structure, *e.g.*, task execution restrictions (every purchase order must be initially checked before passing to the Manager for approval).
– Data integrity, *e.g.*, every Purchase Order must contain a valid purchase request number.

– Resource allocation, *e.g.*, segregation of duty constraint (the creation and approval of purchase order must not be by the same officer).
– Temporal restrictions, *e.g.*, deadline (all purchase requests must be closed within $2k$ days).

2.1 Modeling Control Objectives

Although our work is primarily targeted at achieving compliance by design by adopting a preventative approach facilitated by business process models, the work on formal modeling of control objectives has taken into account the violations and resultant reparation policies that may surface at runtime. The objective is to be able to examine how compliant the (possible) runtime behaviors of a process model is with regard to the control objectives. We consider the behaviors of a process model to be reflected by actual *execution sequences* (of tasks in the process). The focus is then on the measurement for how "close" between the behaviors of the process model, and the compliance controls. To allow for the comparison, the formal representations of compliance controls, and the model behaviors (execution sequences) are given.

The compliance controls can be represented in a formal language, such as Formal Contract Language (FCL) [4, 5]. FCL is a combination of an efficient non-monotonic formalism (defeasible logic) and a deontic logic of violations. We illustrate how to use this formalism to represent and reason about "normative" specifications relative to a business process. For detailed presentation of the rationale and formalism of FCL, we refer to [4, 5].

Definition 1 (FCL Rule). A rule in FCL is an expression of the form

$$r\colon A_1,\ldots, A_n \Rightarrow B$$

where r is the name of the rule (unique for each rule), $A_1,\ldots, A_n$ are the premises (propositions in the logic), and B is the conclusion of the rule (also a proposition of the logic).

The propositions of the logic are built from a finite set of atomic propositions, and the following operators: $\neg$ (for negation), O (for obligation), P (for permission), and $\otimes$ (for violation/reparation). The formation rules are as follows:

– every atomic proposition is a proposition;
– if p is an atomic proposition, then $\neg p$, is a proposition;
– if p is a proposition then Op is an obligation proposition and Pp is a permission proposition; obligation propositions and permission propositions are deontic propositions
– if $p_1, \ldots, p_n$ are obligation propositions and q is a deontic proposition, then $p_1\otimes \ldots \otimes p_n \otimes q$ is a reparation chain.

A simple proposition corresponds to a factual statement. A reparation chain, for example $B_1\otimes B_2$ captures obligations and normative positions arising in response to violations of obligation. Thus the expression above means that it is obliged to perform

B_2, in case B_1 is not fulfilled (i.e., the obligation is violated) then the "secondary" obligation B_2 has to be fulfilled. The control objectives shown in Table 1 can be expressed in the following FCL rules:

Purchase request should be supplied with sufficient background information in order to streamline the approval process.

r_1: *CreatePurchaseRequest*, *ReceiveRequest* $\Rightarrow$ *ExpressApproval* $\otimes$(*CheckWareHouseAvailability*;*CheckExpenseHistory*;*ManagerApproval*)

Supplier can be charged a penalty if goods not received within k days of receipt of goods shipment notice, while manager should be alerted.

r_2: *SendPurchaseRequest* $\Rightarrow$ *ReceiveeDeliveryWithinkDays* $\otimes$(*ChargePenalty&AlertManager*;*ReceiveDelayDelivery*)

If purchase order is not closed within 2k days the manager should be alerted.

r_3: *ReceiveDeliveryWithinkDays* $\Rightarrow$ *ClosePurchaseRequestWithin2kDays* $\otimes$(*AlertManager&CloseRequest*)

r_4: *ReceiveDelayDelivery* $\Rightarrow$ *ClosePurchaseRequestWithin2kDays* $\otimes$(*AlertManager&CloseRequest*)

For the ease of discussion, we use the letters associated with each task on Fig. 1 to denote the tasks in the process model. r_1 - r_4 can thus be denoted by:

r_1: $A, B \Rightarrow F \otimes (C;D;E)$; r_2: $G \Rightarrow J \otimes (H;I)$; r_3: $I \Rightarrow M \otimes K$; r_4: $J \Rightarrow M \otimes K$

2.2 Business Process Model

We provide a formal definition for a simple business process model. Through which the runtime behaviors of the process as reflected by execution sequences can be defined.

Definition 2 (Process Model). A process model W is a pair (N, E), which is defined through a directed graph consisting a finite set of nodes N, and a finite set of flow relations (edges) $E \subseteq N \times N$. Nodes are classified into tasks T and coordinators C, where $N = C \cup T$, and $C \cap T = \varnothing$. T is the set of tasks in W, and C contains coordinators of the type {*Begin*, *End*, *Fork*, *Synchronizer, Choice, Merge*}, which have typical workflow semantics. A sub-process model is a special type of W, which is a fragment of a process model in which {*Begin*, *End*} is excluded from its coordinator nodes.

Given a process model W and a task $T_i \in T$, *Trigger*(W, Ti) denotes the set of tasks that can be triggered by task T_i in W as the result of execution. E.g., *Trigger*(W, A) = {B} (*cf.* Fig. 1). For tasks followed by a *Fork* (*AND-SPLIT*) or a *Choice* (*XOR-SPLIT*) coordinator, we consider all subsequent tasks after the coordinator can be triggered. E.g., *Trigger*(W, B) = {C, D, F}, *Trigger*(W, G) = {H, J}. *Disable*(W, T_i) denotes the set of tasks disabled as the consequence of executing Ti, which is defined to realize the semantics of the *Choice* coordinator. For example, *Disable*(W, H) ={J}, which means either H or J is executed but not both. *Initial*(W) is a function returning the first task node in W.

An *execution sequence* of a process models referred to as the trace of execution in a process model, which reflects a possible order of task executions at runtime. Typically, a process model with parallel branches (*Fork*) or alternative branches (*Choice*) contains more than one possible execution sequences.

For example, for tasks A, B, C, D, E, and F in W (*cf.* Fig. 1), there are three possible execution sequences <A, B, F>, <A, B, C, D, E> and <A, B, D, C, E>, since F and C, D, E are in alternative branches, and C, D in parallel branches.

We follow the general sequence definition to define an execution sequence: A finite sequence $s = \{s_1, s_2, \ldots, s_n\}$ is a function with the domain $\{1, 2, \ldots, n\}$, for some positive integer n. The i-th element of s is denoted by s_i.

Definition 3 (Execution Sequence). An execution sequence s^W of a process model W is a finite sequence of tasks $T' \subseteq T$ in W, which is defined by the sequence $<T_1, T_2, \ldots, T_n>$, $n \geq 1$. An execution sequence ss^W is a *subsequence* of s^W if every element in ss^W is an element of s^W, and the elements in ss^W occur in the same order as in s^W.

2.3 Measurement of Compliance

It is desirable to transform the control objectives given in FCL into a form such that it is comparable to business process design. We establish the connection between FCL and business process model through execution sequences and the so called state of idealness [6]. Through the states of idealness we can determine whether a process model is compliant with the control objective (i.e., how well the process model supports such “ideal” states in execution).

Intuitively an *ideal* situation is a situation where execution sequences do not violate FCL expressions, and thus the execution sequences are fully compliant with the control rule. A *sub-optimal* situation is a situation where there are some violations, but these are repaired. Accordingly, processes resulting in *sub-optimal* situations are still compliant to a control rule even if they provide sub-optimal performance of the control objective. A situation is *non-ideal* (*non-compliant*) if it violates a control objective and the violations are not repaired.

There are two possible reasons for a process not to comply with a control rule: 1) the process executes some tasks which are prohibited by the control rule (or equivalently, it executes the opposite of obligatory tasks); 2) the process fails to execute some tasks required by the control rule. For example consider the rule

$$r: A \Rightarrow B \otimes C$$

which means that, if A occurred then it must be followed by B, or in alternative, in case B does no occur, it must be followed by C. An *ideal* state for r is the situation (a possible execution sequence) s_1 = <A, B>. A *sub-optimal* situation can be s_2 = <A, C> where the first obligation B is not fulfilled. Note that we also consider s_3 = <A, B, C> a *sub-optimal* situation since it is not required to perform C when B is already in place. The *non-ideal* situation is s_4 = <A>.

Definition 4 (Idealness of execution sequence). Let S^W be the set of all possible execution sequences of a process model W, r: $A_1, \ldots, A_m \Rightarrow B_1 \otimes \ldots \otimes B_n$ be a control objective in FCL.

- A sequence $s \in S^W$ is an *ideal* execution sequence to r *iff* sequence $<A_1, \ldots, A_m, B_1>$ is a subsequence of s.
- A sequence $s \in S^W$ is a *sub-optimal* execution sequence to r *iff* $\exists B_i$, $1< i \leq n$ such that $<A_1, \ldots, A_m, B_i>$ is a subsequence of s.
- A sequence $s \in S^W$ is a *non-ideal* execution sequence to r *iff* sequence $<A_1, \ldots, A_m>$ is a subsequence of s and s is neither *ideal* nor *sub-optimal.*

Given a control rule r, we denote the set of *ideal* and *sub-optimal* execution sequences as S^r_{ideal} and $S^r_{sub\text{-}optimal}$ respectively. Table 2 shows such for control rules $r_1 - r_4$. Note that for compliance checking purpose, *sub-optimal* execution sequences only contain the *consequences* of the control rule, *i.e.*, right hand side of r. Because the *antecedent*, *i.e.*, left hand side of r is irrelevant in *sub-optimal* states.

The above definition for *non-ideal* covers the second type of non-compliant situation where the process fails to execute some required tasks. We argue that the first situation where the process executes prohibited task(s) can be checked by simple sequence (string) matching technique (for execution sequences between control rule and process model) and hence not discussed further. In the next section, we discuss the technique to check for compliance degree for *ideal* and *sub-optimal* execution sequences.

Table 2. State of idealness of control rules $r_1 - r_4$

Control Rule	S^r_{ideal}	$S^r_{sub\text{-}optimal}$
r_1: $A, B \Rightarrow F \otimes (C;D;E)$	$<A, B, F>$	$<C, D, E>$, $<F, C, D, E>$, $<C, D, E, F>$
r_2: $G \Rightarrow J \otimes (H;I)$	$<G, J>$	$<H, I>$, $<J, H, I>$, $<H, I, J>$
r_3: $I \Rightarrow M \otimes K$	$<I, M>$	$<K>$, $<M, K>$, $<K, M>$
r_4: $J \Rightarrow M \otimes K$	$<J, M>$	$<K>$, $<M, K>$, $<K, M>$

3 Compliance Degree

We now have all the machinery to define the measure for compliance between a process model and a given control rule. We propose to use the notion of compliance degree as a quantitative measurement. The notion further utilizes the concept of *support*: Given a set of execution sequences S and a process model W, the *support* of W based on a sequence $s \in S$ is given by the proportion of tasks in s that can be executed in W. The range of support is a real number between 0 and 1, where 0 indicates no support (s is not executable in W at all) and 1 complete match (the entire sequence s can be executed in W, i.e., it is possible to derive an execution sequence s^W from W such that $s = s^W$). The *support* of W based on S is the weighted sum of *support* from all sequences in S, which is also between 0 and 1.

In order to calculate the *ideal* and *sub-optimal compliance degree*, we need to first extract the set of *ideal* and *sub-optimal* execution sequences for each control rule r, and calculate the degree of support for these sequences in the process model. The rationale of this technique is to measure how well a given process model W represents the *ideal* and *sub-optimal* situations in control rule r by calculating the support for W against the set of *ideal* and *sub-optimal* execution sequences representing r. We refer to the support for *ideal* and *sub-optimal* sequences as *ideal* and *sub-optimal compliance degree* respectively. The first measurement indicates whether the *ideal* situation (the exact sequence) can be fully or partially supported in W (*ideal compliance degree* = 1, or between [0, 1]) respectively). Similarly, the latter measurement indicates whether W allows *sub-optimal* situation(s) and by what degree.

We first extract a sub-process from the process model which contains only the relevant tasks as in the set of *ideal* and *sub-optimal* execution sequences of r. To achieve this we use a technique called `SELECTIVE_REDUCE` [8]. For example, the procurement process model W (*cf*. Fig. 1) is reduced into W_1, W_2, W_3 and W_4 (Fig. 2) against control rule r_1, r_2, r_3 and r_4 respectively.

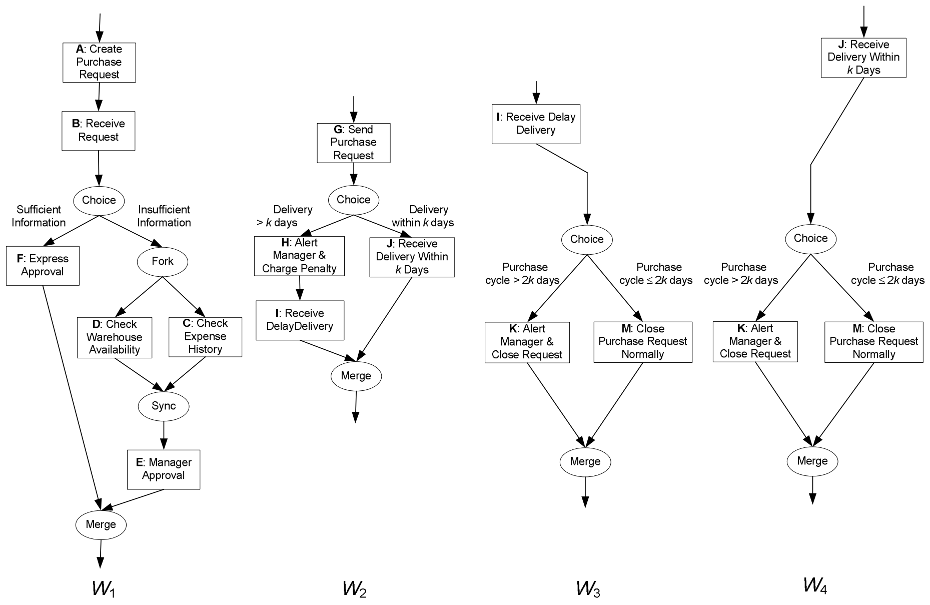

Fig. 2. Sub-processes of the procurement process

We then calculate the compliance degree through the algorithm given in Fig. 3. The algorithm takes as inputs a process model W, a set of sequences S, and the control rule r, produces the compliance degree *comp*. Functions *Trigger*, *Disable* and *Initial* given in Definition 2 are utilized. An additional function *SubInitial*(W, r) returns the set of task node(s) which are immediate after the last antecedent task in r. For example, *SubInitial*(W_2, r_2) = $\{H, J\}$, where G is the last task in the antecedent of r_2. Function *Sub-optimal*(S^W) returns TRUE if is S^W the set of *sub-optimal* sequences.

For each sequence s in S, Tr is initially given the first task in W in step 4. For each task T_i in a sequence s (in this case, $T_i = s_i$ where s_i is the i-th element in s), Tr is the current set of triggered tasks as the result of executing task T_i in W. Step 8 checks whether the triggered tasks in Tr includes T_i. Step 11 calculates the proportion of tasks in *W* triggered by tasks in *s*. After all different sequences in *S* have been accounted for, the final compliance degree is scaled according to the total number of sequences in *S* and returned (step 12). The algorithm complexity is bound by the number of tasks in the sequence and the number of different sequences in *S*.

For example, to compute the *ideal compliance degree* of *W* with regard to r_1: $A, B \Rightarrow F \otimes (C;D;E)$, we input W_1, the sub-process of *W* relevant to r_1 (*cf.* Fig. 2), and S^{r1}_{ideal}, the set of *ideal* execution sequences of r_1, where $S^{r1}_{ideal} = \{<A, B, F>\}$. Since there is only one sequence in S^{r1}_{ideal}, the *ideal compliance degree* is (1+ 1+1)/3 = 1 (step 11), because <*A*, *B*, *F*> is an exact execution sequence executable in W_1.

```
Procedure. COMPLIANCE_DEGREE
Input W, S, r
Output degree
1. degree, count, comp ← 0
2. For each different sequence s in $S^W$
3.     If Sub-optimal($S^W$)                     // for sub-optimal compliance degree
4.         Tr ← SubInitial(W, r)
5      Else                                       // for ideal compliance degree
6.         Tr ← Initial(W)
7.     For each task in s denoted by $T_i$, $i$ ← 1, …, |s|
8.         If $T_i \in Tr$
9.             count = count + 1
10.        $Tr \leftarrow (Tr - \{T_i\} - Disable(W, T_i)) \cup Trigger(W, T_i)$
11.    $comp \leftarrow comp + \frac{count}{|s|}$
12. Return $degree \leftarrow \frac{comp}{|S|}$
```

Fig. 3. An algorithm to compute compliance degree

The *sub-optimal compliance degree* of *W* with regard to r_1 can also be computed. We again input W_1 and $S^{r1}_{sub\text{-}optimal}$, the set of *sub-optimal* execution sequences of r_1, where $S^{r1}_{sub\text{-}optimal} = \{<C, D, E>, <F, C, D, E>, <C, D, E, F>\}$. For each sequence *s* in $S^{r1}_{sub\text{-}optimal}$, we display in Table 3 the intermediate result of *degree*, which is the support of W_1 received from *s*. Sequence <*C*, *D*, *E*> has *degree* of 1 since it is an exact sequence executable in W_1. Sequence <*F*, *C*, *D*, *E* > has *degree* of 0.25 because after triggering *F* in W_1, C, D, and E will be disabled ((1+0+0+0)/4 = 0.25 in step 9). Similarly, sequence <*C*, *D*, *E*, *F*> has *degree* of 0.75 since after triggering *C*, *D*, and *E* in W_1, *F* will not be triggered ((1+1+1+0)/4 = 0.75). The *sub-optimal compliance degree* is 0.67, which is the average of the three *degrees*.

Suppose there is a process *W'* containing a subgraph (subprocess) W'_1 relevant to r_1, where tasks D and E are not included (*cf.* Fig. 4). In this case the there is no ideal situation in *W* since the *ideal compliance degree* is (1+1+0)/3 = 0.67 ≠ 1. The *sub-optimal compliance degree* is also reduced to 0.5.

We use the *ideal compliance degree* to evaluate how well the process model supports a given control rule. *degree* = 1 indicates all ideal situation(s) of the control objective are represented in the process model *W*, (i.e., it is possible to find out the

Table 3. Intermediate result for applying COMPLIANCE_DEGREE to $S^{r1}_{sub\text{-}optimal}$ and W_1

$S^{r1}_{sub\text{-}optimal}$	*degree*
$< C, D, E >$	1
$< F, C, D, E >$	0.25
$< C, D, E, F >$	0.75
sub-optimal compliance degree	**0.67**

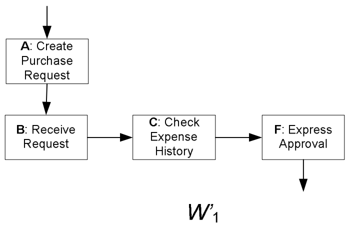

Fig. 4. Sub-process relevant to r_1 of an alternative procurement process

exact ideal execution sequence(s) in the relevant sub-graph of *W*, hence the process is an *ideal design* for the control rule *r*). While 0 indicates none of the ideal situation(s) is represented in *W*, from which we can immediately conclude that *W* is *non-compliant* with *r*. If none of the task in any sequence of *ideal* or *sub-optimal* execution sequences S^{r}_{ideal} is presented in the process model *W*, then one can only derive an empty sub-graph from *W* which contains the relevant tasks in S^{r}_{ideal}, Thus the algorithm returns 0 in this case, which is corresponding to a *non-compliant* situation. Lastly, having a number between 0 and 1 indicates *W* represents part of some *ideal* situation (i.e., it is not possible to find out exact but partial ideal execution sequence(s) in the relevant sub-graph of *W*).

In addition, from the *sub-optimal compliance degree* we can find out whether the process model may contain some *sub-optimal* situations. There can be many interpretations for *sub-optimal compliance degree*. Here we consider it as an auxiliary measurement to examine the expressiveness of the process model, in terms of expressing both *ideal* and *sub-optimal* executions. For example, in the case when two arbitrary process models W_α and W_β are both *ideal* to a control rule, but W_α has a higher *sub-optimal compliance degree* of W_β, then W_α is a better design.

Table 4 lists the *ideal* and *sub-optimal* compliance degree for control rules r_1 - r_4 respectively. The overall compliance degree is the sum of the compliance degree of each control rule. Note that we can also take a weighted approach for calculating the *sub-optimal compliance degree*. For each control rule *r*, a weight can be assigned to

Table 4. Compliance measurement for process model *W*

Control Rules	*Ideal Compliance Degree*	Risk (Weight)	*Sub-optimal Compliance Degree*
r_1	1	10%	0.67
r_2	1	50%	0.67
r_3	1	20%	1
r_4	1	20%	1
TOTAL	**1**	100%	**0.80**

reflect the relative importance of compliance with respect to *r*. Weights are assumed to be determined by experts defining internal controls, as an indication of the risk (or cost) of non-compliance. The overall *sub-optimal compliance degree* for *W* undertakes such approach. The results show that *W* is compliant with all ideal situations according to control rules r_1 - r_4, and *W* supports *sub-optimal* situations to a large extend.

4 Conclusion and Future Work

This paper presents an overall methodology for compliance by design, and specifically proposes a method to measure the degree of compliance between control objectives and business process models during process design. The proposed method based on the notion of compliance degree will assist process designers in undertaking compliance aware design so that an appropriate balance between the two, often conflicting, objectives can be achieved.

The approach presented so far is focused on assessing compliance of a process model through execution sequences. However, control objectives may also refer to other aspects of the process such as resource allocations, or data flow. Consideration of these aspects is part of our future work through which we hope to extend the proposed notion of compliance degree.

References

1. zur Muehlen, M., Ho, D.T.: Risk Management in the BPM Lifecycle. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 454–466. Springer, Heidelberg (2006)
2. Christopher, G., Müller, S., Pfitzmann, B.: From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation. IBM Research Report RZ 3662, IBM Zurich Research Laboratory (2006)
3. Goedertier, S., Vanthienen, J.: Designing Compliant Business Processes with Obligations and Permission. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
4. Governatori, G.: Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems 14(2–3), 181–216 (2005)
5. Governatori, G., Milosevic, Z.: A Formal Analysis of a Business Contract Language. International Journal of Cooperative Information Systems 15(4), 659–685 (2006)
6. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: Proceedings of the 10th IEEE Conference on Enterprise Distributed Object Computing (2006)
7. Hagerty, J.: SOX Spending for 2006. AMR Research, Boston USA (2007)
8. Lu, R., Sadiq, S.: Managing Process Variants as an Information Resource. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, Springer, Heidelberg (2006)
9. Sadiq, S., Governatori, G., Naimiri, K.: Modeling Control Objectives for Business Process Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, Springer, Heidelberg (2007)
10. Zdravkovic, J., Kabilan, V.: Enabling Business Process Interoperability Using Contract Workflow Models. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 77–93. Springer, Heidelberg (2005)