Ensuring Compliance of Distributed and Collaborative Workflows

David Knuplesch and Manfred Reichert and Rüdiger Pryss
Institute of Databases and Information Systems
Ulm University, Germany
{david.knuplesch,manfred.reichert,ruediger.pryss}@uni-ulm.de

Walid Fdhila and Stefanie Rinderle-Ma
Faculty of Computer Science
University of Vienna, Austria
{walid.fdhila,stefanie.rinderle-ma}@univie.ac.at

***Abstract*—Automated workflows must comply with domain-specific regulations, standards and rules. So far, compliance issues have been mainly addressed in the context of intra-organizational workflows. In turn, there exists only little work dealing with compliance of distributed and collaborative workflows. As opposed to intra-organizational workflows, for distributed and collaborative workflows compliance must be addressed at different levels. This includes local compliance rules of a particular partner as well as global compliance rules to be obeyed by multiple partners collaborating in the distributed workflow. As a particular challenge, the private elements of a particular partner workflow are hidden to the partners and hence not known by them. Accordingly, only limited information is available when checking compliance of distributed and collaborative workflows. This paper introduces techniques enabling compliance checking for distributed and collaborative workflows, taking these privacy constraints into account. Hence it enables ensuring compliance of distributed and collaborative workflows at design time.**

***Keywords*—business process compliance, collaborative and distributed workflows, privacy**

I. Introduction

Business process compliance has been identified as a core challenge for process-aware information systems [1]. So far, its focus has been on intra-organizational workflows, and a variety of approaches for checking the compliance of a workflow with domain-specific regulations and rules in different phases of the process life cycle have been proposed [2]–[6]. Except for few approaches (e.g., Contracts [7], [8]), compliance checking for distributed and collaborative workflows has been neglected so far, even though being crucial for any collaborative setting [9], [10]. Hence, the incorporation of compliance rules into distributed and collaborative workflows and the provision of techniques for checking them are indispensable for enterprise computing. Compared to approaches for checking compliance of intra-organizational workflows, two additional challenges emerge [9]: *First*, business process compliance must be ensured at different levels, including local compliance rules of a particular partner as well as global compliance rules to be obeyed all partners of the distributed and collaborative workflow. *Second*, compliance checking must cope with the fact that distributed and collaborative workflows comprise private elements of particular partners that are globally not known.

Consider Fig. 1: First of all, compliance rules relevant for setting up and running distributed and collaborative workflows may refer to different levels. Usually, each partner maintains its own *private process model* that does not only contain activities for exchanging messages with the other partners, but also comprise private activities not relevant for the interactions with the partners, i.e., due to privacy reasons, a partner will usually not reveal all details about its private processes to the other partners. Semantic constraints on such private processes are denoted as *local compliance rules*. In turn, the *public process model* of a partner, which constitutes a restricted view on its private process model, is visible to all partners. Furthermore, *assertions* can be used to augment public process models. In particular, assertions enrich a public process model with additional information about its execution behavior not observable from the public process itself [9], e.g., to assure partners that components produced are tested before shipping. In practice, assertions are contained, for example, in SLAs, certified standards, or business contracts. In turn, an *interaction model* defines the sequence of interactions (i.e., messages exchanged) between the partners of a distributed and collaborative workflow. A *local view* on such an interaction model links the public process model of a particular partner with an interaction model; hence, it reflects the interactions of this partner with the others. In this context, *global compliance rules* are imposed at the interaction level, e.g., all components of a car may have to be tested before integrating them [11]. Altogether, the *public elements* of a distributed and collaborative workflow include the interaction model, local views, and public process models as well as the assertions and global compliance rules. In turn, private process models and local compliance rules constitute *private elements*.As opposed to intra-organizational processes, in the context of a distributed and collaborative workflow, we must consider three levels of compliance: First, we must deal with local compliance rules that constrain private partner processes. Second, we must consider assertions that refine public process models. Third, we must support global compliance rules that constrain the public parts of a distributed and collaborative workflow. Overall, this leads to the following research questions:

RQ 1 How to define business process compliance for the different levels of a distributed and collaborative workflow?

RQ 2 How to design a distributed and collaborative workflow that complies with local and global compliance rules?

RQ 3 How to check compliance of a distributed and collaborative workflow; i.e., how to ensure that all producible traces of the distributed and collaborative workflow will meet the imposed compliance rules?

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254095

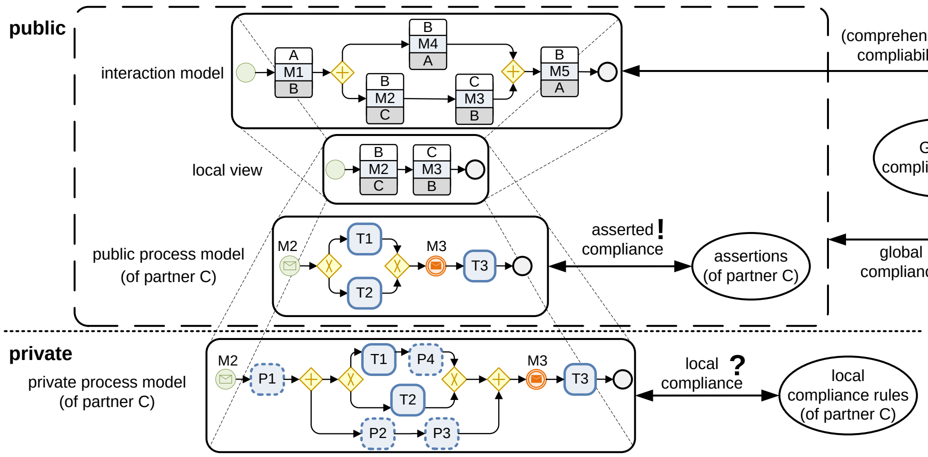

Figure 1: Different levels of compliance rules in distributed and collaborative workflows

Regarding RQ 1, this paper formally defines compliance for the different levels of distributed and collaborative workflows, i.e., local compliance rules, assertions, and global compliance rules. RQ 2 is addressed by following a top-down approach for defining compliant distributed and collaborative workflows. This approach comprises steps to ensure compliance on the aforementioned levels (i.e. RQ 3). In particular, we provide algorithms to automatically check different kinds of *compliability* of interaction models, i.e., the ability of an interaction model to not conflict with global compliance rules. Further, we provide algorithms that automatically check whether the public elements of a distributed and collaborative workflow ensure *global compliance* with global compliance rules. To demonstrate the feasibility of our approach, a proof-of-concept prototype enabling compliability and compliance checking for distributed and collaborative workflows was developed.

Note that we already informally introduced the notion of compliability and algorithms for compliability checking in [12]. This paper significantly extends [12] by adding formal definitions of compliability, comprehensive compliability, and global compliance and formally proofing the significance of these criteria. Other contributions include the algorithms for automatically checking comprehensive compliability and global compliance.

Figs. 2 and 3 depict an example of a distributed and collaborative workflow from the healthcare domain using BPMN 2.0. This process involves three *partners*: `gynecologist`, `laboratory`, and `hospital`. In particular, Fig. 2 depicts the interactions (i.e., messages exchanged) between these partners. In turn, Fig. 3 shows their public process models. Note that the public process model of the hospital is simplified due to lack of space.

The remainder of this paper is structured as follows: Sect. II discusses related work. Sect. III introduces our approach for defining compliant, distributed, and collaborative workflows. Sect. IV comprises our formal framework. In particular, Sect. IV-C provides formal definitions regarding the compliability of interaction models as well as global compliance of distributed and collaborative workflows. Finally, Sect. V provides algorithms for compliability and compliance checking as the main contribution of this paper. Sect. VI closes with a conclusion and outlook.

II. Related Work

In many domains, workflow execution is subject to compliance rules that stem from laws, regulations, and guidelines (e.g. Basel or Sarbanes-Oxley-Act) [1]. Approaches, methods, and techniques for ensuring the compliance of a workflow with respective rules and constraints are covered under the term *business process compliance* [4].

Existing approaches enabling the specification of compliance rules differ with respect to the fundamental formalism used and the process perspectives covered. While early approaches have focused on the control flow perspective, recent works [6], [13], [14] highlight the need to ensure compliance for other process perspectives (e.g., data, resource and time) as well. In the context of distributed and collaborative workflows, the interactions between business partners (i.e. messages exchanged) constitute another crucial perspective, whose compliance must be guaranteed [9], [12].

There exist approaches formalizing compliance rules for the control flow perspective based on temporal logic (e.g., LTL [15], CTL [16]). Since logic-based notions are not easy to comprehend, [17] suggests a pattern-based approach to encapsulate logic. While the latter approach only considers the control flow perspective, [18], [19] introduce compliance patterns that address the data, resource, and time perspectives as well. Furthermore, visual approaches exist that allow modeling of compliance rules referring to control flow [20]–[22]. In addition, [23], [24] support the data perspective. Finally, [25] allows modeling compliance rules with respect to all process perspectives, i.e. control flow, data, time, resource, and interactions between business partners.

Besides the formal specification of compliance rules, their integration along the entire process life cycle has been discussed in literature [3], [4], [6], [26].

Whether or not a workflow satisfies imposed compliance rules has to be ensured for all phases of the process life cycle [27]:

To check the compliance of process models against pre-specified rules at design time, many approaches apply *model checking* [15], [16], [20], [23], [28]. Besides control flow, some of them even address the data and time perspectives. Other approaches rely on the notion of *semantic congruence* [29] or use *Petri Nets* [30]. In turn, [31] applies Mixed-Integer Programming to verify the compliance of process models as well as the validity of change operations. In this context, notions like *degree of compliance*, *validity of change operations*, and *compliance by compensation* are introduced.

Runtime checking and monitoring of business process compliance (e.g., continous auditing [32]) are addressed by several approaches: [33] enriches process models with controls. Another compliance monitoring framework, which is based on common event standards and middleware, is presented in [34]. In turn, [7] discusses the monitoring and enforcement of compliance in the context of distributed and collaborative workflows. More recently, *Compliance Rule Graphs* [35] and colored automata [5] have been used to enable fine-grain compliance diagnostics during runtime.

To complement design time and runtime compliance checking, backward compliance checking of workflow logs has been proposed by [36]. [18] uses alignments to detect compliance violations in workflow logs.

Finally, declarative approaches [37]–[39] ensure compliance in a smart manner. Since workflows are defined in terms of a set of declarative rules, imposed compliance rules only need to be added to the workflow to ensure business process compliance based on ”in-build” techniques.

So far, only little work exists addressing compliance issues in the context of distributed and collaborative workflows [7], [8]. In particular, compliance with respect to distributed and collaborative workflows, taking also privacy constraints into account, has not been sufficiently investigated yet [9]. To remedy this drawback, we started the C^{3}Pro research project and introduced the notion of *compliability* in its context [12].

Various issues related to the modeling of *distributed and collaborative* workflows have been discussed in literature. In particular, there exist industry standards like BPEL4WS, WS-CDL, and RosettaNet as well as powerful modeling frameworks and notations (e.g. ”Let’s dance” [40], iBPMN [41], and BPMN 2.0). Furthermore, *interaction patterns* were suggested in [42], describing practically relevant message exchanges between partners. Finally, [43], [44] discuss how partners may publish restricted public process models, i.e. views on their private process models, to preserve privacy.

Several Top-down-approaches, which either start from an interaction model or a set of public process models, allow determining whether private process models are compatible with the public ones [45]–[48]. Finally, [41] and [49]–[51] discuss *realizability* of interaction models, i.e., whether the partners involved will be able to specify public and private process models being compatible with the interaction model.

III. Compliance-Awareness In Distributed and Collaborative Workflows

This section presents our methodology for ensuring compliance of distributed and collaborative workflows. It comprises 11 steps as outlined in Fig. 4. Furthermore, our methodology follows the top-down paradigmas known from interaction modeling [41], but adds further steps to deal with compliance.

First, imposed regulations (e.g. standards, guidelines, laws) must be selected in Step 1. This corresponds to the definition of a set of *global compliance rules*. We sketch this step in Sect. IV-C. In Step 2, the global view of the interactions (i.e.

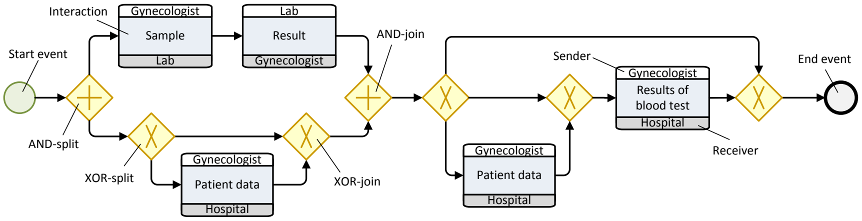

Figure 2: Example: Interaction model of healthcare scenario from [9]

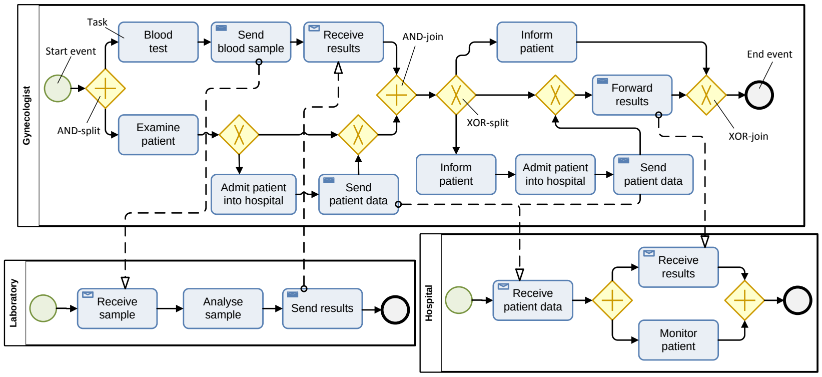

Figure 3: Example: public process models of healthcare scenario from [9]

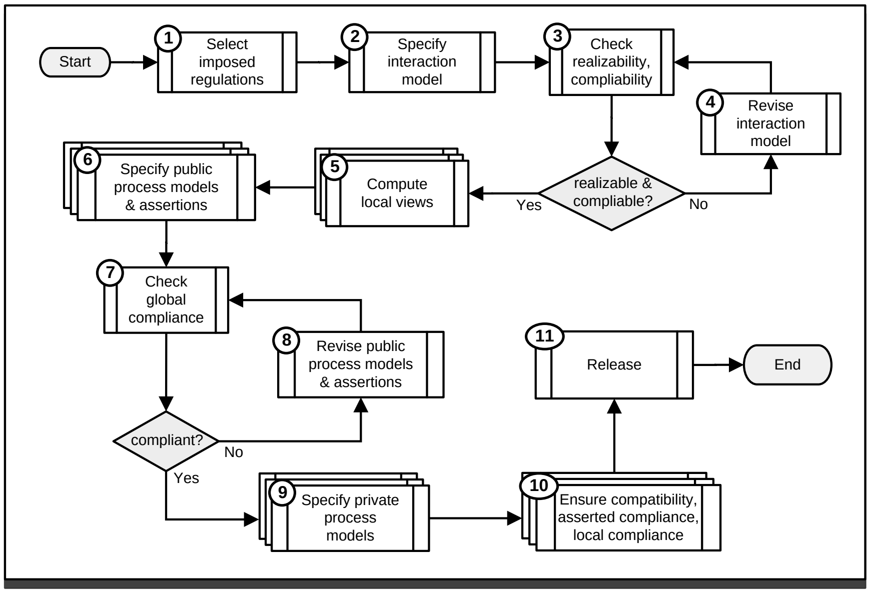
Start
1
Select imposed regulations
2
Specify interaction model
3
Check realizability, compliability
4
Revise interaction model
realizable & compliable?
Yes
No
5
Compute local views
6
Specify public process models & assertions
7
Check global compliance
8
Revise public process models & assertions
11
Release
End
compliant?
No
Yes
9
Specify private process models
10
Ensure compatibility, asserted compliance, local compliance

Figure 4: Creating compliant distributed and collaborative workflows

messages exchanged) between the partners of a distributed and collaborative workflow is specified (i.e., the *interaction model*).

Steps 3 and 4 ensure *correctness of the interaction model*. On one hand, this includes common *behavioral correctness criteria*, e.g. absence of deadlocks and lifelocks [52], [53]. On the other, interaction modeling additionally requires ensuring realizability of an interaction model. The latter indicates, whether every involved partner is able to model a process compatible with the interaction model; [41], [49], [51], and [50] address realizability and its verification in detail.

In our context, partners must be able to model public and private process models, which comply with the given interaction model and meet the global compliance rules defined in Step 1. In particular, this requires interaction models to not conflict with the set of global compliance rules. This constitutes the semantic correctness criterion *compliability* [12]. In this paper, we further introduce the notion of *comprehensive compliability*. Sect. IV-C provides formal definitions of both compliability notions, while Sect. V presents algorithms for checking compliability of interaction models.

After ensuring behavioral correctness, realizability, and (comprehensive) compliability, Step 5 computes the local view of each partner [54]. In particular, a local view covers the behavior of the interaction model from the viewpoint of a particular partner. Based on the obtained local views, the partners define public process models of their processes in Step 6 by adding public tasks. Additionally, partners may refine their public process models by adding assertions in order to ensure particular properties of their private process models (created in Step 9). In Step 7, we check global compliance, i.e., we check whether the interaction model, the public process models, and the corresponding assertions together meet the global compliance rules defined in Step 1. In particular, we consider Step 7 to be the most challenging one in this context. Note that the private processes of the partners are not known and hence common techniques for compliance checking (e.g. [2], [15], [55]) cannot be directly applied. Sect. IV-C formally defines *global compliance*, while Sect. V presents algorithms for global compliance checking. If global compliance cannot be ensured, the public process models as well as their assertions must be revised in Step 8 before re-checking global compliance of the public parts.

After ensuring global compliance, the required public parts (i.e. interaction model, public process models, and assertions) of the distributed and collaborative workflow are complete. Hence the subsequent steps are performed privately by each partner. In particular, the partners specify their private process models in Step 9, which must be compatible with the corresponding public ones (cf. Step 6). Further, they must comply with the assertions defined in Step 6, as well as local compliance rules. This must be checked by each partner separately due to the privacy of his private process models in Step 10. In this context, [45], [47], [48] describe how compatibility can be verified. In turn, compliance with assertions and local compliance rules can be ensured through the application of common compliance checking techniques (e.g. [2], [15], [55]). Finally, the distributed and collaborative workflow is released in Step 11.

IV. FORMAL FRAMEWORK

A. *Process Models and Interaction Models*

As emphasized, we aim at distributed and collaborative workflows that comply with domain-specific regulations and rules. Thereby, we distinguish between different, but overlapping viewpoints [54]: private process models, public process models (i.e., public views on private process models), local views (on the interaction model), and the interaction model itself.

A *private process model* (also denoted as orchestration or local process model) describes the internal business logic of a partner and defines the execution constraints for its activities. Thereby, *private process models* may contain activities as well as internal knowledge, a partner does not want make public to other partners. For this purpose, partners provide restricted *public process models* that abstract from *private activities* and internal details of their private processes. The *local view* on the interaction model — also denoted as *behavioral interface* — then defines the message exchanges from the perspective of a single partner as well as the sequencing of these messages. Finally, the *interaction model* provides a global view on the interactions and messages exchanged in a distributed and collaborative workflow.

Def. 1 summarizes the different notions introduced above. Since our approach is independent from a particular process modeling formalism, we abstract from language-specific details, i.e., we assume that process and interaction models are characterized in terms of a set of producible execution traces.

Definition 1 (Process Models and Views):
In a distributed and collaborative workflow, let $\mathcal{P}$ be the set of involved partners and $p \in \mathcal{P}$ be an arbitrary partner. Then:

- $\mathcal{T}$ is the set of all tasks (i.e., activities) and $T_p \subseteq \mathcal{T}$ is the set of the tasks performed by p. T_p is partitioned into the set of public tasks V_p and the set of private tasks S_p; i.e. $T_p = V_p \dot{\cup} S_p$.
- $\mathcal{M}$ corresponds to the set of all interactions, and $M_p \subseteq \mathcal{M}$ to the set of all interactions, in which p is involved, i.e., messages sent or received. For a particular interaction $I(m,s,r)$, a message m is sent from $s \in \mathcal{P}$ to $r \in \mathcal{P}$. $SEND(I(m,s,r))$ ($REC(I(m,s,r))$) denotes the sending (receiving) task of $I(m,s,r)$.
- π_p is the private process model of p. It comprises tasks of T_p and interactions of M_p. Π corresponds to the set of all private process models.
- γ_p is the public process model of p. It comprises public tasks of V_p and interactions of M_p. Γ corresponds to the set of all public process models.
- l_p is the local view of p on the interaction model. It sends and receives elements of M_p. $\mathcal{L}$ is the set of all local views.
- $\mathcal{G}$ is the global interaction model that consists of interactions of $\mathcal{M}$.
- $\mathfrak{M} := \Pi \cup \Gamma \cup \mathcal{L} \cup \{\mathcal{G}\}$ is the set of all models of a distributed and collaborative setting.

Based on Def. 1, Def. 2 introduces traces on interaction models as well as public and private process models.

Definition 2 (Traces):
Let $m \in \mathfrak{M}$ be a model of a distributed and collaborative workflow and $M \subseteq \mathfrak{M}$ be a subset of models. Further, let $t \in \mathcal{T} \cup \mathcal{M}$ be a task or interaction and $E \subseteq \mathcal{T} \cup \mathcal{M}$ be a subset of the latter. Finally, let $\mathfrak{T} \subseteq ((\mathcal{T} \times \mathcal{P}) \cup \mathcal{M})^\star$ be a set of *traces* that consist of tasks and messages. Then:

- $t \in^+ m$ expresses that t is used in m. Further, $m^\in := \{t \in \mathcal{T} \cup \mathcal{M} | t \in^+ m\}$ denotes the set of tasks and interactions used by m,
- $\mathfrak{T}(m) \in ((\mathcal{T} \times \mathcal{P}) \cup \mathcal{M})^\star$ is the set of traces producible on m. Each trace entry contains either a message exchanged or a task performed by a particular partner,
- $\mathfrak{T}^\star(M) := \{\tau \in (\bigcup_{m \in M} m^\in)^\star | \forall m \in \mathcal{M} : \Phi_{m^\in}(\tau) \in \mathfrak{T}(m)\}$ is the set of traces producible on the composition of the models in M. Thereby $\Phi_E(\tau)$ restricts a trace τ to entries that are elements of E,
- $[\mathfrak{T}]^E \in ((\mathcal{T} \times \mathcal{P}) \cup \mathcal{M})^\star$ corresponds to the set of traces obtained by arbitrarily supplementing traces with elements (i.e. tasks and messages) of E.

Following Def. 2, $\mathfrak{G} := \mathfrak{T}^\star(\{\mathcal{G}\} \cup \Pi)$ describes the set of traces producible by a distributed and collaborative workflow. Due to privacy constraints of the partners, however, Π might be unknown and, thus, $\mathfrak{G}$ might never be obtained exactly when checking for compliability and compliance. In turn, this highlights the need for available approximations of the set of traces producible by a distributed and collaborative workflow. For this purpose, we provide two approximations of $\mathfrak{G}$:

1) The first one is solely based on knowledge about an interaction model $\mathcal{G}$ and the set of tasks T_p of each partner p. $\mathfrak{A}_{\mathcal{G}} := [\mathfrak{T}(\mathcal{G})]^{\bigcup_{p \in \mathcal{P}} T_p}$ is the set of traces that conform with interaction model $\mathcal{G}$, but may contain additional tasks.
2) Our second approximation further includes knowledge about the public process models Γ and the set of private tasks S_p of each partner p. In this context, $\mathfrak{A}_{\mathcal{G},\Gamma} := [\mathfrak{T}^\star(\{\mathcal{G}\} \cup \Gamma)]^{\bigcup_{p \in \mathcal{P}} S_p}$ is the set of traces that conform with interaction model $\mathcal{G}$ as well as the public process models Γ, but may additionally contain private tasks.

If compatibility between $\mathcal{G}$, γ_p and π_p (cf. Def 1) can be ensured, $\mathfrak{A}_{\mathcal{G}} \supseteq \mathfrak{A}_{\mathcal{G},\Gamma} \supseteq \mathfrak{G}$ holds. Furthermore, we consider the approximation $\mathfrak{A}_p := [\mathfrak{T}(\gamma_p)]^{S_p} \supseteq \mathfrak{T}(\pi_p)$ as the set of traces conforming with the public process model π_p of a partner p that may additionally contain private tasks of p.

B. Compliance Rules

In general, *business process compliance* means that a workflow does not violate compliance rules. As aforementioned, for distributed and collaborative workflows, three different levels of compliance need to be distinguished: local compliance, asserted compliance, and global compliance. First, *local compliance* means that the private process of a partner does not violate any of its *local compliance rules*. Note that this corresponds to the common definition of compliance known from intra-organizational workflows. Second, *asserted compliance* means that a particular partner assures to the other partners that its private processes comply with a particular set of compliance rules, i.e. its set of *assertions*. Consequently, each assertion corresponds to a local compliance rule with respect to the private process model of this particular partner. Finally, *global compliance* means that a distributed and collaborative workflow does not violate any *global compliance rule*.

Definition 3 (Levels of Compliance Rules):
In the following, we consider GR as the set of global compliance rules. For any partner $p \in \mathcal{P}$ we consider

- LR_p as the set of local compliance rules of partner p,
- $A_p \subseteq LR_p$ as the set of (public) assertions of partner p, and $A := \bigcup_{p \in \mathcal{P}} A_p$ as the set of all assertions of a distributed and collaborative workflow.
- $\beta(r)$ as corresponding linear temporal logic (LTL) formula for each local compliance rule, assertion, or global compliance rule r.

We omit a detailed specification of β. Note that there exists work that comprehensively models compliance rules and transforms them into LTL expressions [2], [17].

Tab. I shows examples of compliance rules that affect the distributed and collaborative healtcare workflow from Figs. 2 and 3. Furthermore, these rules are classified according to Def. 3 and mapped onto LTL expressions in Tab. II.

C. Compliance and Compliability

Based on the definitions provided for compliance rules and assertions, we are now able to describe their formal semantics. In the context of a distributed and collaborative workflows, different levels must be considered, for which compliance shall be checked. In particular, we define local compliance of a private process from the viewpoint of a particular partner as well as compliability and global compliance taking a global perspective (cf. Def. 4).

Our definition of local compliance meets the common definition of *process compliance* and hence can be omitted here (for details see [2], [15], [55]).

In turn, our definition of *compliability* is based on [12], which describes compliability as the absence of conflicts between an interaction model and the set of imposed global compliance rules. Thus, compliability of an interaction model $\mathcal{G}$ can be defined as existence of a trace τ, which meets all global compliance rules and conforms with the message exchange defined in $\mathcal{G}$. Besides, τ may contain arbitrary tasks (i.e., $\tau \in \mathfrak{A}_{\mathcal{G}}$). Note that compliability is a necessary, but not a sufficient condition for the ability of the partners to define public and private process models in such a way that the resulting distributed and collaborative workflow is compliant (cf. Theorem 1).

However, there might still exist an interaction i of a compliable interaction model $\mathcal{G}$ violating global compliance rules; i.e., any trace of $\mathfrak{T}(\mathcal{G})$ containing i violates at least one global compliance rule. This would force public and private process models of the partners to completely ignore i as well. To cope with this, we introduce the notion of *comprehensive compliability* for interaction models. It requires that no interaction of an interaction model $\mathcal{G}$ must be ignored by the public and private partner processes to ensure global compliance. In turn, for each interaction i of $\mathcal{G}$, comprehensive compliability requires the existence of a trace $\tau \in \mathfrak{A}_{\mathcal{G}}$ containing i and meeting all global compliance rules.

We want to ensure global compliance taking a global perspective. Thus, one may assume that interaction model $\mathcal{G}$, public process models Γ, and assertions A are known, but not the private process models Π themselves. This implies that the set of producible traces $\mathfrak{G}$ of the distributed and collaborative workflow cannot be exactly obtained. Instead, we may use the approximation $\mathfrak{A}_{\mathcal{G},\Gamma}$. The latter can be further enhanced taking assertions into account, i.e. only traces complying with all assertions are considered. Nevertheless, this definition of global compliance ensures compliance of each each trace producible on the entire distributed and collaborative workflow (cf. Theorem 2).

Altogether, we formally define compliance and compliability as follows:

Definition 4 (Compliance and Compliability):
Let $\mathcal{G}$ be an interaction model with partner set $\mathcal{P}$ and Γ be the set of their public process models. Let further $p \in \mathcal{P}$ be a partner, and $\pi_p \in \Pi$ be its private process model, and $\gamma_p \in \Gamma$ be its public process model. Finally, let $lr \in LR_p$ be a local compliance rule of p, and $gr \in GR_p$ a global compliance rule, and A the set of all assertions. Then:

TABLE I: Examples of compliance rules

	Rule
r_1	After a blood test, the blood sample has to be destroyed.
r_2	The hospital shall only receive patient data if the gynecologist has admitted the patient to hospital before
r_3	After the gynecologist has admitted the patient to hospital, she has to be observed in the hospital.
r_4	Laboratory ensures to destroy the blood sample after analysis.

TABLE II: Compliance rules in LTL and their classification

	$\beta(\mathbf{r})$
$r_1 \in GR$	$\mathbf{G}$(Task= ’blood test’ $\Rightarrow$ $\mathbf{F}$ Task=’destroy sample’))
$r_2 \in GR$	(¬Task=’REC(patient data, Gyn)’ ∧ Performer=’Hosp’) $\mathbf{U}$(Task=’admit patient to hosp’ ∧ Performer=’Gyn’)
$r_3 \in GR$	$\mathbf{G}$(Task=’admit patient to hosp’ $\Rightarrow$ $\mathbf{F}$ (Task=’observe patient’ ∧ Performer=’Hosp’))
$r_4 \in A_{\text{Lab}}$ and LR_{Lab}	$\mathbf{G}$((Task=’start analysis’ ∧ Performer=Lab) $\Rightarrow$ $\mathbf{F}$ (Task=’destroy sample’ ∧ Performer=Lab))

- $\pi_p \vDash lr :\Leftrightarrow \forall \tau \in \mathfrak{T}(\pi_p) : \tau \vDash \beta(lr)$. This denotes that π_p complies with lr.
- $\mathcal{G} \vdash GR :\Leftrightarrow \exists \tau \in \mathfrak{A}_{\mathcal{G}} : \tau \vDash \bigwedge_{gr \in GR} \beta(gr)$. This denotes that $\mathcal{G}$ is compliable with the set of global compliance rules GR.
- $\mathcal{G} \Vdash GR :\Leftrightarrow \forall I(m,s,r) \in^{+} \mathcal{G} : \exists \tau \in \mathfrak{A}_{\mathcal{G}} : \tau \vDash \beta(gr)$ and $I(m,s,r)$ occurs in τ. This denotes that $\mathcal{G}$ is comprehensively compliable with the set of global compliance rules GR.
- $\mathcal{G}, \Gamma \vDash gr :\Leftrightarrow \forall \tau \in \mathfrak{A}_{\mathcal{G},\Gamma} : \tau \vDash \bigwedge\limits_{a \in A} \beta(a) \Rightarrow \beta(gr)$. This denotes $\mathcal{G}, \Gamma$ globally complies with the global compliance rule gr.

Theorem 1 (Necessarity of Compliability): Compliability of the interaction model is a necessary condition for global compliance; i.e, if compliability is violated, global compliance can not be ensured.

Proof 1 (Necessarity of Compliability):
Since $\mathfrak{A}_{\mathcal{G}} \supseteq \mathfrak{A}_{\mathcal{G},\Gamma} \supseteq \mathfrak{G}$ holds, we may directly conclude:
$\mathcal{G} \nvdash GR \Rightarrow \nexists \tau \in \mathfrak{G} \subseteq \mathfrak{A}_{\mathcal{G}} : \tau \vDash \bigwedge_{gr \in GR} \beta(gr)$,
$\Rightarrow \forall \tau \in \mathfrak{G} : \tau \nvdash \bigwedge_{gr \in GR} \beta(gr)$,
$\Rightarrow \forall \tau \in \mathfrak{G} : \exists gr \in GR : \tau \nvDash \beta(gr)$,
i.e., non-compliability enforces a compliance violation of the distributed and collaborative workflow.

Theorem 2 (Compliance Preservation): Global compliance ensures compliance of each trace producible on the entire distributed and collaborative workflow.

Proof 2 (Compliance Preservation):
Since $\mathfrak{A}_{\mathcal{G}} \supseteq \mathfrak{A}_{\mathcal{G},\Gamma} \supseteq \mathfrak{G}$ holds, we may directly conclude:
$\mathcal{G}, \Gamma \vDash gr :\Leftrightarrow \forall \tau \in \mathfrak{G} \subseteq \mathfrak{A}_{\mathcal{G},\Gamma} : \tau \vDash \bigwedge\limits_{a \in A} \beta(a) \Rightarrow \beta(gr)$,
i.e., global compliance ensures compliance of each trace producible on the entire distributed and collaborative workflow (assuming that assertions are not violated).

V. Compliability and Compliance Checking

Taking the formal definition of compliability and compliance (cf. Def. 4), this section shows how these properties can be checked and verified. We provide algorithms that extend interaction and process models with additional structures and tasks (i.e., activities) in order to approximate the private (i.e., non-visible) behavior of the partner processes. Further, we extend compliance rules with preconditions reflecting the respective assertions. These *extended* models and *combined* compliance rules are then used as input for model checking.

A. Compliability Checking

Compliability constitutes a novel semantic correctness criterion for interaction models in the context of distributed and collaborative workflows. It ensures that an interaction model does not conflict with the set of imposed compliance rules. In the context of our methodology for ensuring compliance of distributed and collaborative workflows (cf. Sect. III), compliability is checked in Step 3.

Similar to approaches for checking compliance of intra-organizational processes, we apply model checking techniques to ensure compliability [12]. As opposed to these approaches, however, we cannot directly take the interaction model $\mathcal{G}$ as input for model checking, but have to consider all tasks executed by the partners (cf. Def. 4). Furthermore, we do not want to show that all traces comply with particular rules, but that there is a trace satisfying all compliance rules (cf. Def. 4). Thus, we can not apply model checking directly, but have to add preprocessing steps: First, function $extendInteractionModel$ (cf. Alg. 1) utilizes the knowledge about the set of tasks T_p that p can execute. In particular, the interaction model $\mathcal{G}$ is enriched with parts simulating the behavior of the partners involved and reflecting $\mathfrak{A}_{\mathcal{G}}$. This enrichment is expressed through the use of process structures like sequences (SEQ), parallelism (PAR), choice (CHC), and repetition (RPT). Note that this does not require the interaction model to be well structured.

Second, function $combineRules$ (cf. Alg. 2) builds the conjunction of all global compliance rules. Third, in Alg. 3 applies LTL model checking to the result of function $extendInteractionModel$ (i.e, the extended interaction model EIM) and the negated result of function $combineRules$ (i.e., the negated conjunction of all global compliance rules). In case the interaction model is compliable, at least one trace τ is producible through EIM satisfying all global compliance rules; consequently, the negated conjunction of the global compliance rules does not hold. For this case, LTL model checking returns false (and outputs τ as counter-example), which means that compliabily holds. Otherwise, all traces violate at least one of the global compliance rules, and EIM satisfies the negated formula. In this case, model checking returns true and compliability is violated. Thus, our

Algorithm 1: Extend interaction model

```
Function $extendInteractionModel(\mathcal{I}, \mathcal{P}, A_p)$ ;
begin
    $PM := EMPTY$;
    foreach partner $p \in \mathcal{P}$ do
        foreach task $t \in T_p$ do
            $PM := CHC(t, PM)$;
        end
    end
    $EIM := PAR(\mathcal{I}, RPT(PM))$;
end
Output: $EIM$ Extend interaction model
```

Algorithm 2: Combine global compliance rules

```
Function $combineRules(GR)$ ;
begin
    $CGR := true$;
    foreach global compliance rule $r \in GR$ do
        $CGR := CGR \wedge r$;
    end
end
Output: $CGR$ the combined global compliance rules
```

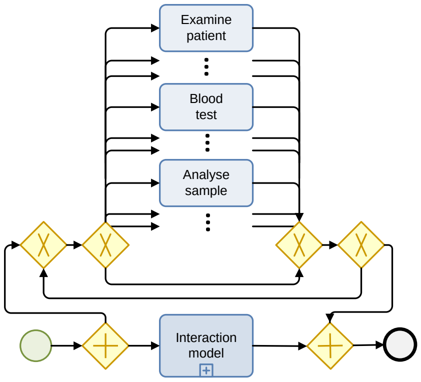

Figure 5: Extended interaction model EIM

Algorithm 3: Compliability checking

```
begin
    $EIM := extendInteractionModel(\mathcal{I}, \mathcal{P}, T_p)$;
    $CGR := combineRules(GR)$;
    $MC\_Res := LTLModelChecking(EIM, \neg CGR)$;
    $C := \neg MC\_Res.propertyValid$;
end
Output: $C$ the compliability of $\mathcal{G}$ with $GR$
```

algorithm for compliability checking basically applies model checking over the extended interaction model EIM and the negated conjunction of the global compliance rules inverting the result (cf. Alg. 3).

Regarding *comprehensive compliability*, we must additionally ensure that for each interaction i of the global interaction model $\mathcal{G}$, at least one trace τ is producible through EIM, which then satisfies all global compliance rules and also contains i. For a particular interaction i, such a trace can be found by adding the LTL expression $\mathbf{F}i$ to the negated conjunction of the global compliance rules before applying model checking as in Alg. 4. $\mathbf{F}i$ requires i to occur eventually. A simple, but costly approach would be to accomplish this for each interaction individually. However, it is more efficient to analyze the counter example τ returned by LTL model checking after these steps have been accomplished for the first interaction i. Besides, τ may contain additional interactions i', i'', and so forth. Consequently, it is not necessary to search for a compliant trace that contains $i', i'', \ldots$ explicitly. Hence comprehensive compliability checking can be continued for the remaining interactions (cf. Alg. 4).

Algorithm 4: Comprehensive compliability checking

```
begin
    $EIM := extendInteractionModel(\mathcal{G}, \mathcal{P}, T_p)$;
    $CGR := combineRules(GR)$;
    $Q_I := \{I(m,s,r) \in \mathcal{M} | I(m,s,r) \in^+ \mathcal{G}\}$;
    $C^\star := true$;
    while $Q_I \neq \emptyset$ do
        Chose arbitrary interaction $I(m,s,r) \in Q_I$;
        $Q_I := Q_I - \{I(m,s,r)\}$;
        $\psi := \neg((\mathbf{F}\ I(m,s,r)) \wedge CGR)$;
        $MC\_Res := LTLModelCecking(EIM, \psi)$;
        if $\neg MC\_Res.propertyValid$ then
            foreach interaction $I'(m',s',r') \in MC\_Result.CounterExample$ do
                $Q_I := Q_I - \{I'(m',s',r')\}$;
            end
        end
        else
            $C^\star := false$; break;
        end
    end
end
Output: $C^\star$ the comprehensive compliability of $\mathcal{G}$ with $GR$
```

Algorithm 5: Extend global models

```
Function $extendGlobalModels(\mathcal{G}, \mathcal{P}, S_p, \gamma_p)$ begin
    $EGM := \mathcal{G}$;
    foreach partner $p \in \mathcal{P}$ do
        $SB_p := EMPTY$;
        foreach secret task $s \in S_p$ do
            $SB_p := CHC(s, SB_p)$;
        end
        $SB_p := RPT(SB_p)$;
        $EPM_p := CHC(EMPTY, PAR(\gamma_p, SB_p))$;
        $EGM := PAR(EGM, EPM_p)$;
    end
end
Output: $EGM$ Extended global models
```

B. Compliance Checking

As discussed in Sect. II, there exist several approaches for checking local compliance [2], [15], [16], [55]. Thus, this subsection focuses on checking global compliance of distributed and collaborative workflows, which is performed in Step 7 of our methodology (cf. Sect. III). Obviously, this would not constitute a particular challenge if the private process model of each partner had been completely known. In this case, one could synchronize the private process models based on the messages exchanged and then apply standard compliance checking techniques (e.g., model checking). However, from a global point of view, the private process models π_p are unknown, i.e., only the public process models γ_p and assertions A_p are known. Further, the set of tasks S_p partner p may use in its private process model is known. In case S_p can not be obtained, T_p must be used instead.

Consider Alg. 5: $extendGlobalModels(\mathcal{G}, \mathcal{P}, S_p, \gamma_p)$ describes how to build an extended public process model EPM_p for any partner p, which reflects $\mathfrak{A}_p$. After EPM_p is derived for each partner, these models are combined with interaction model $\mathcal{G}$. The resulting extended global model EGM then approximates $\mathfrak{A}_{\mathcal{G},\Gamma}$ (cf. Sect. IV-A). However, EGM does not ensure correct message exchange and, thus, not exactly reflects $\mathfrak{A}_{\mathcal{G},\Gamma}$. To compensate this, Alg. 6 extends global compliance rules of GR with preconditions that filter out incorrect message exchanges. Furthermore, the other preconditions of gr take assertions into account as introduced in the definition of global compliance (cf. Def. 4). Finally, Alg. 7 applies model checking to verify global compliance.

Algorithm 6: Extend compliance rule

```
Function $extendComplianceRule(cr, A, \mathcal{M})$ ;
begin
    $\alpha := true; \quad \iota := true;$
    foreach assertion $a \in A$ do
        $\alpha := \alpha \wedge \beta(a);$
    end
    foreach interaction $i = I(m,s,r) \in \mathcal{M}$ do
        $s := (Task = SEND(m,r) \wedge Performer = s);$
        $i := (Task = I(m,s,r));$
        $r := (Task = REC(m,s) \wedge Performer = r);$
        $\iota := \iota \wedge \big(\mathbf{G}(\ (s \Leftrightarrow \mathbf{X}i) \wedge (i \Leftrightarrow \mathbf{X}r)\ )\big);$
    end
    $ER := (\alpha \wedge \iota) \Rightarrow \beta(cr);$
end
Output: $ER$ the extended compliance rule
```

Algorithm 7: Global compliance checking

```
begin
    $EGM := extendGlobalModels(\mathcal{G}, \mathcal{P}, S_p, \gamma_p);$
    $ER := extendComplianceRule(cr, A, \mathcal{M});$
    $MC\_Result := LTLModelCecking(EGM, ER);$
    $C := MC\_Result.propertyValid;$
end
Output: $C$ global compliance with $cr$
```

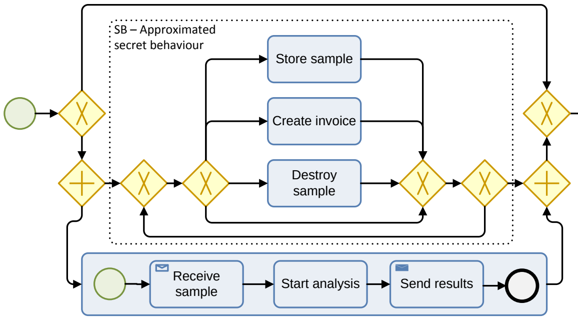

Figure 6: Extended partner model EPM_{Lab}

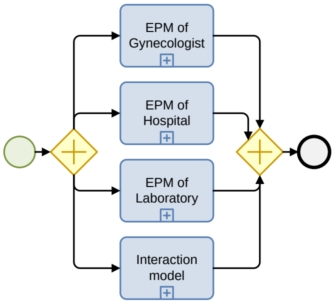

Figure 7: Extended global models EGM

C. Proof-of-Concept Prototype

We have demonstrated the feasibility of our approach by implementing a proof-of-concept prototype. We applied this prototype to different application scenarios including the presented healthcare example [9]. More precisely, the presented compliance checking techniques have been implemented as plug-in of the Aristaflow BPM Suite [56] and partially base on libraries of the Seaflows toolset [55]. [12] provides provide a screenshot of the compliability checker.

VI. Summary and Outlook

Ensuring compliance of workflows with guidelines, standards, and laws is crucial for both intra-organizational as well as distributed and collaborative settings. However, most existing proposals have only considered intra-organizational workflows so far [9]. This paper introduces a sophisticated approach that allows ensuring compliance of distributed and collaborative workflows with multiple partners at different levels. More precisely, we introduce three levels of business process compliance reflected by local compliance rules, assertions, and global compliance rules. Based on this, we enrich interaction-modeling [41] with additional tasks to detect potential compliance violations at an early stage. In this context, we formally introduce two notions of *compliability*, i.e., the general ability of an interaction model not to conflict with a given set of compliance rules in a distributed and collaborative setting, independent from the particular process models of the partners. Our formal definitions of compliability are complemented by algorithms for compliability checking. Further, we introduce the notion of global compliance and present algorithms for checking it. As opposed to compliability, compliance checking examines the public elements of a distributed and collaborative workflow, i.e. the interaction model, public process models, and assertions. To the best of our knowledge, there exists no other approach ensuring compliance and compliability of distributed and collaborative workflows taking privacy constraints into account, i.e. the non-availability of information on the private elements of partner processes.

In future work, we will consider additional perspectives in the context of compliance and compliability checking (e.g. data, time, resources). Furthermore, we will consider compliance of distributed and collaborative workflows in all phases of their life cycle and further evaluate our approach in industrial case studies.

Acknowledgment

This work was done within the research project C^{3}Pro funded by the German Research Foundation (DFG), Project number: RE 1402/2-1, and the Austrian Science Fund (FWF), Project number: I743. Further, we would like to thank Andreas Lanz for his support with the Aristaflow BPM Suite.

References

[1] Sadiq, S., Governatori, G., Naimiri, K.: Modeling control objectives for business process compliance. In: BPM'07. (2007) 149–164

[2] Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: BPM'08. (2008) 326–341

[3] Ly, L.T., Rinderle-Ma, S., Göser, K., Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems. Information Systems Frontiers **14**(2) (2012) 195–219

[4] Knuplesch, D., Reichert, M.: Ensuring business process compliance along the process life cycle. Technical Report 2011-06, University of Ulm (2011)

[5] Maggi, F., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring business constraints with linear temporal logic: an approach based on colored automata. In: BPM'11. (2011) 132–147

[6] Ramezani, E., Fahland, D., Werf, J.M., Mattheis, P.: Separating compliance management and business process management. In: BPM'11 Workshops, Springer (2012) 459–464

[7] Berry, A., Milosevic, Z.: Extending choreography with business contract constraints. Int J Coop Inf Sys **14**(2-3) (2005) 131–179

[8] Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: EDOC'06. (2006) 221–232

[9] Knuplesch, D., Reichert, M., Mangler, J., Rinderle-Ma, S., Fdhila, W.: Towards compliance of cross-organizational processes and their changes - research challenges and state of research. In: BPM'12 Workshops. Volume 132 of LNBIP. (2013) 649–661

[10] Leitner, M., Mangler, J., Rinderle-Ma, S.: Definition and enactment of instance-spanning process constraints. In: WISE'2012. (2012) 652–658

[11] Müller, D., Herbst, J., Hammori, M., Reichert, M.: It support for release management processes in the automotive industry. In: BPM'06. (2006) 368–377

[12] Knuplesch, D., Reichert, M., Fdhila, W., Rinderle-Ma, S.: On enabling compliance of cross-organizational business processes. In: BPM'13. Volume 8094 of LNCS. (2013) 146–154

[13] Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Hints on how to face business process compliance. In: JISBD'10. (2010)

[14] Ly, L.T., Maggi, F.M., Montali, M., StefanieRinderle-Ma, van der Aalst, W.M.: A framework for the systematic comparison and evaluation of compliance monitoring approaches. In: EDOC'13. IEEE (2013)

[15] Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling data-aware compliance checking of business process models. In: ER'2010. Volume 6412 of LNCS. (2010) 332–346

[16] Ghose, A.K., Koliadis, G.: Auditing business process compliance. In: ICSOC'07. (2007) 169–180

[17] Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Property specification patterns for finite-state verification. In: FMSP'98. (1998)

[18] Ramezani, E., Fahland, D., van der Aalst, W.M.: Where did i misbehave? Diagnostic information in compliance checking. In: BPM'12, Springer (2012)

[19] Turetken, O., Elgammal, A., van den Heuvel, W.J., Papazoglou, M.: Capturing compliance requirements: A pattern-based approach. IEEE Software (2012) 29–36

[20] Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Systems Journal **46**(2) (2007) 335–261

[21] Ly, L.T., et al.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: CAiSE'10. (2010) 9–23

[22] Feja, S., Speck, A., Witt, S., Schulz, M.: Checkable graphical business process representation. In: ADBIS'11, Springer (2011) 176–189

[23] Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: ICSOC'09. (2009) 500–515

[24] Awad, A., Weidlich, M., Weske, M.: Visually specifying compliance rules and exp-laining their violations for business processes. Vis Lang Comp **22**(1) (2011) 30–55

[25] Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: Visual modeling of business process compliance rules with the support of multiple perspectives. In: ER'13 (accepted for publication). (2013)

[26] Ly, L.T., et al.: Integration and verification of semantic constraints in adaptive process management systems. Data & Knowl Eng **64**(1) (2008) 3–23

[27] Weber, B., Reichert, M., Wild, W., Rinderle-Ma, S.: Providing integrated life cycle support in process-aware information systems. Int J Coop Inf Sys **18**(1) (2009) 115–165

[28] Kokash, N., Krause, C., de Vink, E.: Time and data aware analysis of graphical service models. In: SEFM'10. (2010)

[29] Höhn, S.: Model-based reasoning on the achievement of business goals. In: SAC '09, New York, NY, USA, ACM (2009) 1589–1593

[30] Accorsi, R., Lowis, L., Sato, Y.: Automated certification for compliant cloud-based business processes. Business & Inf Sys Engineering **3**(3) (2011) 145–154

[31] Kumar, A., Yao, W., Chu, C.: Flexible process compliance with semantic constraints using mixed-integer programming. INFORMS J on Comp (2012)

[32] Alles, M., Kogan, A., Vasarhelyi, M.: Putting continuous auditing theory into practice: Lessons from two pilot implementations. Inf Sys **22**(2) (2008) 195–214

[33] Namiri, K., Stojanovic, N.: Pattern-Based design and validation of business process compliance. In: CoopIS'07. (2007) 59–76

[34] Giblin, C., et al.: From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Technical Report RZ-3662, IBM (2006)

[35] Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: CoopIS'11. (2011) 82–99

[36] van der Aalst, W.M.P., Beer, H.D., van Dongen, B.: Process mining and verification of properties: An approach based on temporal logic. In: CoopIS'05. (2005) 130–147

[37] Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: full support for loosely-structured processes. In: EDOC'07. (2007) 287–300

[38] Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In: BPM'06 Workshops. (2006) 5–14

[39] Alberti, M., et al.: Expressing and verifying business contracts with abductive logic programming. In: NorMAS'07. Dagstuhl Seminar Proceedings (2007)

[40] Zaha, J., Barros, A., Dumas, M., ter Hofstede, A.: Let's dance: A language for service behavior modeling. In: CoopIS'06. (2006) 145–162

[41] Decker, G., Weske, M.: Interaction-centric modeling of process choreographies. Inf Sys **35**(8) (2010)

[42] Barros, A., Dumas, M., ter Hofstede, A.: Service interaction patterns. In: BPM'05. (2005) 302–318

[43] Liu, D.R., Shen, M.: Business-to-business workflow interoperation based on process-views. Decision Support Sys **38**(3) (2004) 399–419

[44] Maamar, Z., Benslimane, D., Ghedira, C., Mrissa, M.: Views in composite web services. IEEE Internet Comp **9**(4) (2005) 52–57

[45] Decker, G., Weske, M.: Behavioral consistency for B2B process integration. In: CAiSE'07. (2007) 81–95

[46] van der Aalst, W.M.P.: Inheritance of interorganizational workflows to enable Business-to-Business E-Commerce. Elec Com Research **2**(3) (2002) 195–231

[47] Fdhila, W., Rouached, M., Godart, C.: Communications semantics for wsbpel processes. In: ICWS'08. (2008)

[48] Rouached, M., Fdhila, W., Godart, C.: Web services compositions modelling and choreographies analysis. IJWSR **7**(2) (2010) 78–110

[49] Lohmann, N., Wolf, K.: Realizability is controllability. Web Services and Formal Methods (2010) 110–127

[50] Knuplesch, D., Pryss, R., Reichert, M.: Data-aware interaction in distributed and collaborative workflows: Modeling, semantics, correctness. In: IEEE CollaborateCom'12. (2012) 223–232

[51] Lohmann, N., Wolf, K.: Decidability results for choreography realization. In: ICSOC'11. (2011) 92–107

[52] van der Aalst, W.: Verification of workflow nets. Application and Theory of Petri Nets 1997 (1997) 407–426

[53] Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer (2012)

[54] Fdhila, W., Rinderle-Ma, S., Reichert, M.: Change propagation in collaborative processes scenarios. In: IEEE CollaborateCom'12. (2012) 452 – 461

[55] Ly, L.T., Knuplesch, D., Rinderle-Ma, S., Göser, K., Pfeifer, H., Reichert, M., Dadam, P.: Seaflows toolset–compliance verification made easy for process-aware information systems. Inf Syst Evolution (2011) 76–91

[56] Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science-Research and Development **23**(2) (2009) 81–97