ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Enabling Multi-Perspective Business Process Compliance

Dissertation zur Erlangung des Doktorgrades Dr. rer. nat.
an der Fakultät für Ingenieurwissenschaften, Informatik und Psychologie der Universität Ulm

Vorgelegt von:
David Knuplesch
geboren in Tübingen

2019

Amtierender Dekan:	Prof. Dr.-Ing. Maurits Ortmanns
Gutachter:	Prof. Dr. Manfred Reichert
	Prof. Dr. Stefanie Rinderle-Ma

Tag der Promotion: 22.07.2019

Vorwort

Zwischen dem Tag, an dem der Entschluss für meine Dissertation fiel, und dem heutigen Tag, an dem ich dieses Vorwort schreibe, sind viele Jahre vergangen, auf die ich nun mit einer Mischung aus Wehmut und Erleichterung zurückblicke. Jetzt, wo das Ende dieses bisweilen faszinierenden und steinigen Weges erreicht ist, bleibt mir als letzter Schritt nur noch das Formulieren des Dankes an jene, die mich auf diesem Weg begleitet haben.

An erster Stelle danke ich meinem Doktorvater Prof. Dr. Manfred Reichert für die Möglichkeit zur Promotion und für seine stets umfangreichen Korrekturen, die in nicht unerheblichem Maße die Lesbarkeit und Verständlichkeit meiner Texte verbessert haben.

Im Anschluss daran danke ich Prof. Dr. Stefanie Rinderle-Ma und Dr. Walid Fdhila für die gemeinsame, publikationsreiche Zeit im C^3Pro Projekt. Prof. Dr. Akhil Kumar danke ich für die stets angenehme Zusammenarbeit und seinen umsichtigen und konstruktiven Rat.

Ich danke all meinen Kolleginnen und Kollegen des Instituts für Datenbanken und Informationssysteme für die schönen Momente, die gemeinsamen Konferenzreisen und die interessanten Diskussionen. Besonders danke ich Dr. Linh Thao Ly, die mein Interesse am Themenkomplex Business Process Compliance weckte und mit deren Unterstützung meine ersten Publikationen entstanden. Dr. Andreas Lanz danke ich für die Latex-Vorlage und für seine hilfreichen Erfahrungswerte zur kumulativen Dissertation. Luisa Reinbold danke ich für die organisatorische Unterstützung in der kritischen Endphase meiner Dissertation.

Herzlich bedanken möchte ich mich bei Raphael Herfort, Franziska Semmelrodt, Hannes Beck und Manoubiya Zidi, welche im Rahmen ihrer Abschluss- und Projektarbeiten mit Prototypen, Fallstudien und empirische Untersuchungen die umfangreiche Validation meiner Forschung erst ermöglichten.

Zu Dank verpflichtet bin ich auch der Friedrich-Ebert-Stiftung für die finanzielle und ideelle Unterstützung, die sie mir in den ersten Jahren meiner Dissertation im Rahmen eines Stipendiums gewährte.

Mein größter Dank gilt aber meinen beiden Kinder Miên-Lia Mai und Đan Erik Rei sowie meiner Frau.

Neu-Ulm, August 2019

Abstract

A particular challenge for any enterprise is to ensure that its business processes conform with compliance rules, i.e., semantic constraints on the multiple perspectives of the business processes. Compliance rules stem, for example, from legal regulations, corporate best practices, domain-specific guidelines, and industrial standards. In general, compliance rules are multi-perspective, i.e., they not only restrict the process behavior (i.e. control flow), but may refer to other process perspectives (e.g. time, data, and resources) and the interactions (i.e. message exchanges) of a business process with other processes as well.

The aim of this thesis is to improve the specification and verification of multi-perspective process compliance based on three contributions:

1. The extended Compliance Rule Graph (eCRG) language, which enables the visual modeling of multi-perspective compliance rules. Besides control flow, the latter may refer to the time, data, resource, and interaction perspectives of a business process.
2. A framework for multi-perspective monitoring of the compliance of running processes with a given set of eCRG compliance rules.
3. Techniques for verifying business process compliance with respect to the interaction perspective. In particular, we consider compliance verification for cross-organizational business processes, for which solely incomplete process knowledge is available.

All contributions were thoroughly evaluated through proof-of-concept prototypes, case studies, empirical studies, and systematic comparisons with related works.

Kurzfassung

Eine wichtige Herausforderung für Unternehmen ist es, die *Compliance* ihrer Geschäftsprozesse mit semantischen Constraints (sog. Compliance Regeln) sicherzustellen. Compliance Regeln leiten sich z.B. aus gesetzlichen Vorgaben, Best Practies, domänenspezifischen Richtlinien und Standards ab. Sie formulieren u.a. Bedingungen an die Ausführungsreihenfolge von Arbeitsschritten., beziehen sich im Allgemeinen aber nicht nur auf die Kontrollflussperspektive, sondern auch auf andere Prozessperspektiven (z.B. Zeit, Daten und Resourcen) sowie die Interaktionen mit Partnerprozessen (d.h. den Austausch von Nachrichten).

Ziel der vorliegenden Arbeit ist die Unterstützung der Modellierung und Verifikation von Compliance Regeln unter Einbeziehung aller oben genannten Prozessperspektiven. Dazu werden drei wissenschaftliche Beiträge gemacht:

1. Die *extendend Compliance Rule Graph* (eCRG) Sprache für die visuelle Modellierung von Compliance Regeln unter Einbeziehung von Verhalten (d.h. Kontrollfluss), Zeit, Daten, Ressourcen, ebenso wie auf den Nachrichtenflüssen mit Partnerprozessen.
2. Ein Rahmenwerk zur Überwachung von eCRG-basierten Compliance Regeln für laufende Geschäftsprozesse.
3. Techniken für die Verifikation von Compliance Regeln im Kontext unternehmenübergreifender Prozesse, für welche lediglich eingeschränktes Wissen über die beteiligten Partnerprozesse vorliegt.

Die Beiträge dieser Arbeit werden mittels Proof-of-Concept Prototypen, Fallstudien, Empirischen Studien und systematischen Vergleich mit verwandten Arbeiten evaluiert.

Contents

1 Introduction

1.1 Motivation

Ensuring the correctness of its business processes constitutes a crucial task for any enterprise [Ly13, Aa18, FZ14]. Besides guaranteeing structural and dynamic soundness criteria, processes need to comply with semantic constraints, which we denote as *compliance rules* in this thesis. Corresponding rules can be derived from a variety of artifacts like, for example, legal regulations, corporate best practices, domain-specific guidelines, and industrial standards [GS09, Ly13, LR07]. In literature, *compliance rules* are often restricted to control flow constraints on a business process (i.e. to the occurrence and order of activities). In general, however, compliance rules may also refer to other process perspectives, including the time, resource, and data perspectives. In the context of cross-organizational business processes [FIRMR15], in addition, compliance rules may refer to interactions (i.e., message exchanges) between business partners.

As a prerequisite for ensuring business process compliance, a language for the formal and computer-readable specification of compliance rules is needed; in current practice, these rules are usually described informally. Formally specified compliance rules, can then be taken as input for automated compliance checking. However, depending on whether compliance shall be checked at design or run time and on the kind of process (i.e., intra- vs. cross-organizational), different process information is available. Consequently, several compliance checking techniques are required to validate business process compliance for the various scenarios.

1.2 Scientific Contribution

This cumulative thesis provides three contributions that shall enable the specification, monitoring and verification of multi-perspective business process compliance.

The first contribution focuses on the visual specification of multi-perspective compliance rules. For this purpose, [KRL$^+$13b, KR17] introduce the visual *extended Compliance Rule Graph* (eCRG) language, which enables the visual modeling and verification of compliance rules that may refer to the control flow, resource, data, time, and interaction

perspectives of a business process. In particular, eCRGs rely on a formal semantics specified in terms of first-order logic. To evaluate eCRGs several steps are taken: First, a proof-of-concept prototype is implemented, enabling the modeling of eCRGs as well as their verification against the event logs of completed process instances. Second, case studies, empirical studies, and a systematic comparison with related works shall demonstrate the applicability and usefulness of the eCRG language.

The second part of the thesis introduces a framework for the online monitoring (i.e., compliance checking) of eCRG compliance rules during process execution [KRK15b, KRK17]. For this purpose, the events produced by running processes are observed. In response to these events, the elements of an eCRG are marked step-by-step with text labels, colors and symbols in order to highlight the respective state of the compliance rule. In particular, the eCRG monitoring framework meets all compliance monitoring functionalities (CMF), which are considered as relevant in [LMM+15]. For example, the framework is able to continuously and reactively monitor multiple instances of the same eCRG. In addition, the eCRG monitoring framework provides compliance metrics and supports root-cause analyses at the occurrence of compliance rule violations. The eCRG monitoring framework is evaluated through a proof-of-concept implementation, which is applied to different use cases. Finally, the prototype is subject to performance

Table 1.1: Contributions

Scientific contribution	**Evaluation**	**Publications**
Modeling process compliance rules • expressive visual notation for modeling compliance rules; • support of multiple process perspectives, i.e., control flow, data, time, resources, and interaction; • formal specification.	• proof-of-concept prototype • case study • pattern-based evaluation • empirical studies	[KRL+13b] [KR17]
Framework for monitoring business process compliance • support of multiple process perspectives, i.e., control flow, data, time, resources, and interaction; • enables continuous monitoring of multiple compliance rule instantiations; • visual feedback for users.	• proof-of-concept Prototype • performance measurements • evaluation against CMF	[KRK15a] [KRK15b] [KRK17]
Checking the compliance of cross-organizational processes • compliance criteria for cross-organizational processes • support of privacy and not-yet-specified behavior • estimates the effects of changes on compliance	• proof-of-concept Prototype	[KRM+13] [KRFRM13] [KRP+13] [KFRRM15]

measurements.

The third part of the thesis introduces compliance checking techniques for cross-organizational business processes. Thereby, the particular challenge is to cope with incomplete process information. [KRFRM13, KRP+13] introduce compliance criteria for cross-organizational processes as well as corresponding techniques for verifying compliance in this context. Finally, [KFRRM15] limits the number of compliance rules to be rechecked after changing a cross-organizational business process. A sophisticated proof-of-concept implementation evaluates the presented contributions.

Table 1.1 summarizes the contributions of this thesis.

1.3 Outline of the Thesis

The cumulative thesis is structured as follows: Chapter 2 introduces fundamental concepts and notions needed for understanding the thesis. Chapter 3 then presents the extended Compliance Rule Graph (eCRG) language, which enables the modeling of multi-perspective compliance rules. The framework for visually monitoring multi-perspective compliance rules during process execution (i.e. at run time) is presented in Chapter 4. Chapter 5 introduces the approach for verifying the compliance of cross-organizational business processes at design time. Finally, Chapter 6 provides a summary and an outlook.

All publications related to the cumulative thesis are contained in the appendices.

2 Backgrounds

This chapter introduces fundamental concepts and basic terminology needed for understanding this thesis.

2.1 Business Processes

According to [Wes12], a *business process* is a set of *activities* that are performed in order to achieve a certain and recurring business goal, e.g., the processing of an order or the treatment of a patient [DRMR18, LR07]. A *process model*, in turn, describes the execution constraints for the activities of a business process.

Example 2.1 (Business process)
A simple example of a process from the healthcare domain is depicted in Fig. 2.1. It outlines an outpatient treatment that comprises five activities. The process starts with the admission of the patient. Then, the latter is examined and a diagnosis is made. Finally, medicine is prescribed and the patient is discharged.

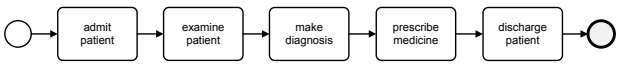


Figure 2.1: Patient treatment process

The life cycle of a business processes comprises four phases [WRRM08]:

- In the **modeling phase** (i.e., *design time*), a (visual) *process model* is specified.
- In the **execution phase** (i.e., *run time*), instances of the *business processes* model are created and then executed according to the logic prescribed by this process model. To allow for a comprehensive analysis of running processes, *process logs* record all events occurring during the execution of the process instances.
- The **change phase** deals with instance-specific changes, which are recorded in *change logs*.

- The **analysis phase** refers to the a-posteriori analysis of process logs created during the execution of process instances. The result of this phase, in turn, might trigger changes of the process model.

The four phases may overlap if multiple instances of the same process are considered. While some process instances are still running, business analysts may already investigate the logs of completed process instances of the same model. Likewise, changes may be applied to a model, while corresponding process instances are still running [RRD04].

2.1.1 Process Models

According to [Wes12], a *process model* describes the execution constraints between the activities of a business process. In this thesis, *process models* are considered to be directed graphs that are composed of different kinds of nodes and edges. In particular, the Business Process Model and Notation 2.0 (BPMN 2.0) standard [OMG14] is applied.

Besides the *start* and *end nodes* of a process, the most basic nodes of a BPMN process model refer to process activities. The edges linking the *activity nodes* are denoted as *sequence flow (edges)* and define the order between the activities. In turn, *gateways* are special nodes that allow splitting and merging sequence flows in order to realize alternative or parallel execution branches. For example, if an *AND-split gateway* is reached, each of its outgoing sequence flows becomes enabled, whereas an *XOR-split gateways* enforces the selection of exactly one of the outgoing flows. Accordingly, *AND-join gateways* correspond to synchronization points that wait for all incoming sequence flows, whereas *XOR-join gateways* wait for exactly one of the incoming sequence flow edges, before processing with control flow. All these elements form the *control flow perspective* of a process model.

Example 2.2 demonstrates the use of these elements along the process of an X-ray examination for an inpatient in a hospital [KSR96b, Sem13].

Example 2.2 (X-ray examination process for inpatients)
The process starts with a physical examination of the inpatient by a ward physician who then orders an X-ray examination. For this purpose, a corresponding form has to be filled, which collects information about the examination as well as required preparations and aftercare. The order form is then forwarded to the Radiology Department, which schedules an appointment for the X-ray examination.

Before the examination, the ward physician informs the patient about the risks of the X-ray examination and asks for her informed consent. If required, a nurse prepares the inpatient before transporting her to the Radiology Department at the time of the examination. However, this transport must not be started unless the secretary of the Radiology Department asks for the inpatient. While the patient is dropped off, the signed informed consent is sent to the Radiology Department.

The Radiology Department informs the ward as soon as the X-ray examination is completed. Following this, a nurse from the ward picks up the patient, bringing her back to the ward. If required, aftercare is provided by the nurse. At some time, the Radiology Department sends the digital X-ray images. Finally, the ward physician uses this images to make a diagnosis, prescribes a therapy for the inpatient, and documents the therapy.

Fig. 2.2 depicts the process model introduced by Example 2.2. Note that there exist other expressive process model languages, like ADEPT2 [RRKD05, DR09] or YAWL [AH05], which enable the visual modeling of business processes as well.

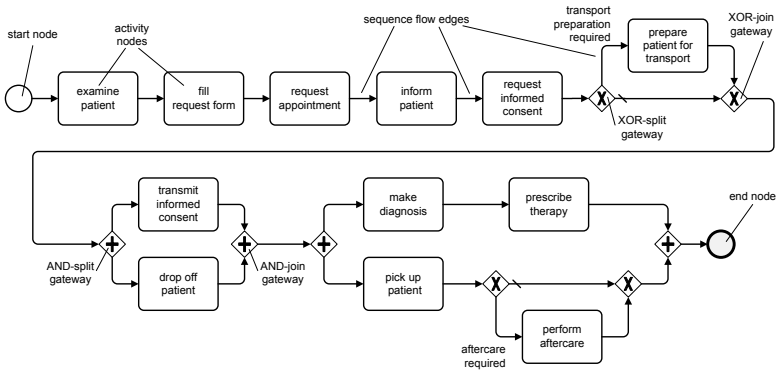


Figure 2.2: X-ray examination process expressed as BPMN 2.0 model

2.1.2 Process Execution

In general, *process models* should be based on a well-defined operational semantics that enables their proper simulation and execution. Thus, process models are transformed into a stateful representation that allows determining the current process state as well as the set of valid actions (e.g. start or completion of an activity) that may be applied in this state. Furthermore, for each of these actions the state resulting afterwards must be clear. A stateful representation of a process model can be achieved, for example, through its translation into a stateful representation using another formalism, e.g., Petri-nets, or by annotating process elements with tokens.

Process Instances

A *process instance* represents a concrete case of a business process [Wes12]. In particular, a *process instance* reflects a particular state of the corresponding process model.

Example 2.3 (Token-based process execution)
Fig. 2.3 shows an instance of the X-ray examination process as depicted in Fig. 2.2 (cf. Example 2.2). Thereby, the instance state is indicated through tokens. Regarding the depicted instance, activities `examine patient`, `fill request form`, `request appointment`, `inform patient`, and `request informed consent` are completed, whereas activity `drop off patient` is running and activity `transmit informed consent` may still be started.

Example 2.4 (Marking-based process execution)
Fig. 2.4 shows the process instance using the ADEPT2 formalism [RRKD05, Rei00]. Markings are used to indicate the current state of the process instance. Thereby, activities are either marked as *activated*, *running*, *completed*, *skipped*, or *not active* to highlight the current state as well as possible actions. For example, activities `examine patient` and `fill request form` were already executed, whereas activity `prepare patient for transport` was skipped, i.e., the other branch of the preceding XOR-split gateway was chosen (cf. Section 2.1.1). Finally, activities `drop off patient` and `transmit informed consent` are marked as *running* and *activated*, respectively.

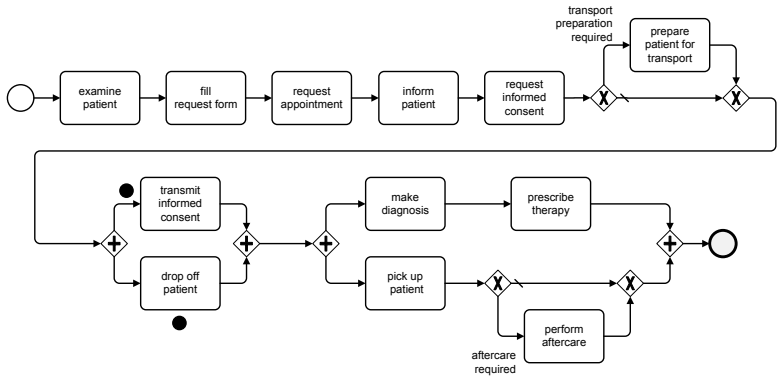


Figure 2.3: Token-based instance of the X-ray examination process

Process logs and traces

Process logs record *activities* and *events* that have occurred during the execution of the respective process instance.

A fundamental kind of process log is provided by *activity logs*, which record the sequence in which the activities of a process instance were started and completed respectively. Table 2.1 shows a possible activity log of the X-ray examination process introduced in Example 2.2.

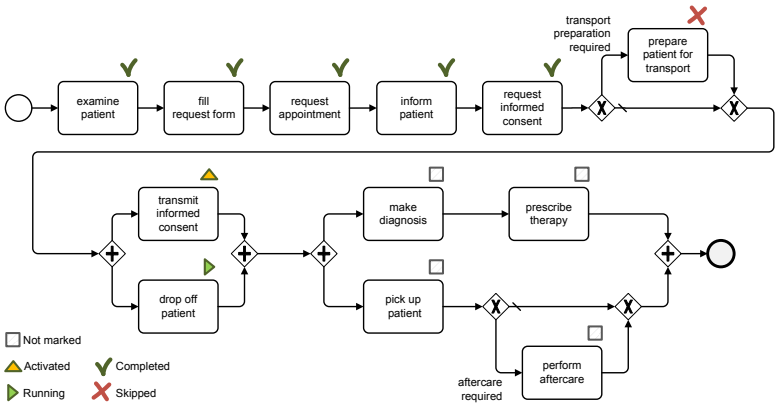


Figure 2.4: Instance of the X-ray examination process in ADEPT2

Activity logs (cf. Table 2.1) contain exactly one atomic entry for each activity instance and, hence, cannot describe concurrent executions of activities without additional information. In turn, *event logs* (i.e., another kind of process logs) allow distinguishing between different events referring to the same instance of an activity. Event logs contain at least the events referring to the start and end of the executed activities. Entries of an event log need additional attributes to denote the type of the respective event (e.g., start event) and to relate the events referring to the same activity instance.

Example 2.5 (Process log)
Table 2.2 shows an example of an event log of the X-ray examination process (cf. Example 2.2). It refers to the same process instance as the activity log from Table 2.1. However, Entries 13-16 highlight that activities `drop off patient` and `transmit informed consent` are concurrently executed, whereas this does not become clear from the activity log depicted in Table 2.1.

Table 2.1: Activity log of the X-ray examination process (cf. Example 2.2)

#	activity
1	examine patient
2	fill request form
3	request appointment
4	inform patient
5	request informed consent
6	prepare patient for transport
7	drop off patient
8	transmit informed consent
9	pick up patient
10	perform aftercare
11	make diagnosis
12	prescribe therapy

Table 2.2: Event log of the X-ray examination process (cf. Example 2.2)

#	type	id	activity
1	start	1	examine patient
2	end	1	examine patient
3	start	2	fill request form
4	end	2	fill request form
5	start	3	request appointment
6	end	3	request appointment
7	start	4	inform patient
8	end	4	inform patient
9	start	5	request informed consent
10	end	5	request informed consent
11	start	6	prepare patient for transport
12	end	6	prepare patient for transport
13	start	6	drop off patient
14	start	7	transmit informed consent
15	end	7	transmit informed consent
16	end	6	drop off patient
17	start	8	pick up patient
18	end	8	pick up patient
19	start	9	perform aftercare
20	end	9	perform aftercare
21	start	10	make diagnosis
22	end	10	make diagnosis
23	start	11	prescribe therapy
24	end	11	prescribe therapy

To distinguish between logs of still running and logs of already completed process instances, the former are denoted as *partial logs*, whereas the latter are summarized under the term *completed log*. Note that, in general, process logs refer to real process instances. In turn, a *process trace* refers to a theoretically possible execution of a process model. Again, *partial* and *completed traces* need to be distinguished.
Completed traces correspond to theoretically possible executions that reached a final state and, hence, will not be continued, i.e., no activity is running and no activity can be started anymore. In turn, *partial traces* may refer to intermediate states and be continued through the completion of still running activities or the start of not yet enabled ones.

The activity log from Table 2.1 and the event log from Table 2.2 present examples of a completed logs of the X-ray examination process (cf. Example 2.2). In turn, Table 2.3 shows a partial event log reflecting the state of the process instance depicted in Fig. 2.3.

2.1.3 Process Change

The ability to adapt its business process is crucial for any enterprise to cope with environmental changes and exceptional situations [WSR09, RRMD09, RW12, LR16].

Table 2.3: Partial event log of the X-ray examination process (cf. Example 2.2)

#	type	id	activity
1	start	1	examine patient
2	end	1	examine patient
3	start	2	fill request form
4	end	2	fill request form
5	start	3	request appointment
6	end	3	request appointment
7	start	4	inform patient
8	end	4	inform patient
9	start	5	request informed consent
10	end	5	request informed consent
11	start	7	drop off patient

Formally, a *process change* can be considered as a transformation of a process model M into a process model M'.

Basic process model changes include the insertion and deletion of activities [WRRM08]. Based on these changes, more complex changes may be composed, e.g., moving of an activity within a process model can be realized by first removing the activity and then re-inserting it at another position [WPTR13].

Similar to process logs, which record the activities performed during the execution of a process instance, *change logs* record the basic changes that compose a complex process changes [GRRA06, RJR07]. *Change profiles*, in turn, provide a more condensed view on changes as they only outline which activities were added or removed.

Example 2.6 (Process change)
A change of the X-ray examination process (cf. Example 2.2) is illustrated in Fig. 2.5. In particular, activity `make diagnosis` should be postponed until the patient returns from the radiology. Furthermore, the room of the patient should be cleaned during her absence. Table 2.4 shows the corresponding change log, the change profile is depicted in Table 2.5

Table 2.5: Change profile of the X-ray examination process (cf. Example 2.6)

#	activity	change
1	clean room	+
2	make diagnosis	±
3	prescribe therapy	±

Legend:
+ add activity
− remove activity
± add and remove activity

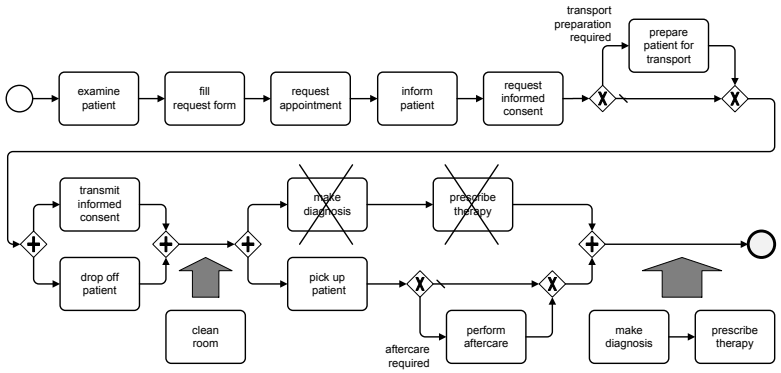


Figure 2.5: A change of the X-ray examination process (cf. Example 2.6)

Table 2.4: Change log of the X-ray examination process (cf. Example 2.6)

#	type	activity	position
1	add	clean room	after: transmit informed consent, drop off patient before: make diagnosis, pick up patient
2	remove	make diagnosis	
3	remove	prescribe therapy	
4	add	make diagnosis	after: pick up patient, perform aftercare
5	add	prescribe therapy	after: make diagnosis

2.1.4 Processes Perspectives other than Control Flow

Obviously, the control flow perspective is not sufficient to specify all aspects of a business process. For example, the process model from Fig. 2.2 neither specifies whether the two activities `request informed consent` and `transmit informed consent` refer to the same `informed consent` nor does the model state that a ward physician shall examine the patient. Finally, Fig. 2.2 does not define the point in time, at which the patient needs to be prepared for the transport. Corresponding aspects are covered by the following process perspectives:

- **Resource perspective**. This perspective specifies the (human) resources that shall execute the various process activities. In general, an activity is not directly assigned to a specific resource, but rather to a resource class based on capabilities, roles, affiliations, or historic process information [RAHE05, CRC11, CKR+15].

- **Time perspective**. This perspective specifies when activities shall be executed and which temporal constraints need to be obeyed. It includes, for example, constraints on time distances between activities or deadlines for activity execution [LWR14, LRW16].

- **Data perspective**. This perspective refers to the *data objects* created, exchanged and processed by activities during the execution of a process instance [Rei12, KPR12a].
- **Interaction perspective**. This perspective defines the interactions a business process has with its partner processes. In general, the interactions refer to the exchange of *messages* with partners [BDH05, FIRMR15].

In order to cover these additional perspectives, process models need to be enriched accordingly. For example, BPMN 2.0 allows modeling the resource perspective with *pools* and *lanes*. *Pools* group the activities according to their affiliation, whereas *lanes* group activities of a particular partner based on roles, positions or capabilities. In turn, *temporal events* enable the modeling of deadlines or delays, whereas the *data flow* can be expressed with *data objects* and *data flow edges* [RD98, Rei00]. Moreover, the exchange of information between partner processes can be expressed through specific activities (or events) sending or receiving messages. Finally, a *message flow edge* connects the sender of a message with its receiver.

Fig. 2.6 depicts the X-ray examination process for inpatients covering all mentioned perspectives.

To properly support the latter, a process log not only needs to record the events referring to the start/end of activities, but additional information on the resource, time, data, and interaction perspectives.

Example 2.7 (Multi-perspective activity and event logs)
Table 2.6 enriches the partial event log depicted in Table 2.3 with additional information covering the resource, time, data, and interaction perspectives. First, the point in time of an event occurred is captured. Second, each human activity is annotated with the human actor (i.e. resource) processing it. As opposed to the event log from Table 2.3, the event log from Table 2.6 contains additional events related to the sending/receipt of messages (e.g., Events 6-8 and 16-17) or describing how data objects are processed (e.g., Events 4 and 7). A partial activity log of the same process instance is depicted in Table 2.7. Note that its entries are enriched with timestamps related to the start and completion of activities. In addition, the involved resources and data objects are listed.

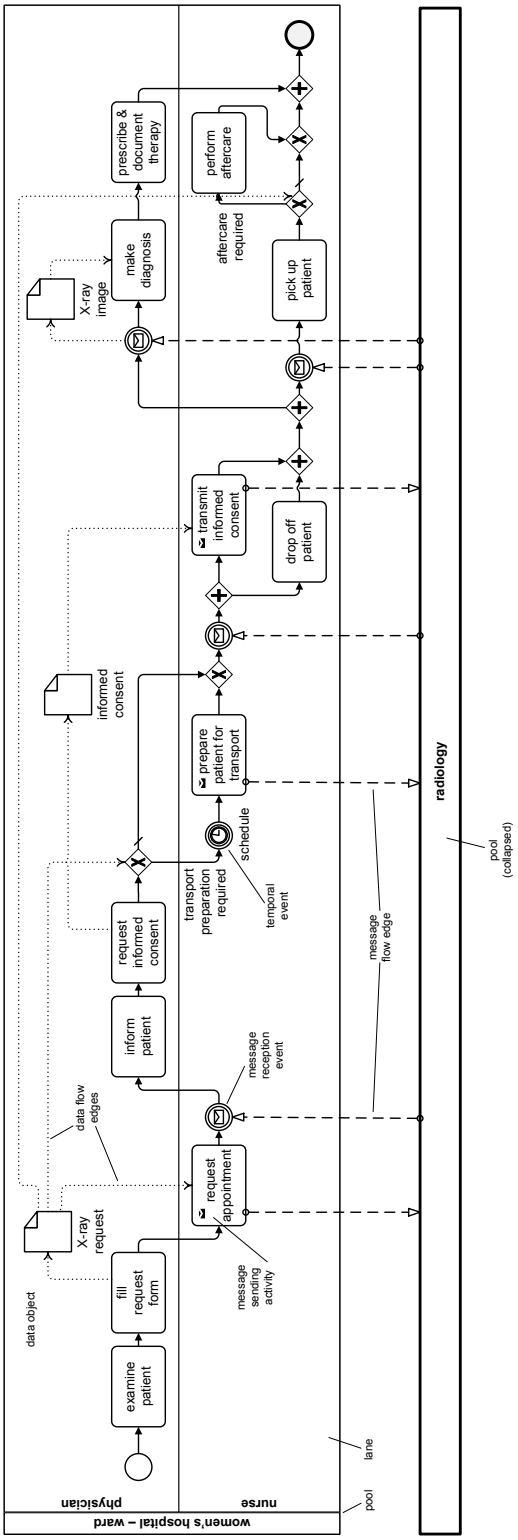


Figure 2.6: Process model covering the control flow, resource, time, data, and interaction perspectives (cf. Example 2.2)

Table 2.6: Event log of the X-ray examination process instance (cf. Fig. 2.2) covering the control flow, resource, time, data, and interaction perspectives.

#	date	time	type	id	activity / data
1	3.5.11	11:25	start	1	examine patient (Mrs. E - physician)
2	3.5.11	11:35	end	1	examine patient
3	3.5.11	11:35	start	2	fill request form (Mrs. E - physician)
4	3.5.11	11:38	write	2	X-ray request (id=27051)
5	3.5.11	11:39	end	2	fill request form
6	3.5.11	11:39	send	3	request appointment (to radiology)
7	3.5.11	11:39	read	3	X-ray request (id=27051)
8	3.5.11	11:39	end	3	request appointment
9	3.5.11	13:10	receive	4	appointment (from radiology)
10	3.5.11	13:11	end	4	appointment
11	3.5.11	13:10	start	5	inform patient (Mrs. E - physician)
12	3.5.11	13:14	end	5	inform patient
13	3.5.11	13:15	start	6	request informed consent (Mrs. E - physician)
14	3.5.11	13:19	write	6	informed consent (id=27091)
15	3.5.11	13:20	end	6	request informed consent
16	5.5.11	09:28	receive	7	call for patient (from radiology)
17	5.5.11	09:28	end	7	call for patient
18	3.5.11	13:14	start	7	drop off patient (Mr. H - nurse)

Table 2.7: Activity log of the X-ray examination process instance (cf. Fig. 2.2) covering the control flow, resource, time, data, and interaction perspectives.

#	date	duration	activity	data
1	3.5.11	11:25-11:35	examine patient (Mrs. E - physician)	
2	3.5.11	11:35-11:39	fill request form (Mrs. E - physician)	*out*: X-ray request (id=27051)
3	3.5.11	11:39-11:39	request appointment (to radiology)	*in*: X-ray request (id=27051)
4	3.5.11	13:10-13:11	appointment (from radiology)	
5	3.5.11	13:10-13:14	inform patient (Mrs. E - physician)	
6	3.5.11	13:15-13:20	request informed consent (Mrs. E - physician)	*out*: informed consent (id=27091)
7	5.5.11	09:28-09:28	call for patient (from radiology)	
8	3.5.11	13:14-...	drop off patient (Mr. H - nurse)	

2.2 Cross-Organizational Processes

Usually, the manufacturing of complex products and the provision of sophisticated services require the collaboration of multiple enterprises or organizational units [MHHR06]. In particular, the enactment of business processes is not necessarily restricted to one enterprise, but may cross organizational borders, i.e., it may involve multiple autonomous partners collaborating with each other to achieve of a common business goal [FIRMR15]. Note that this applies to the healthcare scenario from Example 2.2 as well. In turn, Example 2.8 provides the view of the Radiology Department on the overall process.

Example 2.8 (X-ray examination process for inpatients)
From the viewpoint of the Radiology Department, the process starts with the receipt of an X-ray request form of the Women's Hospital. Then, the secretary schedules the examination and informs the Women's Hospital accordingly.

When the appointment approaches, the secretary notifies the Women's Hospital. As described in Example 2.2, the Women's Hospital sends the patient and signed informed consent. When the patient arrives at the Radiology Department, she is admitted by the secretary. The informed consent, in turn, is checked by a radiology assistant. Subsequently, the assistant prepares the patient for the X-ray examination, which is then performed by a radiologist. Afterwards the Women's Hospital is informed about the completion of the examination by the secretary, who archives the created X-ray images before transmitting a digital copy to the Women's Hospital.

Fig. 2.7 shows the inpatient X-ray examination process from the viewpoint of both the Women's Hospital and the Radiology Department (cf. Examples 2.2 and 2.8).

2.2.1 Collaboration Models

The synthesis of the process models of collaborating partners results in a *collaboration model*. Fig. 2.7 depicts such a collaboration model that comprising the processes of the two partners involved in the X-ray examination process. In general, collaborating partners solely provide *public models* (i.e., views) of their processes, which hide private aspects from the public.

Example 2.9 (Collaboration model)
Fig. 2.7 shows the public models of the inpatient X-ray examination process for both the Women's Hospital and the Radiology Department. As opposed to the collaboration model depicted in Fig. 2.7, the public model of the Women's Hospital abstracts from activities `examination`, `inform patient`, `drop off patient`, `prescribe & document therapy`, and `perform aftercare`.

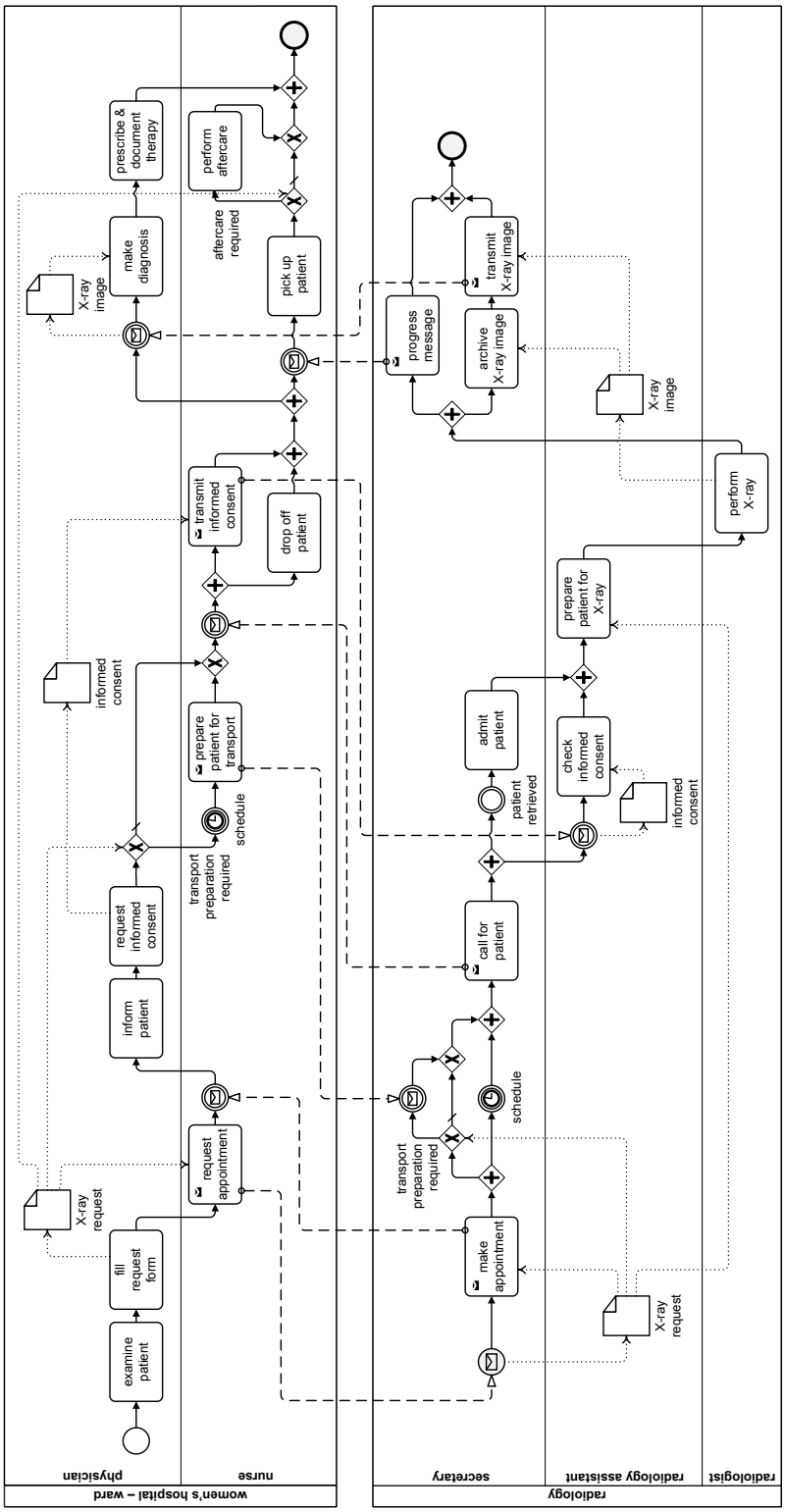


Figure 2.7: Collaboration model of the X-ray examination process

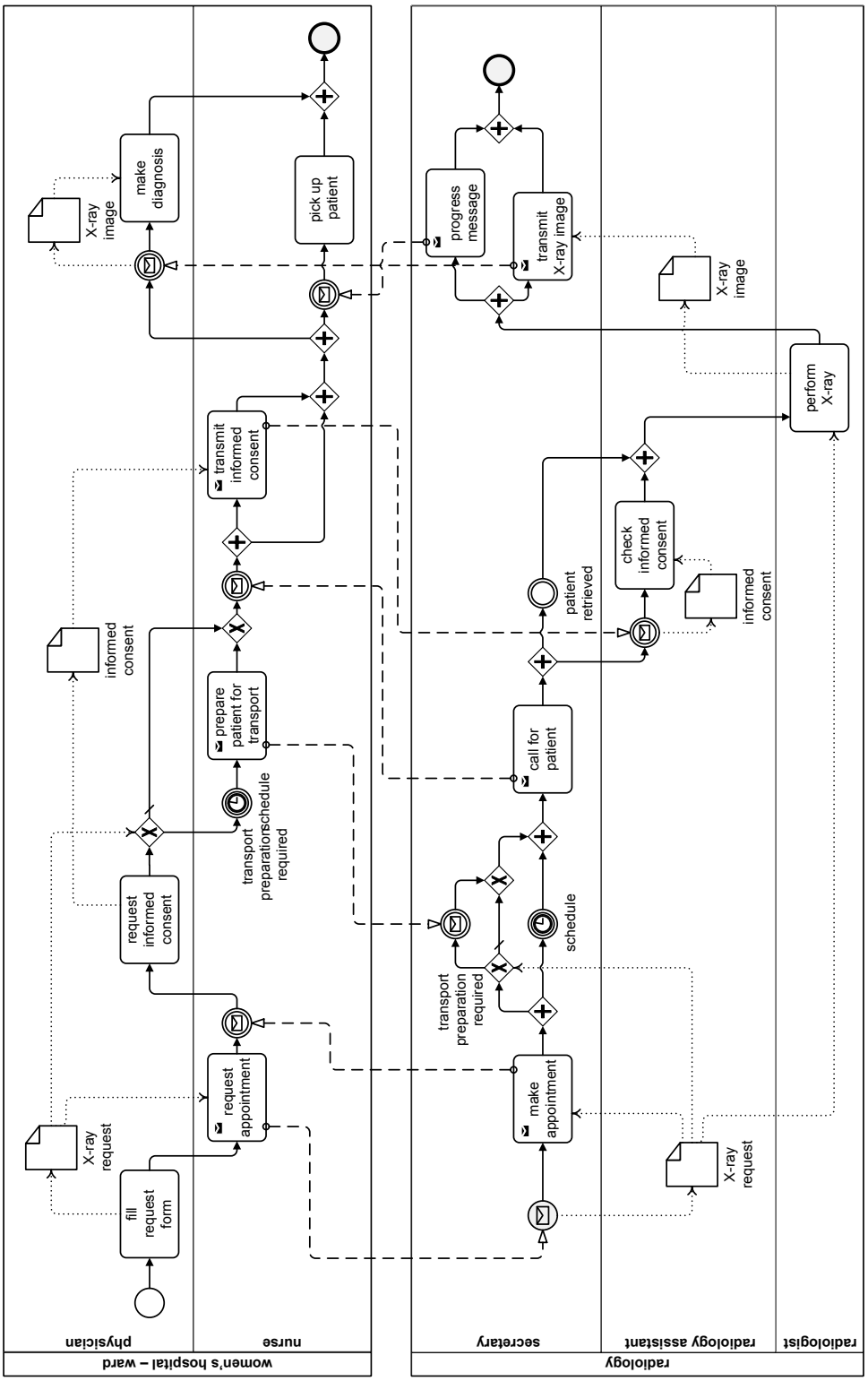


Figure 2.8: Collaboration model showing the public models of the X-ray examination process

2.2.2 Interaction Models

Interaction models provide an abstract view on cross-organizational business processes. In particular, an interaction model abstracts from activities focusing on the messages exchanged instead. Moreover, interaction models specify the control flow for coordinating the interactions between partners through the exchange of *messages*.

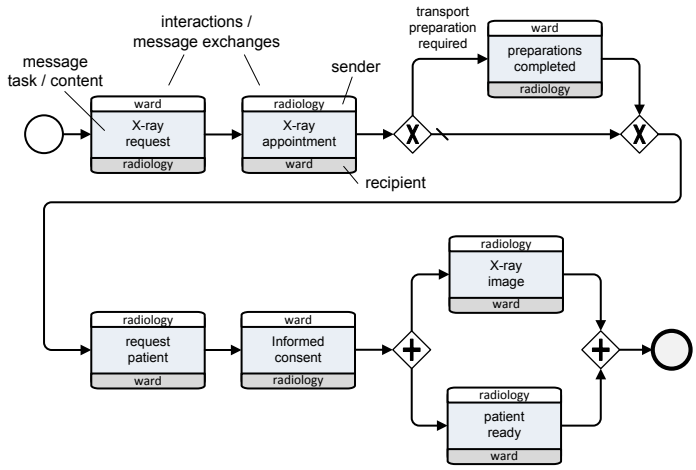


Figure 2.9: Interaction model for the X-ray examination process

2.2.3 Modeling Cross-organizational Processes

Cross-organizational processes may be specified as a meta-process, which starts with the collaborative specification of the interaction model. Then, the partners describe their public models and, finally, they specify and implement their private ones. All partners must agree on the interaction model and all public models are shared among the partners and approved by them. In turn, private models are specified by each partner autonomously (cf. Fig. 2.10).

Figure 2.10: Meta process of modeling cross-organizational processes

2.3 Correctness Criteria for Business Process Models

Like other kinds of dynamic models, business process models are subject to correctness criteria. The latter can be assigned to three different layers, which build on each other as illustrated in Fig. 2.11.

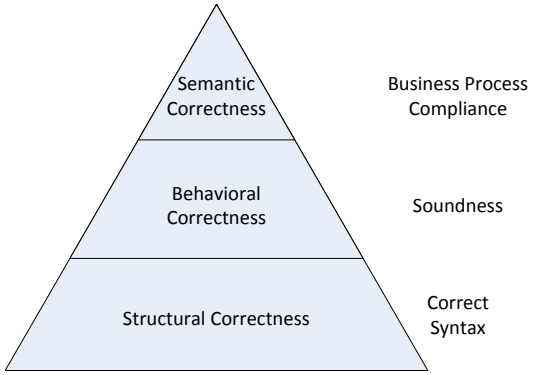


Figure 2.11: Layers of process model correctness [KRM+13]

2.3.1 Structural Correctness

The first fundamental layer of process model correctness refers to *structural model correctness*. A process model is considered as being structurally correct if it does not violate the syntax rules of the process meta model used for its description. Amongst others, this means that the various model elements are used according to the rules of the meta model. For example, start nodes must not have any incoming sequence flows and end nodes must not have any outgoing sequence flows. Structural correctness can be easily verified by checking the existence or absence of required and forbidden relations. Note that the structural correctness of a process model constitutes a prerequisite for its execution.

Example 2.10 (Structural Correctness)
Fig. 2.12 shows a process model that contains a structural error. In particular, the end event is followed by activity `B`, i.e., the end event has an illegal outgoing sequence flow edge. In addition, activity `B` requires an outgoing sequence flow, which is missing.

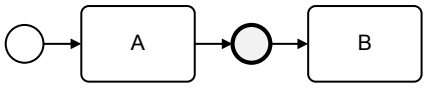


Figure 2.12: Structurally incorrect process model

For cross-organizational business processes, structural correctness additionally necessitates *structural compatibility* [DW07]. The latter, in turn, requires from the processes of a collaboration to use the same messages, i.e., if the process of a partner sends a certain message, the process of the partner receiving the message must comprise a corresponding receipt event and vice versa.

Example 2.11 (Structural Compatibility)
Fig. 2.13 shows two process models which are not structurally compatible. The upper process model sends message `M1`, while the lower model is waiting solely for message `M2`.

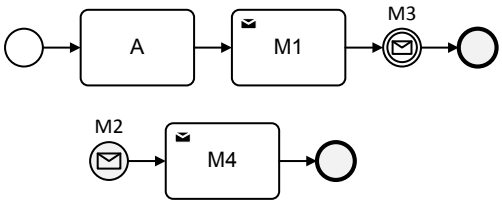


Figure 2.13: Structurally incompatible process models

2.3.2 Behavioral Correctness

In [RW12], *process model soundness* (i.e., behavioral correctness) refers to the three basic correctness criteria *option to complete*, *proper completion*, and *absence of dead activities*. A process model has the *option to complete* if each running instance, not being in a final state yet, can still be completed, i.e., neither *deadlocks* nor *livelocks* have occurred. In addition, once a final state is reached, there must be no activity in state *running* or *enabled*, i.e. the process completes properly. Finally, each activity is part of any valid possible execution of the process model, i.e. there are *no dead activities*.

Example 2.12 (Deadlock)
An example of a process model with a deadlock is provided by Fig. 2.14. No instances of the depicted process model can complete. Note that the *AND-join gateway* waits for both incoming sequence flows, whereas the preceding *XOR-split gateway* only triggers one of the two sequence flows.

Example 2.13 (Livelock)
Fig. 2.15 shows an example of a process model with a livelock. If the *XOR-split gateway* selects the upper sequence flow, the process will never exit the loop block containing activity `B`.

Example 2.14 (Proper completion)
Fig. 2.16 shows a process model whose instances will not properly complete as they may reach the end node even though there may be remaining tokens. In detail, when reaching the *AND-split gateway*, both `B` and `C` become enabled. As soon as one of the two activities completes, the end node is reached, even though the other activity is still enabled or running.

Example 2.15 (Dead activities)
The problem of *dead activities* is illustrated by Fig. 2.17. In this process model, there exists no possible execution path that includes activity `B`.

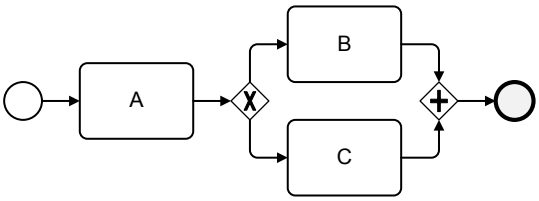


Figure 2.14: Process model with deadlock

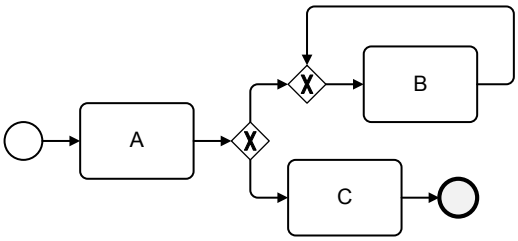


Figure 2.15: Process model with potential livelock

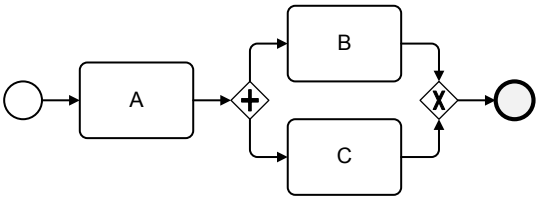


Figure 2.16: Process model with improper completion

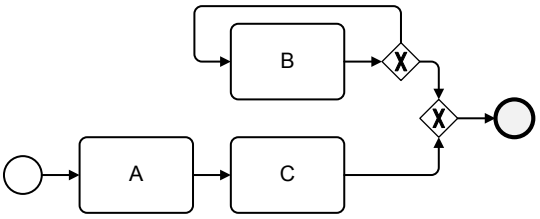


Figure 2.17: Process model with dead activity

Regarding cross-organizational processes, additional criteria need to be met in order to ensure soundness of the overall process.

First, *behavioral compatibility* between the process models of a process collaboration shall ensure that the collaboration is sound again [AW01, DW07].

Example 2.16 (Behavioral Compatibility)
Fig. 2.18 shows an example of two incompatible process models. Though the individual process models of the two partners are structurally compatible and sound, their

combination results in a deadlock, i.e., the collaborative process is not sound. After *Partner 1* sends message `M1`, either message `M2` or `M3` is sent by *Partner 2*. However, the process of *Partner 1* cannot complete as it waits for both messages `M2` and `M3`, i.e., the collaboration ends in a deadlock.

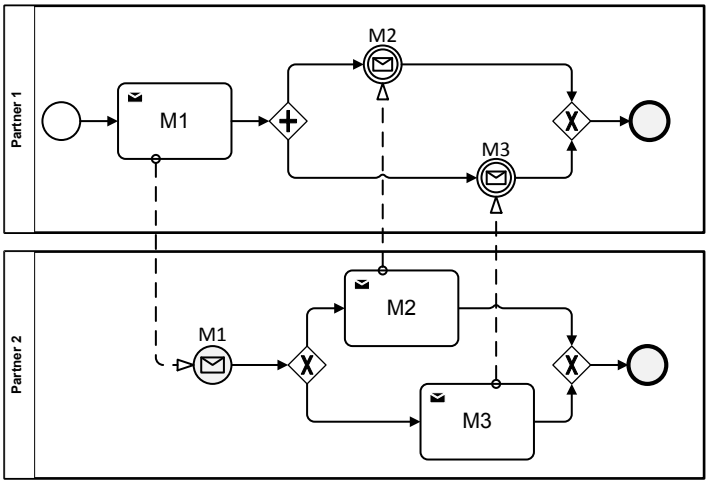


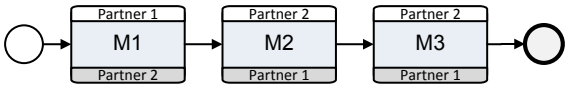
Figure 2.18: Behaviorally incompatible process models

The *conformance* of a public or private process model with an interaction model ensures that the process implements the behavior of a certain partner as required by the interaction model. Accordingly, *conformance* of a private process with the corresponding public process ensures that the private model implements the behavior of the public model [DW07].

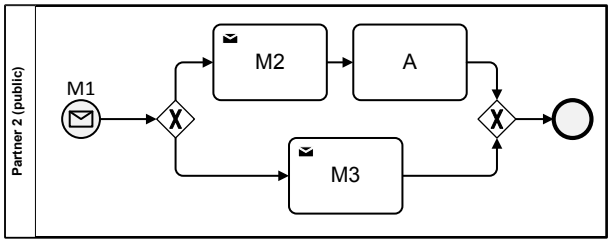
Example 2.17 (Conformance)
In Fig. 2.19, neither the public process model (cf. Fig. 2.19b) conforms to the interaction model (cf. Fig. 2.19a), nor does the private process model (cf. Fig. 2.19c) conform to the public process model or the interaction model. In particular, the interaction model requires from *Partner 2* to send *M2* and `M3` after receiving `M1`. In turn, the public and private models of *Partner 2* either send message `M2` or `M3`. Furthermore, the public model of *Partner 2* requires the execution of activity `A` after sending message `M2`. However, the private model executes activity `Q` instead of activity `A`.

Realizability shall ensure that an interaction model can be realized by a collaboration of partner processes, i.e., there exists a collaboration of partner process models, which interact exactly as described by the interaction model.

(a) Interaction model



(b) Public process model



(c) Private process model

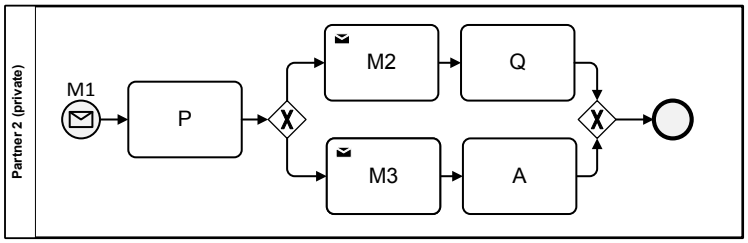


Figure 2.19: Non-conformant process models

Example 2.18 (Realizability)
Fig. 2.20 shows a non-realizable interaction model. No process model can be constructed for *Partner 3* as this model must not send `M3` before `M2` was sent from *Partner 2* to *Partner 1*. However, *Partner 3* and its model are not notified about the exchange of `M2`. Thus, *Partner 3* cannot determine when to send `M3`.

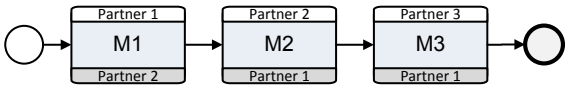


Figure 2.20: Non-realizable interaction model

2.3.3 Verifying Cross-organizational Processes

The process of modeling cross-organizational processes (cf. Fig. 2.10) can be enhanced by adding steps for verifying the presented criteria. Fig. 2.21 illustrates when soundness, compatibility and realizability are verified.

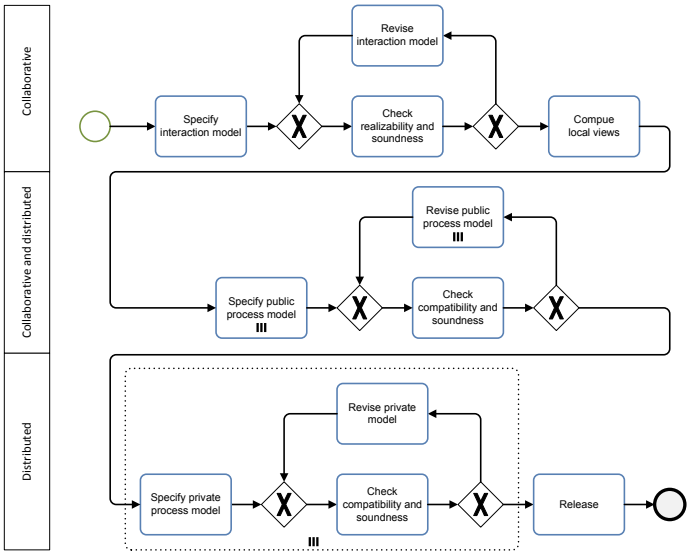


Figure 2.21: The process of modeling cross-organizational processes

When considering the resource, time, data, time, and interaction process perspectives, additional soundness criteria become necessary. For example, the data perspective requires the values of data objects to be written before accessing them [RD97, RD97, RRMD09]. Time constraints, in turn, must not conflict with each other [LRW16, KSB15], and human resource assignment must ensure that for each activity there is at least one resource that is allowed to perform the activity [CRC11].

For the sake of brevity, a detailed presentation of the correctness criteria is omitted. Interested readers are referred to [RD97, RRMD09, CRC11, RW12, KSB15, LRW16].

2.4 Business Process Compliance

Even if a process model is structurally and behaviorally correct, it still might be semantically incorrect, i.e., violate domain-specific *compliance rules*.

Business process compliance summarizes languages, techniques and methods that aim to ensure compliance of business processes with a set of semantic constraints.

Example 2.19 (Compliance rules)
Table 2.8 lists five examples of compliance rules that refer to the X-ray examination process (cf. Examples 2.2 and 2.8).

As opposed to the structural and behavioral correctness criteria presented in Section 2.3, business process compliance not only needs to be ensured for the modeling phase and the

Table 2.8: Compliance rules adopted from [KR17, KSR96b]

C1	Before a physician requests an informed consent, he must inform the patient about risks.
C2	Before the X-ray examination may take place, the informed consent of the patient must be checked by a medical technical assistant (MTA) of the radiology department.
C3	Diagnoses shall be provided by physicians after having received all X-ray images from the radiology department; i.e., no X-ray image should be received afterwards.
C4	After an X-ray examination, X-ray images must be archived and then be forwarded to the requesting ward.
C5	A patient shall be formally admitted within one week after referral to the hospital.

artifacts (i.e. process models) it produces, but also for the other phases of the process life cycle, i.e. modeling, execution, change, and analysis.

The following section discusses business process compliance along the process life cycle according to [LRD08, KR11a].

2.4.1 Modeling Compliance Rules

Usually, guidelines, standards and regulations capture compliance rules in a narrative and informal manner (cf. Table 2.8). To enable any computer-aided checking of business process compliance, however, compliance rules need to be specified in a machine-interpretable way. As compliance rules are domain-specific, their specification should not be solely the responsibility of IT experts, but involve domain experts as well.

Early approaches on business process compliance suggested the use of logic-based languages for specifying compliance rules [GMS06, GK07, LMX07, SGN07]. An example of such a language is Linear Temporal Logic (LTL) [Pnu77], which enhances ordinary propositional logic with additional temporal operators (X next, F eventually, G globally, U until, W weakly until) that allow navigating from point to point on a discrete time line.

Example 2.20 (LTL)
Table 2.9 illustrates the specification of the five compliance rules from Table 2.8 in terms of LTL.

Table 2.9: Compliance rules expressed in LTL

#	LTL expression
C1	$\neg$ `request informed consent` **W** `inform patient`
C2	$\neg$ `perform X-ray` **W** `check informed consent`
C3	**G**(`make diagnosis` $\rightarrow$ $\neg$**F** `receive X-ray image`)
C4	**G**(`perform X-ray` $\rightarrow$ **F**(`archive X-ray image` $\wedge$ **F** `transmit X-ray image`))
C5	**G**(`refer patient` $\rightarrow$ **F** `admit patient`)

Logic languages, however, are complex to use and error-prone. Therefore, they are not appropriate for domain experts, who will have difficulties in comprehending logic-based expressions [TEvP12, LRMD10, AWW11, CTZ+16]. To address this issue, pattern-based and visual languages have been proposed as alternative approaches for specifying compliance rules.

Inspired by the pattern-based specification of properties for finite-state systems in [DAC99], pattern-based compliance rule languages were introduced [TEvP12, RFA12, ZF15]. These languages were developed based on insights gained from the analysis of regulations, corporate standards, process model collections, and related works. Pattern-based compliance rule languages rely on pre-specified *compliance patterns*. The latter, in turn, cover formal details (e.g., expressions in LTL) behind descriptive labels as well as textual descriptions. Furthermore, they use placeholders to abstract from specific activities. Thereby, the individual patterns were discovered through extensive analyses of large sets of real-world compliance rules.

For example, compliance pattern "*X LeadsTo Y*" from [TEvP12] is described as "X must be followed by Y", hiding LTL expression $\mathbf{G}(X \rightarrow \mathbf{F}\, Y)$. Note that there are pattern-based approaches (e.g. [TEvP12]) that support the combination of compliance patterns through logic operators, but do not allow for their nesting.

Selected examples of compliance patterns are shown in Table 2.10.

Table 2.10: Compliance patterns in the style of [TEvP12]

pattern	**description**
X Exists	X must exist in the process specification
X Absent	X must not be present in the process specification
X Precedes Y	X must precede Y
X LeadsTo Y	Y must follow X
X NegLeadsTo[1] *Y*	Y must not follow X
$<X_1,\ldots,X_n>$ *ChainLeadsTo* $<Y_1,\ldots,Y_m>$	A sequence of $Y_1,\ldots,Y_m$ must follow a sequence of $X_1,\ldots,X_n$

[1] Pattern *NegLeadsTo* is not included in [TEvP12], but occurs in other sets of compliance patterns, e.g., as pattern *Response Negative* in [RFA12]. It is noteworthy that *NegLeadsTo* is not equivalent to the logic negation of *LeadsTo* Y.

Example 2.21 (Compliance patterns)
Table 2.11 illustrates the pattern-based specification of the five compliance rules from Table 2.8 using the compliance patterns from Table 2.10.

As a major drawback, pattern-based approaches are restricted to a set of pre-specified compliance patterns and, hence, need to be adapted each time a new pattern is discovered. For example, the pattern set suggested by [TEvP12] is unable to express compliance rule C3 (cf. Table 2.8) as it does not contain the *NegLeadsTo* pattern. Note that the

Table 2.11: Compliance rules from Table 2.8 expressed with compliance patterns

#	compliance pattern
C1	`inform patient` *Precedes* `request informed consent`
C2	`check informed consent` *Precedes* `perform X-ray`
C3	`make diagnosis` *NegLeadsTo* `receive X-ray image`
C4	`perform X-ray` *ChainLeadsTo* `archive X-ray image, transmit X-ray image`
C5	`refer patient` *LeadsTo* `admit patient`

latter is not equivalent to the logic negation of *LeadsTo*. In turn, more extensive sets of compliance patterns (e.g., [RFA12]) tend to become too large and complex as they contain numerous patterns that are strongly related, but only differ in minor details. This makes it challenging for domain experts to select the appropriate patterns [RFA14].

Visual compliance rule languages, in turn, use a graph-based approach for specifying compliance rules. For example, BPMN-Q [ADW08, AWW11] is a visual language for specifying and verifying compliance rules. BPMN-Q uses annotated *path edges* that can be placed between start nodes, end nodes, and activity nodes to express and visualize compliance patterns. Finally, labels on path edges express the absence of activities within a certain scope (e.g., the interval between two activities).

Example 2.22 (BPMN-Q)
Fig. 2.22 shows the compliance rules from Table 2.8 expressed with BPMN-Q. Note that C4 is not considered as BPMN-Q does not support the *ChainLeadsTo* pattern (cf. Fig. 2.22).

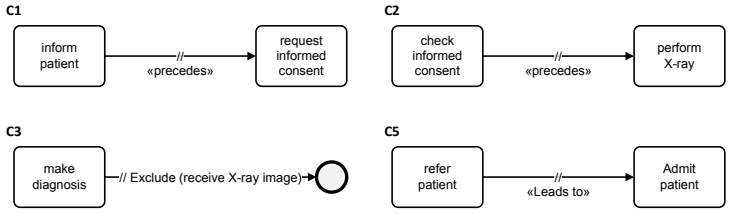
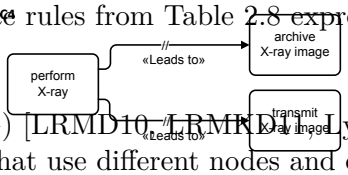


Figure 2.22: Compliance rules from Table 2.8 expressed with BPMN-Q



Compliance Rule Graphs (CRG) [LRMD10, LRMK11, Ly13] provide a special kind of non-deterministic automatons that use different nodes and edges to define the scope and requirements of a compliance rule. In particular, angular *antecedence* activity nodes are used to express when a compliance rule shall be triggered, whereas rounded *consequence* activity nodes specify the actions required or forbidden by the rule. Edges between the nodes, in turn, restrict the order in which activities may occur. As opposed to BPMN-Q, CRGs do not build on any set of compliance patterns.

Example 2.23 (CRG)
Fig 2.23 shows the compliance rules from Table 2.8 expressed as CRGs.

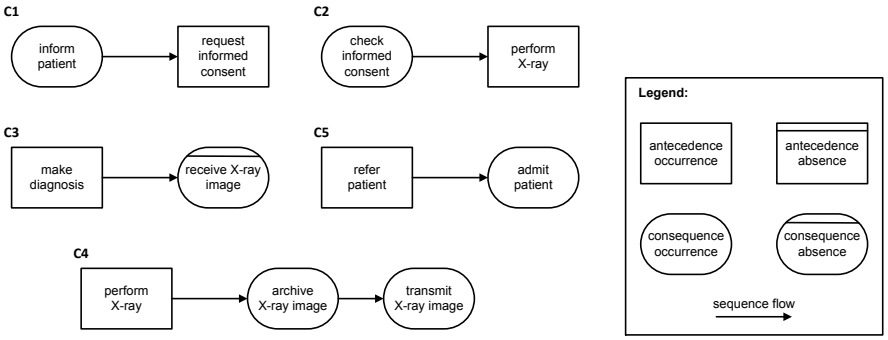


Figure 2.23: Compliance rules from Table 2.8 expressed as CRGs

Note that some logic languages and pattern-based approaches for modeling compliance rules support the control flow, data, time, and resource perspectives. The visual CRG language, however, focuses on the control flow perspective solely, whereas BPMN-Q covers some aspects of the data perspective as well.

2.4.2 A-priori Compliance Checking

Based on formally specified compliance rules, the conformance of existing process models with these rules can be verified. Such verification is denoted as *a-priori compliance checking* as accomplished prior to the creation and execution of corresponding process instances. A process model *complies with* a given compliance rule, if and only if the model solely allows for traces satisfying the rule [KR11a]. In turn, a process model *violates* a compliance rule if it allows for at least one execution (i.e., trace) for which the compliance rule does not apply.

Example 2.24 (A-priori compliance checking)
Consider the process model from Fig 2.24. It meets compliance rule *A LeadsTo B*, but violates compliance rule *C Precedes B*. Note that the model only allows for traces $\sigma_1 =< A, B, C >$ and $\sigma_2 =< A, C, B >$. In particular, `A` is always executed prior to `B` in both traces, i.e., *A LeadsTo B* holds. In turn, `C` is only preceding `B` in σ_2, but not in σ_1. Consequently, rule *C Precedes B* is violated by the process model.

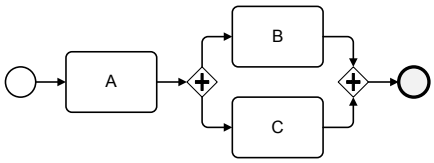


Figure 2.24: A-priori compliance checking

For a limited set of compliance patterns, there exist algorithms enabling process compliance verification by analyzing the structure of the respective process models [LRD08, LRMGD12]. In general, a-priori compliance checking is accomplished through model checking [HR04], i.e., the exploration of the state space of a process model. A well-known problem in this context is *state space explosion*, which describes the fact that even small models might have a huge state space that cannot be fully explored anymore. As this problem also occurs in the context of a-priori compliance checking, [LMX07, ADW08, KLRM+10] provide contributions mitigating the state space explosion problem.

2.4.3 Run-time Compliance Monitoring

Sometimes it is not possible to verify process compliance *a priori*. This applies, for example, if the state space of a process model becomes too large to be explored or the process is only loosely specified at design time, i.e., the process dynamically evolves during run time [RW12, MGKR15].
Compliance monitoring [AHW+11, LRMKD11] allows controlling and monitoring the status of compliance rules during the execution of process instances. Note that compliance rules may not only have final *compliance states* (e.g., *satisfied* or *violated*), but intermediary states as well (e.g., *pending* or *violable*). State *pending*, for example, expresses that the compliance rule is violated by the considered process instance in its current state, but the instance may still be continued in a way that restores conformance with the compliance rule. In turn, state *violable* expresses that the compliance rule is not violated yet, but it is still possible to continue the process instance in a way violating the rule.

If a compliance rule is triggered more than once, the different *instances* or *activations* of the rule need to be monitored. For example, a new instance of compliance rule *A LeadsTo B* becomes *activated* (i.e., triggered) at every occurrence of *A*. In turn, this means that one instance of a compliance rule may be *permanently satisfied*, whereas another instance of this rule is *pending* or even *permanently violated* at the same time.

Example 2.25 (Compliance monitoring)
To illustrate the various compliance states, we consider the partial activity log from Table 2.12. When considering compliance rules *B LeadsTo C*, *E NegLeadsTo A*, *E Precedes F*, *D Precedes E*, and *D LeadsTo G*, one can easily see that the two instances of the first

rule (*B LeadsTo C*) become activated by entries #2 and #4. In turn, *E NegLeadsTo A*, *E Precedes F*, and *D Precedes E* become activated once, whereas *D LeadsTo G* has not been activated yet.

Table 2.12: Partial activity log for compliance monitoring

#	activity
1	A
2	B
3	C
4	B
5	E
6	F

compliance rule	activation	compliance state
B LeadsTo C	(2|B)	permanently satisfied
	(4|B)	pending
E NegLeadsTo A	(5|E)	violable
E Precedes F	(6|F)	permanently satisfied
D Precedes E	(5|E)	permanently violated
D LeadsTo G	-	(not activated)

2.4.4 A-Posteriori Compliance Checking

Even for completed process instances, their conformance with a set of compliance rules can be subject of interest, e.g., to support audits or to check the conformance of completed instances with newly emerging compliance rules. Verifying compliance for completed process instances is summarized under the term *a-posteriori compliance checking*. Accordingly, the state of a compliance rule is either *satisfied* or *violated*. However, similar to compliance monitoring, different activations of the same rule may have to be distinguished.

Example 2.26 (A-posteriori compliance checking)
Table 2.13 completes the partial process log from Table 2.12. Hence, the states of the related compliance rules are fixed and must not be changed anymore. In particular, completing the process instance affects both compliance rules *B LeadsTo C* and *E NegLeadsTo A*. The second activation of *B LeadsTo C* becomes *violated* as activity `C` was not executed at all. Finally, the activation of *E NegLeadsTo A* becomes violated as `A` has occurred after `E`.

Table 2.13: Completed activity log for a-posteriori compliance checking

#	activity
1	A
2	B
3	C
4	B
5	E
6	F
7	A
8	G

compliance rule	activation	compliance state
B LeadsTo C	(2|B)	(satisfied)
	(4|B)	violated
E NegLeadsTo A	(5|E)	violated
E Precedes F	(6|F)	satisfied
D Precedes E	(5|E)	violated
D LeadsTo G	-	(not activated)

2.4.5 Effects of Process Changes on Process Compliance

Process models may be subject to changes [RHD98, WSR09, RRMD09]. In turn, this might affect the compliance of the models with existing compliance rules. Accordingly, changes of process models need to be classified with respect to their effects on the conformance with a given compliance rule. When re-checking compliance of a changed model, the change is considered to have *positive* effects if a previously violated compliance rule is met after applying the change. In turn, the change has *negative* effects if it causes a violation of a compliance rule that was satisfied before. Finally, the change has *no* or *neutral* effect if it does not affect the conformance or non-conformance of a process model with the considered compliance rule.

Example 2.27 (Change effects on compliance)
Fig. 2.25 illustrates a change of the process model depicted in Fig. 2.24. Before changing the process model, it conformed with compliance rule *A LeadsTo B*, but violated rule *C Precedes B* (cf. Section 2.4.2). However, after removing *B* the model solely allows for trace $\sigma' =< A, C >$. Consequently, *A LeadsTo B* is then violated. In turn, rule *C Precedes B* is no longer violated after changing the model, as the latter does not contain `B` anymore. Consequently, the change has a negative effect on the former rule, but a positive one on the latter. Regarding rule *A LeadsTo C*, the change has no effect, i.e., the process model conforms to this rule before and after removing `B`.

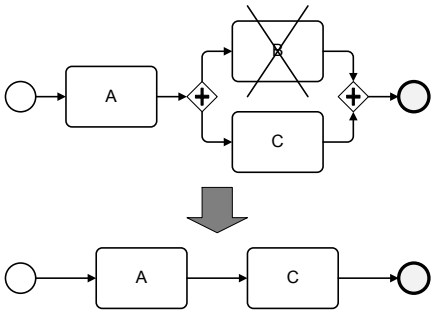


Figure 2.25: Effects of process model changes on process compliance

As not every change affects any compliance rule, change profiles can be used for estimating the effects of a change [LRD08]. In particular, such estimations enable us to determine a subset of compliance rules not affected by the change and, hence, allow limiting the number of compliance rules to be rechecked.

3

Modeling Multi-Perspective Business Process Compliance Rules

This chapter is based on the following publications:

[KRL+13b]: **D. Knuplesch**, M. Reichert, L. T. Ly, A. Kumar, and S. Rinderle-Ma. Visual modeling of business process compliance rules with the support of multiple perspectives. In *32nd International Conference on Conceptual Modeling (ER'13)*, volume 8217 of *LNCS*, pages 106–120. Springer, 2013

[KR17]: **D. Knuplesch** and M. Reichert. A visual language for modeling multiple perspectives of business process compliance rules. *Software and Systems Modeling*, 16(3), pages 715–736, 2017

The original articles can be found in Appendices 4 and 8, respectively.

Additional details on selected aspects have been published in [SKR14]. Finally, formal backgrounds and details of the evaluation were published in a technical report [KRL+13a].

3.1 Research Challenges

As discussed in Chapter 2, compliance rules constitute semantic constraints on business processes restricting both the order and the occurrence of activities. Potential sources of compliance rules range from regulations and legislative documents to internal quality guidelines and standards. These sources have in common that they describe compliance rules in a narrative style, i.e., a non-formal way introducing ambiguities on one hand, but being easy to understand and use by *domain experts*, who have to interpret the rules, on the other. In general, any verification of business process compliance, which is usually accomplished by IT experts, requires a formal and unambiguous specification of the rules to be checked. Ideally, the formalism used for specifying compliance rules is comprehensible to domain experts as well.

Obviously, the expressive power of compliance rule specification languages should cover compliance rule patterns. Like process models, compliance rules must not be restricted

to the control flow perspective, but cover the resource, time, data, and interaction perspectives as well. As a consequence, languages for specifying compliance rules should consider all these perspectives.

Example 3.1 (Compliance rules)
Table 2.8 (cf. Section 2.4) shows examples of multi-perspective compliance rules. Besides the control flow perspective, the resource, time, data, and interaction perspectives are referred as well. For example, C2 covers the *resource perspective* by referring to role *medical technical assistant* and organizational unit *radiology department*. In C1, the binding of duties constraint deals with the *resource perspective* as well. The maximum time distance mentioned in C5 refers to the *time perspective*. Rule C4, in turn, refers to *X-ray images*, i.e., data objects related to the *data perspective*. Finally, C3 and C4 refer to the exchange of messages between different units, i.e., these two rules refer to the *interaction perspective*.

In general any language for specifying compliance rules faces the following challenges:

1. The language should cover characteristic compliance rule patterns. In particular, it should not only consider the control flow perspective, but support the resource, time, data, and interaction perspectives as well.

2. The language should be easy to understand by both domain and IT experts. Note that both user groups are involved in the verification of business process compliance.

3. The language should be equipped with a non-ambiguous and precise semantics to enable its computer-aided verification.

As discussed in Section 2.4.1, several languages have been suggested to formally specify compliance rules, e.g., Linear Temporal Logic (LTL) [GK07] or Formal Contract Language (FCL) [SGN07, GHSW09]. As formal languages are difficult to understand for domain experts and business analysts, pattern- and graph-based approaches were introduced.

Visual notations for modeling business processes and related constraints, however, have shown advantages compared to text-based specifications [OFR+12, HZ14]. In particular, visual notations enable an improved comprehensibility, presuming preceding training, and foster the communication between the various stakeholders.

Most existing visual approaches for modeling compliance rules, however, focus on the control flow perspective solely (cf. Section 2.4). BPMN-Q [ADW08, AWW09] additionally covers data conditions (i.e., an aspect of the data perspective), but does not consider the resource, time and interaction perspectives.

3.2 Scientific Contribution

[KRL+13b] and [KR17] have introduced *extended Compliance Rule Graphs* (eCRGs), i.e, the visual language developed in the context of this thesis for specifying process compliance rules covering the control flow, resource, time, data, and interaction perspectives. On one hand, the eCRG language is expressive as it covers well-known compliance patterns [RFA12, TEvP12]. On the other, eCRGs are understandable to both domain and IT experts. Finally, the eCRG language has a well-defined semantics, which is based on the transformation of eCRGs into FOL expressions.

The requirements for the design of eCRGs were elicited in case studies and literature reviews [Ngu13, KRM+13, RFWM12]. Fig. 3.1 depicts the meta model of the eCRG language concepts. On the left hand side, the entities of the resource perspective as well as their semantic relations are shown; i.e., *organizational unit*, *group*, *role*, and *staff member* [SKR14]. The control flow perspective, in turn, comprises entities *activity*, *activity type*, and *activity event* (e.g., *start/end activity event*) as well as the generic concept *event*. Note that the control flow entities have been adopted from the meta model originally introduced for CRGs [LRMD10] (cf. Section 2.4). In turn, the time perspective covers *points in time*, whereas the interaction perspective adds entities *receive/send message event*, *message type*, and *business partner* (i.e., a party sending or receiving messages). Finally, the data perspective consists of the entities *data object*, *data container*, *read/write event*, *input/output parameter*, and *execution parameter*. Moreover, it comprises a triple relation specifying the *current value* (i.e., *data object*) of a *data container* at a particular *point in time*.

Taking the concepts of the presented meta model (cf. Fig. 3.1), Fig. 3.2 provides an overview of the eCRG language elements. Their application to the compliance rules listed in Table 2.8 is illustrated by Fig. 3.2.

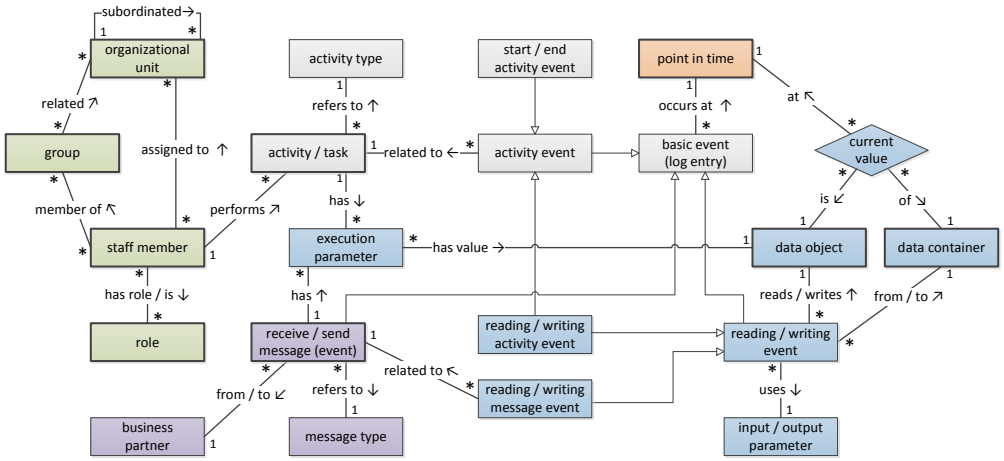


Figure 3.1: Concepts supported by the eCRG language

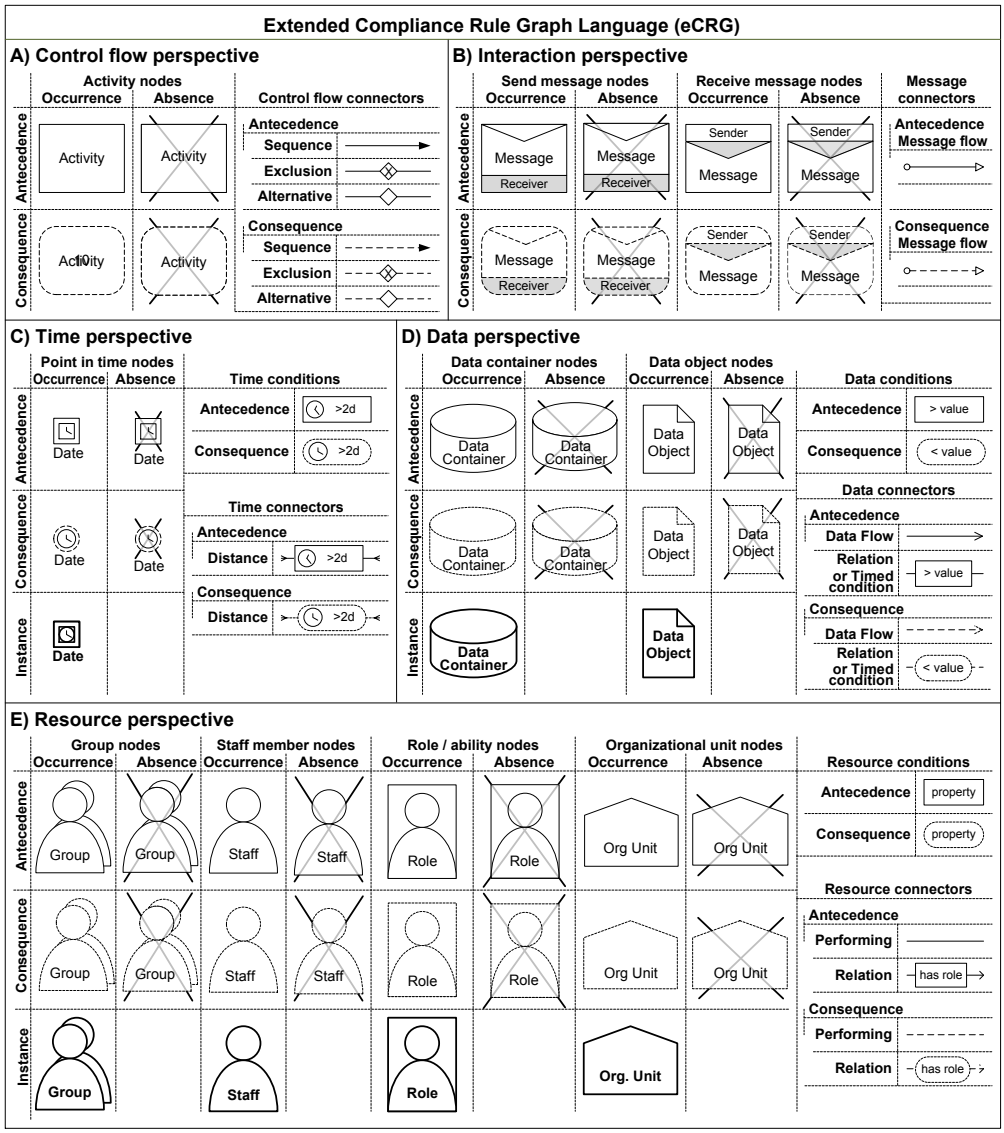


Figure 3.2: Elements of the eCRG language (adopted from [KR17])

An eCRG is composed of *nodes* and *edges* that may be further enriched with *attachments*. Note that nodes either correspond to *events* or *entities*. Events may refer to the start/completion of an activity or the receipt of a message. In turn, data objects and resources constitute examples of entities. Relations between nodes are described by connectors (i.e. edges) referring to semantic relations between events and/or entities. For example, the *sequence flow connector* describes the temporal relations among activities and messages. In turn, the *data flow connector* relates activities and messages with the data objects they access. Finally, the eCRG language allows for *attachments* imposing additional constraints, for example, on the data flow, the time distance between activities,

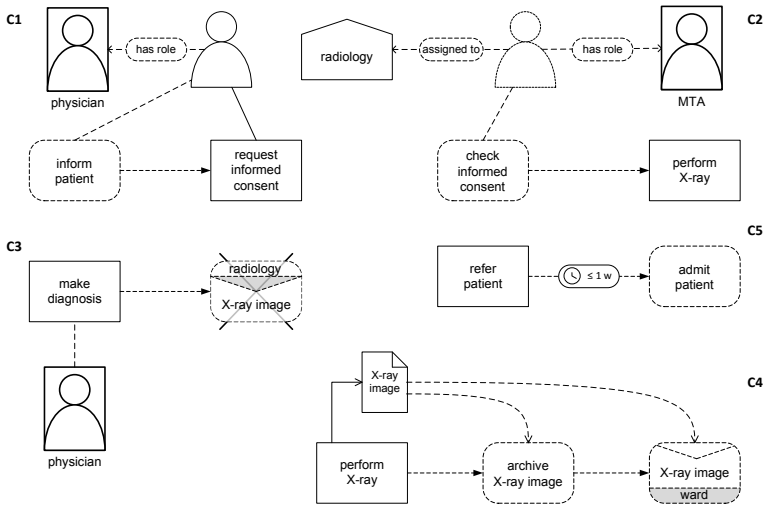


Figure 3.3: Compliance rules from Table 2.8 expressed as eCRGs

the properties of messages, or the capabilities of staff members.

To distinguish between a precondition (i.e. *antecedence*) as well as related postconditions (i.e. *consequences*), different visualizations of the same eCRG elements can be used. In particular, the elements of an eCRG can be partitioned into an *antecedence pattern*, specifying when the compliance rule is triggered (i.e., activated), and at least one *consequence* pattern, specifying what the rule requires. eCRGs may further contain references to real-world objects and entity instances, e.g., a certain organizational unit (e.g. *Ulm University*) or a specific point in time (e.g., *26 October 2013*). Note that corresponding *instance nodes* are neither part of the antecedence nor the consequence patterns.

As compliance rules may require the occurrence or absence of certain events, the antecedence and consequence patterns are sub-divided into an *occurrence* and *absence sub-pattern*.

Following the *principles for designing effective visual notations* [Moo09] and considering the semiotic clarity, perceptual discriminability, semantic transparency, graphic economy, and the cognitive fit, the eCRG language applies different styles for visualizing the mentioned patterns: *solid lines and square shapes* are used for visualizing the antecedence pattern. *Dashed lines and round shapes* are used for visualizing the consequence pattern. *Thick lines and square shapes* are used when drawing entity instances as part of an eCRG. Finally, absence nodes are crossed out by an *oblique cross*.

The partitioning of an eCRG into antecedence and consequence patterns as well as corresponding sub-patterns is illustrated by Fig. 3.4.

The various eCRG patterns constitute the foundation for formulating the eCRG semantics [KRL+13a]. A completed process log complies with a given eCRG, if and only if for

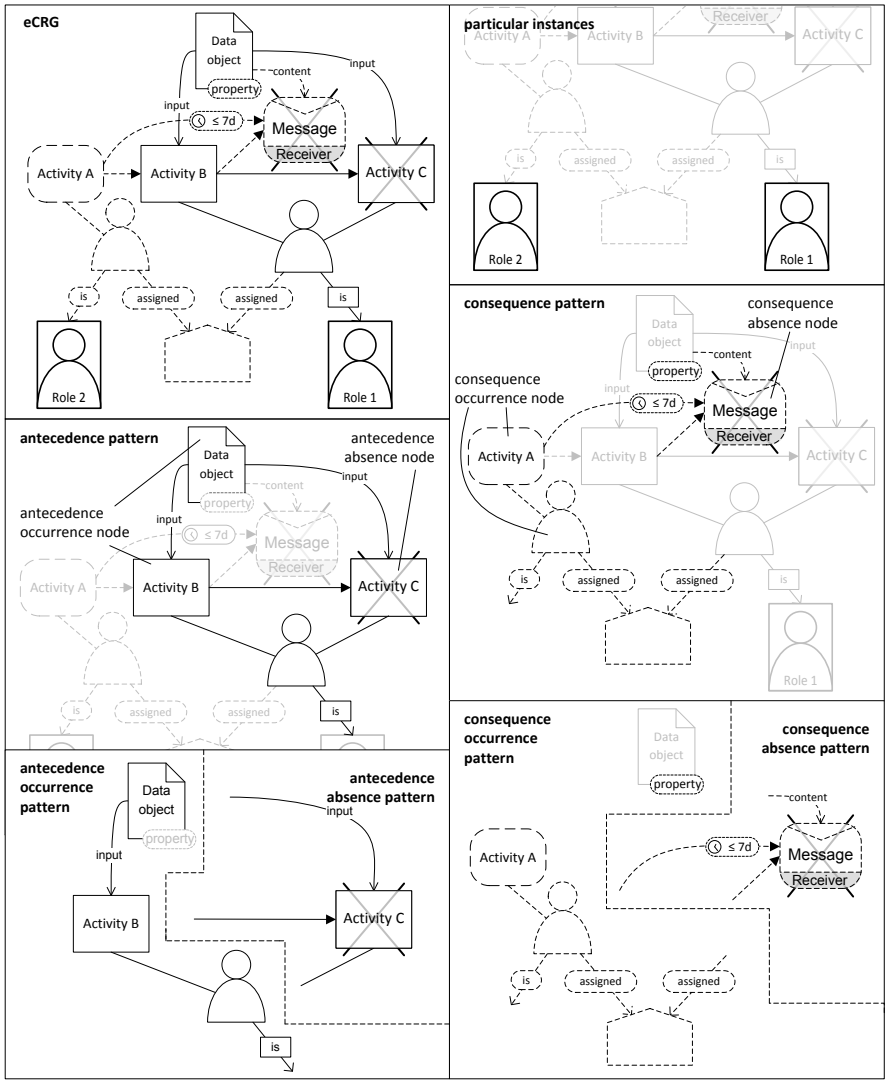


Figure 3.4: Partitioning of an eCRG into separate patterns (adopted from [KR17])

each match of the eCRG antecedence pattern (i.e., satisfaction of the precondition), at least one corresponding match of a consequence pattern of the eCRG can be found (i.e., satisfaction of the postcondition) [KR17]. If no match of the antecedence pattern can be found, the precondition is not met and the process log *trivially complies* with the given rule. A match of a particular antecedence or consequence pattern requires suitable events regarding the nodes of the occurrence sub-pattern. In addition, these events need to satisfy all conditions imposed by the other elements of the pattern (i.e., edges, attachments, and entity nodes). For each absence node, in turn, the corresponding events must be absent or not meet the conditions imposed by connected edges and attachments.

Example 3.2 (eCRG semantics)
The eCRG semantics is illustrated using the activity log depicted in Table 3.1 and eCRG C1 (cf. Fig. 3.3). Consider Table 3.1: C1 is activated by activity `request informed consent`, which is performed by physician *Mrs. E* (cf. entry #6). In particular, a match of the antecedence pattern consists of activity `request informed consent` matching the corresponding antecedence node and physician *Mrs. E* matching the connected antecedence resource node. A related match of the consequence pattern is given by activity `inform patient` (cf. entry #5) that complies with all conditions of the consequence pattern: Activity `inform patient` was executed before activity `request informed consent` (consequence sequence flow); further, it was executed by the same resource *Mrs. E* (consequence performing connector). Moreover, *Mrs. E* is a physician as required by the consequence resource relation.

Consequently, the given activity log conforms to compliance rule C1.

Table 3.1: Exemplary event log (cf. Section 2.1)

#	date	duration	activity	data
1	3.5.11	11:25-11:35	examine patient (Mrs. E - physician)	
2	3.5.11	11:35-11:39	fill request form (Mrs. E - physician)	*out*: X-ray request (id=27051)
3	3.5.11	11:39-11:39	request appointment (to radiology)	*in*: X-ray request (id=27051)
4	3.5.11	13:10-13:11	appointment (from radiology)	
5	3.5.11	13:10-13:14	inform patient (Mrs. E - physician)	
6	3.5.11	13:15-13:20	request informed consent (Mrs. E - physician)	*out*: informed consent (id=27091)
7	5.5.11	09:28-09:28	call for patient (from radiology)	
8	3.5.11	13:14-...	drop off patient (Mr. H - nurse)	

The eCRG semantics is formally specified through a translation of eCRGs into FOL expressions based on completed process logs [KRL+13a]. As example Table 3.2 shows the FOL expression for compliance rule C1.

The elements of the eCRG language (cf. Fig. 3.2) and, in particular, activity and message nodes are optimized for specifying compliance rules from the viewpoint of a given organization, i.e., from the viewpoint of public or private process models. In the context of cross-organizational processes, however, a global viewpoint, i.e., the viewpoint of interaction models, is more convenient for specifying compliance rules referring to multiple partners. For this purpose, [KRL+13b] introduced additional eCRG elements for activity and interaction nodes in the context of cross-organizational compliance rules

Table 3.2: FOL expression for compliance rule C1

$$\begin{aligned}\forall \nu_1^i, \nu_{s1}^t, \nu_{e1}^t, \nu_1^r, \nu_2^r : \big(& (Start(\nu_{s1}^t, \nu_1^i, \texttt{request informed consent}) \wedge End(\nu_{e1}^t, \nu_1^i) \wedge \nu_{s1}^t \leq \nu_{e1}^t \\ & \wedge Perform(\nu_1^i, \nu_1^r) \wedge (\nu_2^r = \nu_1^r)) \\ \rightarrow \exists \nu_2^i, \nu_{s2}^t, \nu_{e2}^t, \nu_3^r, \nu_4^r : & (Start(\nu_{s2}^t, \nu_2^i, \texttt{inform patient}) \wedge End(\nu_{e2}^t, \nu_2^i) \wedge \nu_{s2}^t \leq \nu_{e2}^t \\ & \wedge Perform(\nu_1^i, \nu_3^r) \wedge (\nu_2^r = \nu_3^r) \wedge (\nu_2^t < \nu_1^t) \\ & \wedge rel_{\texttt{has role}}(\nu_2^r, \nu_4^r) \wedge (\nu_4^r = \texttt{physician}))\big)\end{aligned}$$

(cf. Fig. 3.5). In particular, the partner responsible for an activity as well as both sender and receiver of an interaction (i.e., message exchange) can be explicitly specified.

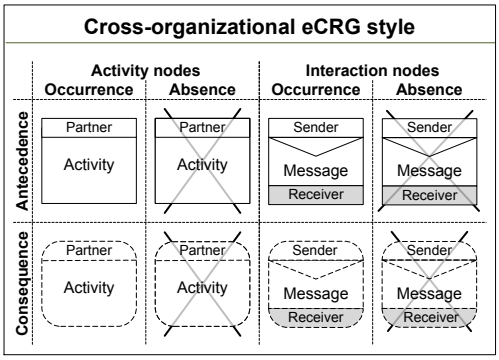


Figure 3.5: eCRG elements for cross-organizational compliance rules (adopted from [KRL+13b])

3.3 Evaluation

The eCRG language was subject to several evaluations that focus on different aspects. Sections 3.3.1 and 3.3.2 investigate the suitability of the eCRG language for modeling compliance rules. In particular, Section 3.3.1 shows that the eCRG language is able to cover the typical compliance patterns known from literature [TEvP12, RFA12]. This claim was further verified in a case study in the medical domain (cf. Section 3.3.2). Section 3.3.3 presents empirical studies investigating the comprehensibility of the eCRG language. Finally, a prototypical implementation of an *a posteriori* compliance checker is introduced in Section 3.3.4; it confirms that eCRGs can be applied for compliance checking.

Table 3.3: Support of compliance patterns

A. Business process control patterns and Property specification patterns			
Precedes	+	USegregatedFrom	+
LeadsTo	+	BondedWith	+
XLeadsTo	+	RBondedWith	+
PLeadsTo	+	Multi-Segregated	0
ChainLeadsTo	+	Multi-Bonded	+
Chain Precedes	+	Within k	+
LeadsTo - Else	+	After k	+
Exists	+	ExactlyAt k	+
Absent	+	Exists Max/Min	+
Universal	+	Exists Every k	+
CoExists	+		
CoAbsent	+		
Exclusive	+		
CoRequisite	+		
MutexChoice	+		
PerformedBy	+		
SegregatedFrom	+		

B. Compliance rule pattern categories	
Existence	+
Bounded existence	+
Bounded sequence	+
Parallel	+
Precedence	+
Chain precedence	+
Response	+
Chain response	+
Between	+
Exclusive	+
Mutual exclusive	+
Inclusive	+
Prequisite	+
Substitute	+
Corequisite	+
Data flow	+
Organizational	+

C. Time patterns	
Time lags between activities	+
Durations	+
Time lags between events	+
Fixed date elements	+
Schedule restricted elements	+
Time-based restrictions	+
Validity period	+
Time-dependend variability	+
Cyclic elements	+
Periodicity	+

Legend	
+	support
0	limited support
-	no support

3.3.1 Pattern-based Evaluation

To evaluate whether the eCRG language is able to cover typical compliance rule patterns, *business process control patterns* [TEvP12], *compliance rule patterns* [RFA12], and *time patterns* [LWR14, LRW16] were modeled as eCRGs.

As a result, all 27 business process control patterns can be modeled using the eCRG language, including 5 time patterns and 7 resource patterns. However, for certain instances of the *Multi-Segregated* pattern, large eCRGs result. Furthermore, the eCRG language covers all 55 compliance rule patterns. Finally, all time patterns described in [LRW16] can be modeled as eCRGs (see [KRL$^+$13a] for the respective eCRGs).

Table 3.3 summarizes the results of the pattern-based eCRG evaluation.

3.3.2 Modeling Real-World Compliance Rules as eCRGs

To validate the practical suitability of the eCRG language, it was applied to real-world compliance rules taken from the healthcare domain. A Master student from Management Science analyzed compliance rules in the context of six process model collections as well as related documents that stem from a large process reengineering project performed at a university hospital [KRS96a, KSR96a, KSR96b, KRS$^+$96b, SMM$^+$96c, SMM$^+$96a, SMM$^+$96b, Sem13].
The student was able to model all of the 30 compliance rules as eCRGs, not only confirming that the eCRG language is able to cover a variety of compliance rules, but also indicating that domain experts are able to comprehend and use eCRGs after having received some training [SKR14].

3.3.3 Experimental Evaluation

To investigate the comprehensibility of the eCRG language, two controlled experiments with students were conducted. The results of the first experiment were discussed in [KR17], whereas the ones of the second experiment have not been published yet.

Comparison between user groups

The first experiment investigated the difference between domain and IT experts regarding the understanding of the eCRG language. In particular, the experiment focused on the evaluation of the following three hypotheses [NK16]:

H1 Trained management scientists are able to understand eCRGs, i.e., reading eCRGs increases their domain understanding.

H2 Trained computer scientists are able to understand eCRGs, i.e., reading eCRGs increases their domain understanding.

H3 There is no large difference between students from Management Science and Computer Science regarding the understanding of eCRGs.

Two different subject groups were involved, i.e., 55 students from Management Science (i.e., prospective domain experts) and 25 students from Computer Science (i.e., IT experts). 59 subjects were male and 21 were female. All subjects attended a course on business process management. Additionally, they were provided with a short training before participating in the experiment.

During the experiment, the subjects had to answer a questionnaire with 30 questions, which were related to 10 eCRGs. Altogether, the subjects could reach a score of up to 30 points. Descriptive statistics and boxplots are shown in Table 3.4 and Fig. 3.6.

Table 3.4: Descriptive statistics (adapted from [NK16, KR17])

group	N	min	max	avg	sd	sem
Management Scientists	55	10.0	30.0	20.182	4.583	0.618
Computer Scientists	25	11.0	29.0	20.920	4.743	0.949
Overall	80	10.0	30.0	20.413	4.616	0.516

The results of Kolmogorov-Smirnov-Lilliefors tests indicate that a normal distribution must not be assumed for the score in general (p-value 0.006). However, for both Management Scientists and Computer Scientists, a normal distribution may be assumed (p-values 0.089 respective 0.200). Thus, non-parametric tests were applied in addition to parametric t-tests.

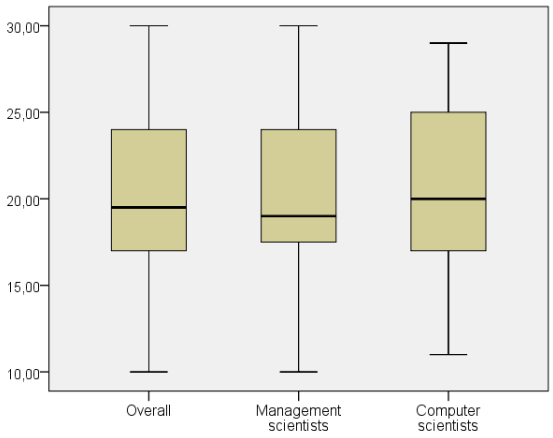


Figure 3.6: Boxplots of the scores for management and computer scientists

As shown in Table 3.5, the scores of both groups significantly differ from the expected value (i.e., random guessing). In particular, *one-sample t-tests* and *one-sample Wilcoxon signed-rank tests* strongly support H1 and H2 based on a 0.05 significance level. Finally, very large effects can be observed.

Table 3.5: Increase in domain understanding (H1 and H2) (adopted from [NK16, KR17])

	one-sample t-test			**one-sample Wilcoxon signed-rank test**		
	t	p-value	effect d	Z	p-value	effect d
H1	8.385	<0.001	1.963	-5.712	<0.001	1.660
H2	6.241	<0.001	2.120	-4.080	<0.001	1.025

Comparing the scores of students from Management and Computer Science reveals no significant difference between the two groups (cf. Table 3.6). In particular, an *unpaired two-sample t-test* and a *Mann-Whitney U signed-rank test* were conducted. Both tests observed only trivial effects and their power was strong enough to exclude large and medium effects with *Cohen's* $d > 0.7$.

Table 3.6: Comparison of the two groups (H3) (adopted from [NK16, KR17])

	unpaired two-sample t-test			**Mann-Whitney U test**		
	t	p-value	effect d	Z	p-value	effect d
H3	0.661	0.511	0.158	-0.432	0.670	0.097

Altogether, the experiment confirms that not only Computer Science students (i.e. IT experts), but also Management Science students (i.e. domain experts) are able to comprehend eCRGs. Furthermore, the eCRG comprehension of both groups does not significantly differ. Consequently, the eCRG language has the potential to foster discussions among the two user groups.

Comparison with LTL and compliance patterns

The second experiment compared the comprehensibility of eCRGs with other compliance rule languages. The experiment compared the eCRGs with an extended set of the compliance patterns from [TEvP12] as well as an LTL extension. In particular, the following hypotheses were investigated in this context:

H4 eCRGs increase domain understanding more than LTL expressions.

H5 eCRGs increase domain understanding more than compliance patterns.

H6 eCRGs increase domain understanding of the control flow perspective more than LTL expressions.

H7 eCRGs increase domain understanding of the control flow perspective more than compliance patterns.

H8 eCRGs increase domain understanding of the time perspective more than LTL expressions.

H9 eCRGs increase domain understanding of the time perspective more than compliance patterns.

H10 eCRGs increase domain understanding of the resource perspective more than LTL expressions.

H11 eCRGs increase domain understanding of the resource perspective more than compliance patterns.

H12 eCRGs increase domain understanding of the data perspective more than LTL expressions.

H13 eCRGs increase domain understanding of the data perspective more than compliance patterns.

Overall, 44 Students from Computer Science (28 male, 16 female) were involved in the experiment. All subjects were provided with a short training before the experiment. The experiment itself was based on a questionnaire with 36 questions referring to 12 compliance rules. The latter, cover the control flow, time, resource, and data perspectives.

Descriptive statistics and the distribution of the experiment are shown in Table 3.7 and Fig. 3.7.

Kolmogorov-Smirnov-Lilliefors tests indicate that a normal distribution must neither be assumed for the overall score (p-value 0.016) nor for the time (p-value <0.001), resource (p-value 0.027), and data perspectives (p-value 0.010). However, for the process

Table 3.7: Descriptive statistics

language	N	min	max	avg	sd	sem
eCRG	44	1.0	12.0	7.352	3.081	0.465
Patterns	44	0.5	12.0	7.000	2.901	0.437
LTL	44	0.0	12.0	6.193	3.067	0.462
eCRG control flow	44	0.0	3.0	1.614	0.993	0.15
Patterns control flow	44	0.0	3.0	1.750	0.973	0.147
LTL control flow	44	0.0	3.0	1.216	0.955	0.144
eCRG time	44	0.0	3.0	1.807	1.007	0.152
Patterns time	44	0.0	3.0	1.818	0.941	0.142
LTL time	44	0.0	3.0	1.716	1.070	0.161
eCRG resource	44	0.0	3.0	2.011	0.918	0.138
Patterns resource	44	0.0	3.0	1.886	1.022	0.154
LTL resource	44	0.0	3.0	1.705	1.101	0.166
eCRG data	44	0.0	3.0	1.920	0.895	0.135
Patterns data	44	0.0	3.0	1.545	1.099	0.166
LTL data	44	0.0	3.0	1.557	1.019	0.154
Overall	44	7.0	30.0	20.545	6.646	1.002

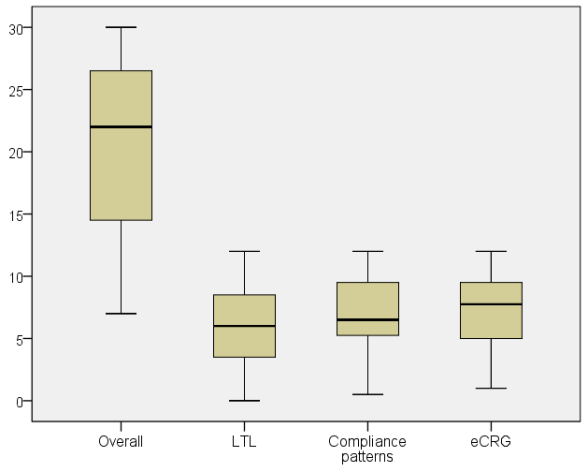


Figure 3.7: Boxplots of the scores comparing eCRGs with LTL and compliance patterns

perspective a normal distribution may be assumed (p-value 0.056). To enable a uniform analysis of hypotheses H4 – H13, both parametric and non-parametric tests were applied in the following.

As shown in Table 3.8, all hypotheses were tested with a *dependent-sample t-test* as well as a *two-sample Wilcoxon signed-rank test*. Both tests show that, in general, eCRGs can be significantly better understood than LTL expressions (cf. H4). Furthermore, a medium effect between both languages can be observed. In turn, the tests were unable to confirm that eCRGs are significantly better understood than compliance patterns (cf. H5), although a small effect can be observed between the two languages.

Table 3.8: Comparisons of the increases in domain understanding (H4 to H13)

	paired dependent-sample t-test			**two-sample Wilcoxon signed-rank test**		
	t	p-value*	effect d	Z	p-value*	effect d
H4 (vs LTL, overall)	2.189	0.017	0.377	-2.004	0.023	0.634
H5 (vs Patterns, overall)	0.657	0.257	0.118	-0.531	0.298	0.161
H6 (vs LTL, control flow)	2.296	0.013	0.408	-2.357	0.009	0.760
H7 (vs Patterns, control flow)	-0.761	0.225	-0.139	-0.655	0.256	0.198
H8 (vs LTL, time)	0.425	0.337	0.088	-0.330	0.371	0.100
H9 (vs Patterns, time)	-0.064	0.475	-0.012	-0.157	0.438	0.047
H10 (vs LTL, resources)	1.840	0.036	0.301	-1.689	0.046	0.527
H11 (vs Patterns, resources)	0.674	0.252	0.129	-0.713	0.238	0.216
H12 (vs LTL, data)	2.190	0.017	0.378	-2.030	0.021	0.643
H13 (vs Patterns, data)	2.178	0.017	0.372	-1.982	0.024	0.626

*) p-values are one-sided

When analyzing the individual perspectives, similar results can be obtained. Except the time perspective (cf. H8), eCRGs can be significantly better understood than LTL (cf. H6, H10 and H12). In each of these cases, a medium effect can be observed between eCRGs and LTL. Furthermore, the tests revealed no significantly better understanding of eCRGs compared to compliance patterns regarding the control flow, time, and resource perspectives (cf. H7, H9 and H11). For these perspectives only trivial effects can be observed, some of them are even negative, i.e., in these cases, *in average*, but without significance, compliance patterns were a little bit better understood. Concerning compliance rules focusing on the data perspective, however, eCRGs are significantly better understood than compliance patterns. In this context, a medium effect can be observed.

Despite the rather low number or participants, the experiment revealed that Computer Science students understand the eCRG language significantly better than LTL expressions. On average, however, eCRGs are also better understood than compliance patterns.

3.3.4 Proof-of-Concept Prototype

A proof-of-concept prototype and a set of Microsoft Visio shapes were developed to demonstrate the feasibility of the eCRG language.

Fig. 3.8 shows a screenshot of the modeling component of the prototype, which enables the web-based modeling and management of eCRGs. The prototype further supports different export formats, including SVG, JPG, PDF, and XML. Another component of the prototype (cf. Fig. 3.9) enables *a posteriori* checking of the compliance of completed process logs with eCRGs (cf. Section 2.4.4). The component can import eCRGs from the web-based modeling prototype, but also provides a Java API for the build-in specification of both eCRGs and event logs.

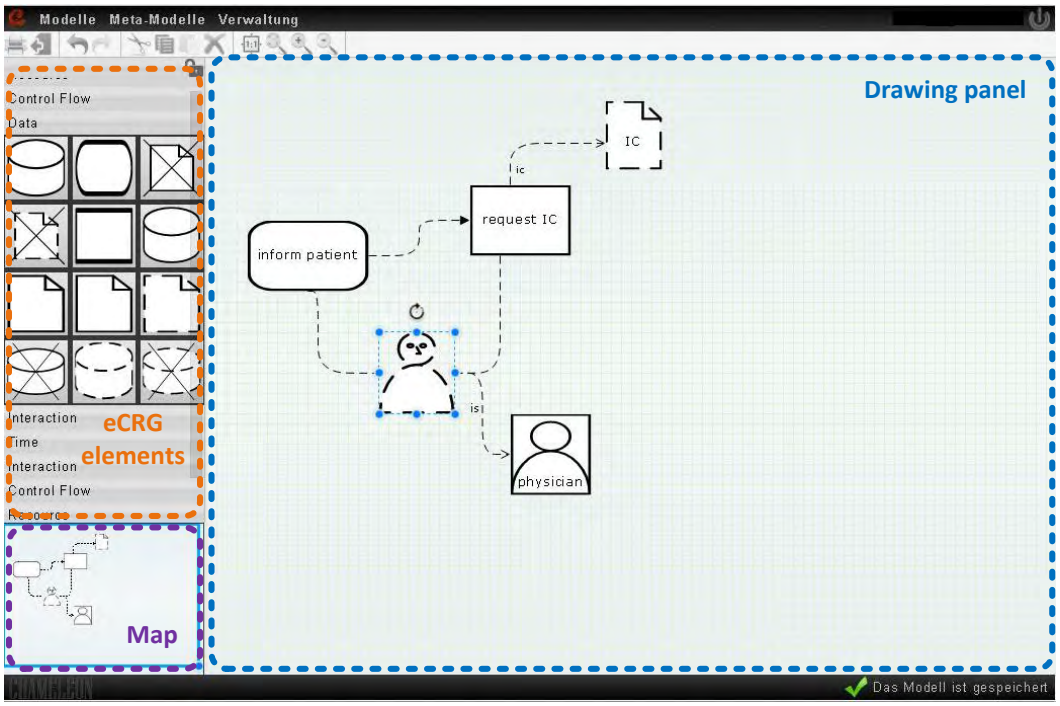


Figure 3.8: eCRG modeling prototype

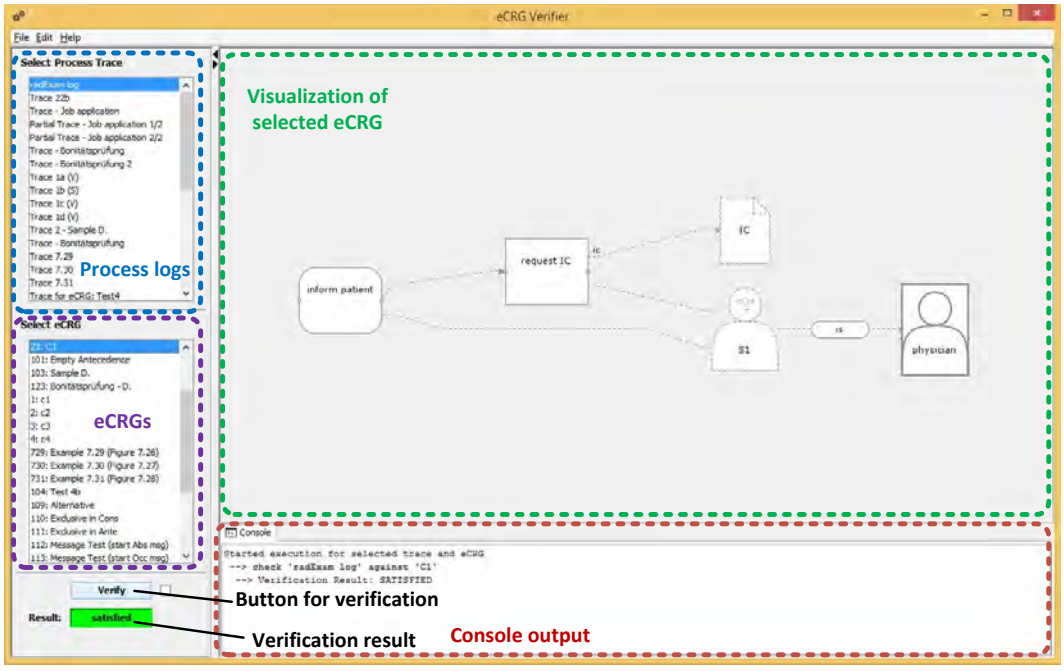


Figure 3.9: A posteriori compliance checking of process logs with eCRGs

Besides the modeling prototype, shape palettes for Microsoft Visio were developed (cf. Fig. 3.10). These cover all elements of the eCRG language. In particular, the Visio shapes provide a useful alternative for drawing eCRGs, which are not intended for any subsequent electronic processing (e.g., automated compliance checking).

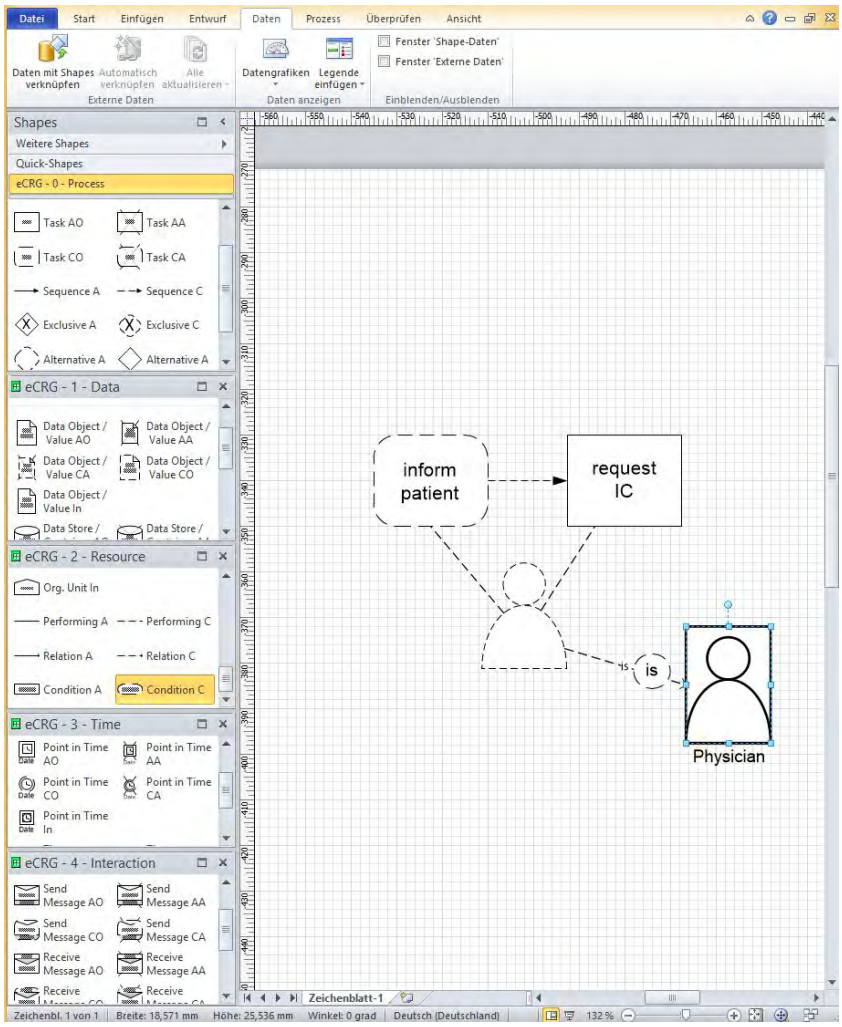


Figure 3.10: Palettes of Microsoft Visio shapes for the eCRG language

3.4 Related Work

Besides the eCRG language there exist other languages for creating machine-readable specifications of process compliance rules. These languages can be categorized into logic-based, pattern-based, and visual languages.

Logic-based languages were used in early approaches for specifying compliance rules:

Temporal logic like LTL [Pnu77] and the computation tree logic (CTL) [CE81] stem from the field of automated finite-state verification and, therefore, can be applied to the state space of a process model if it is finite. Temporal logic enhances ordinary propositional logic with additional temporal operators. For example, LTL uses the path operators *X next*, *F eventually*, *G globally*, *U until*, and *W weakly until* to enable a point-to-point navigation on discrete time lines. CTL considers the branching structure of finite-state machines, which allows for different continuations starting from a given state. In particular, CTL uses path-quantifiers as prefixes for the LTL path operators. Finally, [LMX07, GK07] used temporal logic to specify business process compliance rules.

Temporal logic constraints are not restricted to the control flow perspective, but may refer to the data and resource perspectives as well. Furthermore, there exist LTL extensions properly supporting the time perspective. For example, [TEvP12] uses temporal logic for specifying compliance rule patterns that not only refer to the control flow perspective, but to the time, data, and resource perspectives as well.

The *formal contract language* (FCL) [GS09] constitutes another logic-based formalism that was used for specifying compliance rules. As FCL is based on deontic logic, it provides explicit support for normative concepts like *obligations*, *permissions* and *prohibitions*. Furthermore, FCL allows specifying *compensations* of compliance violations. Due to their deontic nature, the structure of FCL constraints becomes simpler and more intuitive than the one of the respective temporal logic constraints [GH15].

Logic-based approaches, however, are complex to handle [CTZ+16] as even IT experts have difficulties in understanding logic expressions. Literature, therefore, suggests introducing other approaches for domain experts [TEvP12, LRMD10, AWW11].

Rule-based [Her96] and artifact-based [Hul08] approaches allow addressing multiple perspectives of business processes and compliance rules as well. Rule-based approaches are supported by business rule engines as well as rule-specific standards like BRML [GL00], and SBVR [OMG17]. Like logic-based approaches, they are text-based and do not provide an explicit visual support of the different perspectives of business processes and compliance rules respectively.

Compliance patterns cover formal details (e.g., expressions in LTL) behind descriptive names and textual descriptions. They use placeholders to abstract from specific activities. The initial set of compliance patterns suggested by [DAC99] was originally proposed for verifying finite state systems. Several extensions of the compliance patterns were proposed. Focusing on the control flow perspective, [RFA12] presents a set of 55 compliance rule patterns whose semantics was specified in terms of Petri-Nets. [RFDA13] restructured this compliance pattern set, extending it with 15 time-related compliance patterns.

[TEvP12, ZF15] introduce a smaller sets of compliance patterns, which cover the control flow, time, data, and resource perspectives. In particular, the patterns were formalized in terms of temporal logic. [ZF15] suggests further refinements of the data perspective to provide explicit support for *location-* and *temperature-aware* compliance rules.

Obviously, pattern-based approaches still rely on formal specifications that have to be manually defined before the patterns can be used. Consequently, pattern-based approaches only reduce the problems coming with formal approaches [CTZ+16]. In addition, extensive sets of compliance patterns (e.g., [RFA12]) make it more difficult for domain experts to select the right patterns and understand their meaning [RFA14].

Only few visual languages exist for modeling compliance rules. *G-CTL* [FSWS11] enables the visual modeling of CTL focusing on the control flow perspective. In a similar, but more advanced way, [LMX07] supports the visual specification of LTL constraints. Furthermore, it supports the specification and reuse of LTL templates, which can be used as compliance patterns. This work is not restricted to LTL as underlying formalism, but can be applied to CTL as well. Moreover, it already provides support for both the control flow and the data perspective. [LMX07] and [FSWS11] have a one-to-one relationships with the underlying temporal logic. In particular, both languages do not abstract from temporal operators.

BPMN-Q [AWW09, AWW11] provides a high-level language enabling the visual specification of compliance rules that cover the control flow and data perspectives. As described in Section 2.4.1, *path edges* are used to visualize specific compliance patterns. The latter can be refined with data conditions, expressed in terms of stateful data objects. The semantics of BPMN-Q components is defined by a translation to LTL and CTL expressions.

Visual languages for the declarative modeling of business processes [PSA07] can be used to model compliance rules as well. The DECLARE language and its various extensions are based on set of patterns similar to the ones of BPMN-Q. DCR Graphs [HMS12] and ConDec++ [Mon10], in turn, provide support for the time, data and resource perspectives as well. However, there exist empirical works indicating rather low comprehensibility of DECLARE expressions [HZ14, ZSH+15, HBZ+16].

[DSDB15] introduces a generic and visual query language for graph-based models that may be used to specify compliance rules as well. However, this language focuses on the structure of models, but not on their dynamic behavior as covered by compliance rules.

Tables 3.9 and 3.10 use the business process compliance patterns from [TEvP12] as benchmark for comparing the eCRG language with other visual languages for specifying compliance rules and declarative processes, i.e., BPMN-Q, ConDec++, and DCR Graphs. Additionally, Table 3.10 investigates whether data conditions (*Data-Condition*) and data-based correlations of activities (*Data-Bonded*) are supported.

As summarized in Table 3.9, the fundamental existence patterns of the control flow perspective are supported by all approaches. However, the eCRG language is the only language that properly supports the chain precedence and response order patterns. Regarding the resource, time, and data perspectives (cf. Table 3.10), the benefits of the eCRG language become even more obvious. Note that only the eCRG language properly distinguishes between performers and roles, and, hence, it provides different visual

representations for both concepts. BPMN-Q does not support the resource perspective. In turn, ConDec++ solely supports the use of performers in local models and abstract roles in choreographies. Accordingly, the interplay of both concepts becomes difficult to handle. DCR Graphs support the assignment of roles and groups, but only roles are included in the visual model and considered by the formal semantics. Regarding the time perspective, the eCRG language and ConDec++ cover all time patterns. In turn, DCR Graphs only partially support temporal aspects. BPMN-Q does not consider the temporal aspects. In the context of the data perspective, DCR Graphs and BPMN-Q do not enable the binding of activities based on the data they access. Again the eCRG language and ConDec++ provide full support of the considered patterns.

Table 3.9: Language comparison: Control flow perspective

	Order							Existence							
	Precedes	LeadsTo	XLeadsTo	PLeadsTo	ChainLeadsTo	ChainPrecedes	LeadsToElse	Exists	Absent	Universal	CoExists	CoAbsent	Exclusive	CoRequisite	MutexChoice
BPMN-Q	+	+	+	+	-	-	-	+	+	+	+	+	+	+	+
ConDec++	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+
DCR	+	+	+	+	0	0	+	+	+	+	+	+	+	+	+
eCRG	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Table 3.10: Language comparison: Resource, time, and data perspectives

	Resource							Time					Data	
	PerformedBy (Role)	SegregatedFrom	USegregatedFrom	BondedWith	RBondedWith	Multi-Segregated	Multi-Bonded	Within k	After k	ExactlyAt k	Exists Max/Min	Exists Every k	DataCondition	DataBonded
BPMN-Q	-	-	-	-	-	-	-	-	-	-	-	-	+	-
ConDec++	+	0	+	0	0	-	+	+	+	+	+	+	+	+
DCR	+	+	0	0	0	-	+	0	0	0	+	+	0	-
eCRG	+	+	+	+	+	0	+	+	+	+	+	+	+	+

+ support, 0 limited support or missing visualization, - no support

In summary, the eCRG language provides a more sophisticated support and better coverage of the compliance patterns from [TEvP12] compared to other visual languages for modeling compliance rules (i.e. BPMN-Q). Moreover eCRGs even surpass the expressive power of visual declarative approaches (i.e., ConDec++ and DCR Graphs).

A broader discussion of related work can be found in [KR17].

3.5 Summary

[KRL+13b, KR17] introduced the *extended Compliance Rule Graph* (eCRG) language, which enables the visual modeling and verification of compliance rules that may refer to the control flow, resource, time, data and interaction perspectives. eCRGs rely on a formal semantics and are evaluated in several respects. In addition to a proof-of-concept prototype, the application to real-world cases, empirical evaluations, and a systematic comparison with related work show the benefits of the eCRG language.

4

Monitoring Multi-Perspective Business Process Compliance

This chapter is based on the following publications:

[KRK15a]: **D. Knuplesch**, M. Reichert, and A. Kumar. Towards visually monitoring multiple perspectives of business process compliance. In *Advanced Information Systems Engineering (CAiSE'15) Forum*, pages 41–48. CEUR-WS, 2015

[KRK15b]: **D. Knuplesch**, M. Reichert, and A. Kumar. Visually monitoring multiple perspectives of business process compliance. In *13th International Conference on Business Process Management (BPM'15)*, volume 9253 of *LNCS*, pages 263–279. Springer, 2015

A significantly extended version of these works was published as follows:

[KRK17]: **D. Knuplesch**, M. Reichert, and A. Kumar. A framework for visually monitoring business process compliance. *Information Systems*, 64, pages 381 – 409, 2017

The original articles are provided in Appendices 5, 6 and 9, respectively. Additional formal details can be found in [KR14].

4.1 Research Challenges

The conformance of business processes with imposed compliance rules can be verified in different phases of the process life cycle. Approaches that monitor the compliance of running process instances during the execution phase are denoted as *compliance monitoring* (cf. Section 2.4). In particular, such approaches analyze the activity and event logs of running processes instances in order to detect compliance violations in real time.

Compliance monitoring faces several requirements that were analyzed and summarized in [LMM+15]. In particular, [LMM+15] enumerates 10 *compliance monitoring functionalities* (CMF) discovered through a systematic literature review. The 10 CMFs from [LMM+15] are summarized in Table 4.1.

Table 4.1: Compliance monitoring functionalities [LMM+15]

CMF1	time perspective
CMF2	data perspective
CMF3	resource perspective
CMF4	non-atomic activities
CMF5	activity lifecycle
CMF6	multiple instances
CMF7	reactive management
CMF8	proactive management
CMF9	root cause analysis
CMF10	compliance degree

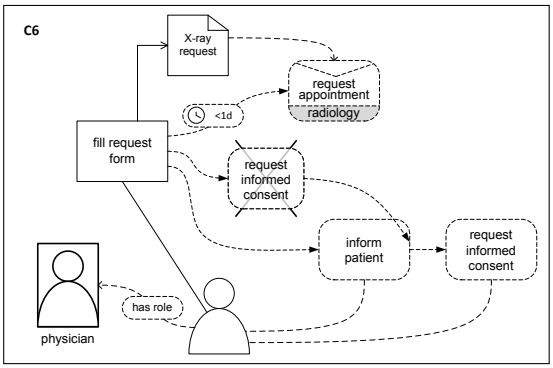


Figure 4.1: Specific eCRG for compliance rule C6

Table 4.2: Partial event log (cf. Section 2.1)

#	date	time	type	act id	activity / data
1	3.5.11	11:25	start	1	examine patient (Mrs. E - physician)
2	3.5.11	11:35	end	1	examine patient
3	3.5.11	11:35	start	2	fill request form (Mrs. E - physician)
4	3.5.11	11:38	write	2	X-ray request (id=27051)
5	3.5.11	11:39	end	2	fill request form
6	3.5.11	11:39	send	3	request appointment (to radiology)
7	3.5.11	11:39	read	3	X-ray request (id=27051)
8	3.5.11	11:39	end	3	request appointment
9	3.5.11	13:10	receive	4	appointment (from radiology)
10	3.5.11	13:11	end	4	appointment
11	3.5.11	13:10	start	5	inform patient (Mrs. E - physician)
12	3.5.11	13:14	end	5	inform patient
13	3.5.11	13:15	start	6	request informed consent (Mrs. E - physician)
14	3.5.11	13:19	write	6	informed consent (id=27091)
15	3.5.11	13:20	end	6	request informed consent
16	4.5.11	09:28	receive	7	call for patient (from radiology)
17	4.5.11	09:28	end	7	call for patient
18	3.5.11	13:14	start	7	drop off patient (Mr. H - nurse)
⋮	⋮	⋮	⋮	⋮	⋮
43	9.5.11	07:57	start	18	examine patient (Mr. G - physician)
44	9.5.11	08:12	end	18	examine patient
45	9.5.11	08:45	start	19	fill request form (Mr. G - physician)
46	9.5.11	08:46	write	19	X-ray request (id=27219)
47	9.5.11	08:46	end	19	fill request form
48	10.5.11	09:15	start	6	request informed consent (Mr. G - physician)
49	10.5.11	09:19	write	6	informed consent (id=27091)

Example 4.1 (Compliance monitoring functionalities)
CMFs are described along an example of an event log (cf. Table 4.2) and a compliance rule (cf. Fig. 4.1), which refer to the X-ray examination process introduced in Section 2.1.4.

Compliance rule C6 (cf. Fig. 4.1) is triggered after filling an X-ray request form. Within one day, the filled form should then be forwarded to the radiology department. The person filling the form needs to be a physician, who has to inform the patient about the risks of the examination and to request her informed consent. Finally, the informed consent must not be requested before completing activity `inform patient`.

Compliance rule C6 not only refers to the control flow perspective, but to the *time perspective* (*within one day*), *data perspective* (data object *X-ray request*), and *resource perspective* (*must be a physician*, *the same person*) as well. Consequently, any compliance monitoring approach needs to consider these perspectives as well (CMF1-3).

Like in the event log from Table 4.2, a particular activity instance may be associated with multiple events (e.g., Events 3-5 are related to activity `fill request form` with activity id 2). In turn, this raises the demand for approaches being able to handle *non-atomic activities* and their *lifecycle*, i.e., to properly correlate events created by the same activity (CMF4+5).

The event log from Table 4.2 triggers compliance rule C6 twice, once in the context of the first X-ray request (with id 27051) and a second time when completing the second X-ray request (with id 27219) six days later. While the former instance of C6 is satisfied, the latter is violated, as no appointment was made within the permitted time. In general, compliance monitoring should allow for the continuous monitoring of *multiple instances* of the same compliance rule concurrently (CMF6).

Reactive compliance monitoring and *management* detects compliance violations after their occurrence (CMF7). In turn, *proactive compliance monitoring* and *management* aims to preventing compliance violations before they occur. This can be accomplished, for example, by recommending the activities whose execution is required to satisfy a compliance rule. In the scenario, `request (an) appointment` could be highlighted after filling the second X-ray request (with id 27219), i.e., after triggering compliance rule C6 a second time (CMF8).

In case of violations, it is useful to be able to explain the *root causes* of compliance violations like the missing activity `inform patient` in the example (CMF9). Finally, compliance metrics (e.g., *degree of compliance*) shall allow assessing the severity of compliance rule violations (CMF10).

Although the CMFs are considered to be fundamental requirements for any compliance monitoring approach, [LMM+15] confirms that existing approaches do not provide full support for more than six CMFs.

4.2 Scientific Contribution

This chapter summarizes the major contribution provided by [KRK15b] and [KRK17], which describe a framework for the visual monitoring of business process compliance during process execution. The framework covers all compliance monitoring functionalities (CMF) introduced by [LMM+15]. Further, it applies the eCRG language (cf. Chapter 3) and, thus, not only covers the control flow perspective, but also the time, data, and resource perspectives (cf. CMF 1-3) as well as the interaction perspective.

Like other monitoring approaches, the eCRG framework reacts on the events created by one or multiple running processes. In response to these events, the elements of an eCRG are marked step-by-step with text labels, colors and symbols to highlight the state of the compliance rule. The latter, in turn, is composed of the states of its individual elements (cf. Fig. 4.2). Activity (and message) nodes may pass several states including *not activated* (□), *activated* (△), *running* (▶), *completed* (✓), and *skipped* (×). Initially, the other eCRG elements are *not marked* (□), and may later be marked as *satisfied* (✓) or *violated* (×). In addition, activity (and message) nodes may be annotated with the timestamps at which the corresponding events occurred. Data flow connectors and data objects, in turn, are annotated with the values and identifiers of related data, whereas activity and resource nodes are further annotated with identifiers of related resources.

In the following, the term *eCRG marking* refers to an eCRG whose elements are annotated with text, colors and symbols.

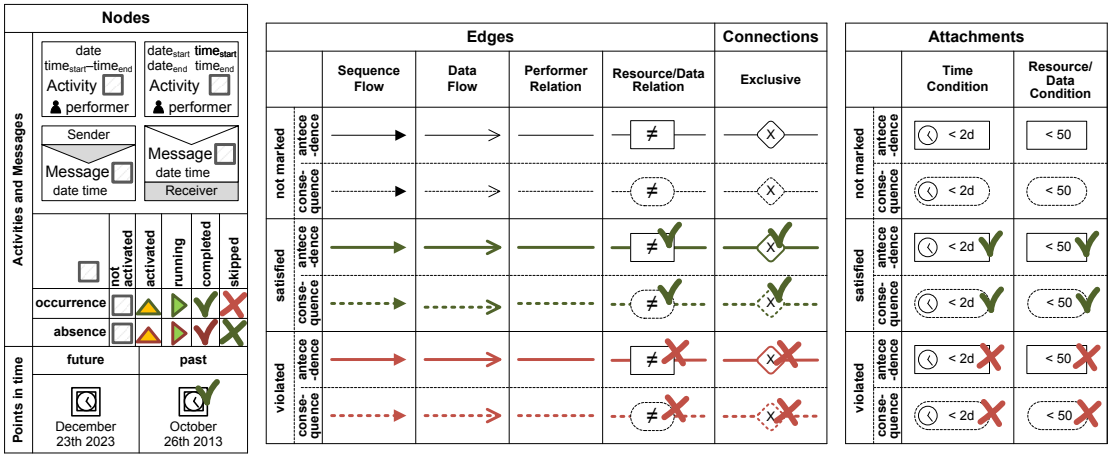


Figure 4.2: Annotations of eCRG elements (adopted from [KRK17])

In order to support the monitoring of multiple instances of the same compliance rule not only an initial marking of the observed eCRG is used and evolved, but additional copies of this marking are dynamically created on-the-fly if required (cf. CMF6). The same copy mechanism is further applied when occurred events allow for different interpretation

(i.e., non-determinism). Accordingly, there exists an eCRG marking for each possible interpretation of the event stream.

The eCRG monitoring framework supports different kinds of events (cf. Table 4.3). The third column of Table 4.3 outlines whether an event adapts existing markings (i.e., the processing of the event is *deterministic*) or works on a newly copied one (i.e., the processing of the event is *non-deterministic*). After processing an event, the obtained set of markings is analyzed to assess compliance (cf. CMF7). In addition, markings are used to proactively recommend required actions to users as well as to indicate forbidden actions (cf. CMF8). In case of compliance violations, in turn, the markings are utilized to highlight potential root causes of the violation (cf. CMF9). Finally, compliance metrics are calculated based on the markings (cf. CMF10).

Table 4.3: Supported events

Category	**Event and signature**	**Determinism**
Activity	`start`(time, activityId, activityType, performer)	non-deterministic
	`end`(time, activityId, activityType)	deterministic
Message	`send`(time, messageId, messageType, receiver)	non-deterministic
	`receive`(time, messageId, messageType, sender)	non-deterministic
	`end`(time, messageId, messageType)	deterministic
Data flow	`write`(time, activityId/messageId, value, param)	deterministic
	`read`(time, activity/messageId, value, param)	deterministic

[KRK15b] and [KRK17] present algorithms that deal with all CMFs including the processing of events.

In the following, the processing of these events and compliance assessments are illustrated. Processing starts with the *initial marking* of the eCRG to be monitored. This *initial marking* assigns state *activated* (△) to all activity and message nodes not depending on any of their predecessors in the control flow. All other elements are marked with □, i.e., as *not activated* and *not marked* respectively. In this context, elements of the antecedence absence and consequence occurrence patterns depend on connected elements of the antecedence occurrence pattern. In turn, elements of the consequence absence pattern depend on connected elements of the antecedence and consequence occurrence patterns [KRK15b, KRK17].

Fig. 4.3 shows the processing of a start event on initial marking *M1*. Since activity node *fill request* is in state *activated* (△) under *M1*, the corresponding start event may be processed. First, the original marking (i.e. *M1*) is copied. Then, the state of activity node *fill request* changes to *running* (▶) on this copy. In addition, the activity node is annotated with the current point in time (i.e., 03.05.11; 11:35) and the corresponding resource (i.e. Mrs. E). Following this, Fig. 4.4 shows how resource information is propagated from the running activity node to a connected resource node via the connecting edge. Finally, resource conditions and relations (e.g. relation *has role*) are evaluated. As no conflicts

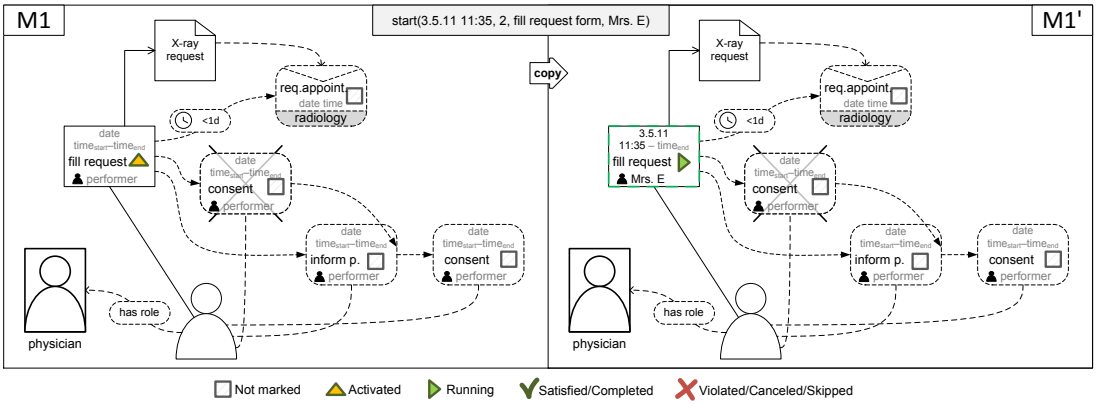


Figure 4.3: Processing start events

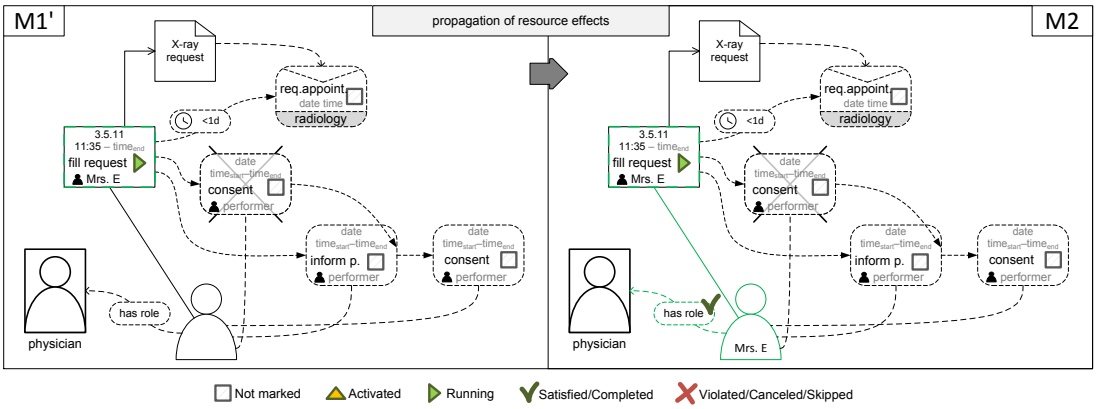


Figure 4.4: Resource propagation

are discovered during this *resource propagation*, all the affected elements are marked as *satisfied* (✓) and are colored green accordingly.

Activity and message nodes in state *running* can handle corresponding data flow events when the latter occur. In general, data flow events are handled deterministically, i.e., their processing works on existing markings as illustrated in Fig. 4.5. In particular, a data event annotates a data flow edge with a corresponding data value (e.g. X-ray request form with id 27051 and filled by Mrs. E). Next, this value is propagated to connected data objects. Finally, data conditions and relations are evaluated (cf. Fig. 4.6).

As shown in Fig. 4.7, end events complete the running activity and message nodes they correspond to, i.e., end events change the respective node state to *completed* (✓). When a node becomes completed, subsequent activity and message nodes become *activated* (△) if they solely depend on completed predecessors (cf. Fig. 4.8). In turn, the completion of an activity or message node skips all predecessors depending on the completed node (cf. Fig. 4.9).

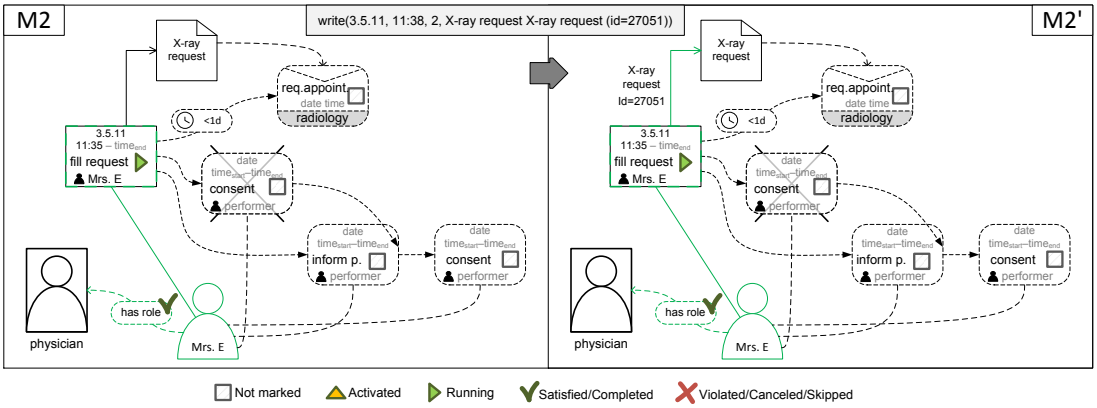


Figure 4.5: Processing data flow events

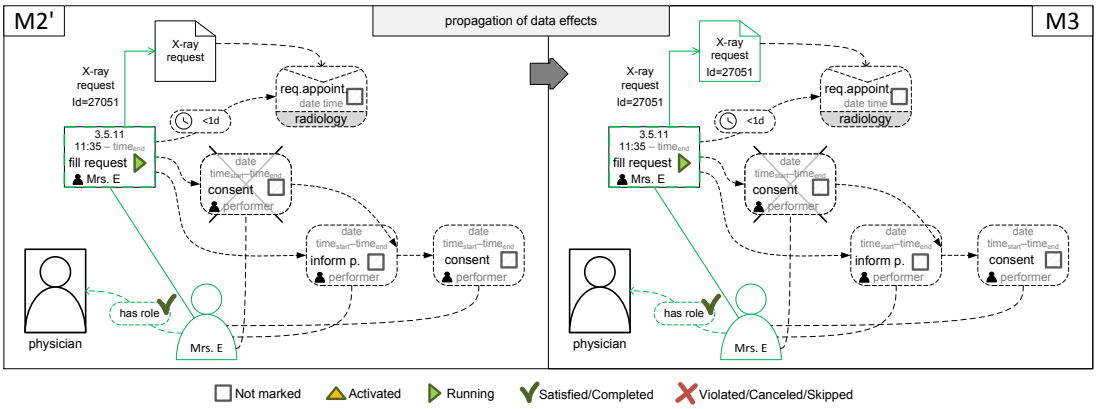


Figure 4.6: Data flow propagation

Due to the non-deterministic processing of start events, multiple markings may match the event to be processed. In this case all eligible markings are processing the event. For example, both *M4* (cf. Fig. 4.8) and *M9* (cf. Fig. 4.9) must handle event `start(3.5.11, 13:15, request informed consent, Mrs. E)`.

No marking results from the event processing matches well to a given compliance rule. In particular, markings may contain *conflicts* indicating a misinterpretation of the events processed. In case of compliance violations, the respective conflicts highlight the reasons why compliance cannot be achieved based on the considered events (cf. CMF9). Moreover, conflicting markings might be ignored when processing further events.

Fig. 4.10 shows an example of a conflict between resources. In particular, the processed start event matches the activated activity `request informed consent`. When propagating resources (cf. Fig. 4.4) the conflict between the expected resource (i.e. *Mrs. E*) and the actual one (i.e. *Mr. G*) is detected. To highlight the conflict all concerned

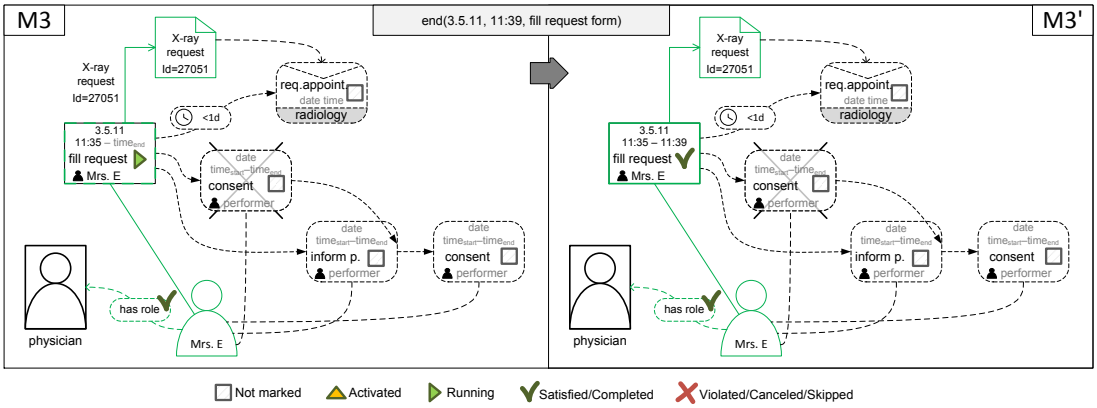


Figure 4.7: Processing end events

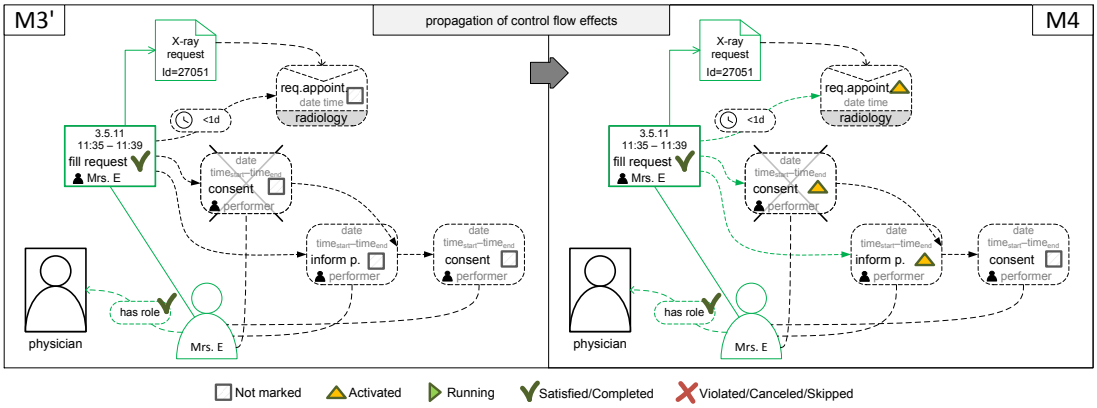


Figure 4.8: Control flow propagation (forwards)

elements of the eCRG are colored red and the corresponding activity node is marked as skipped (×).

To assess compliance (cf. CMF7) the different activations of a compliance rule are determined (cf. CMF6). In particular, an *activation* corresponds to a compliance rule instance that meets the precondition of the respective rule. In the eCRG monitoring framework, *activations* correspond to markings that satisfy exactly the conditions of the antecedence pattern, but ignore any other event not related to the antecedence pattern. Accordingly, all other activity and message nodes are either *not marked* (□) or *skipped* (×) as response to a backward propagation of control flow effects.

Marking M4 (cf. Fig. 4.7) corresponds to an example of an *activation*. It satisfies all conditions of the antecedence pattern, leaving all other activity and message nodes in state *not marked* (□).

In order to determine the compliance state of an activation, all markings extending the

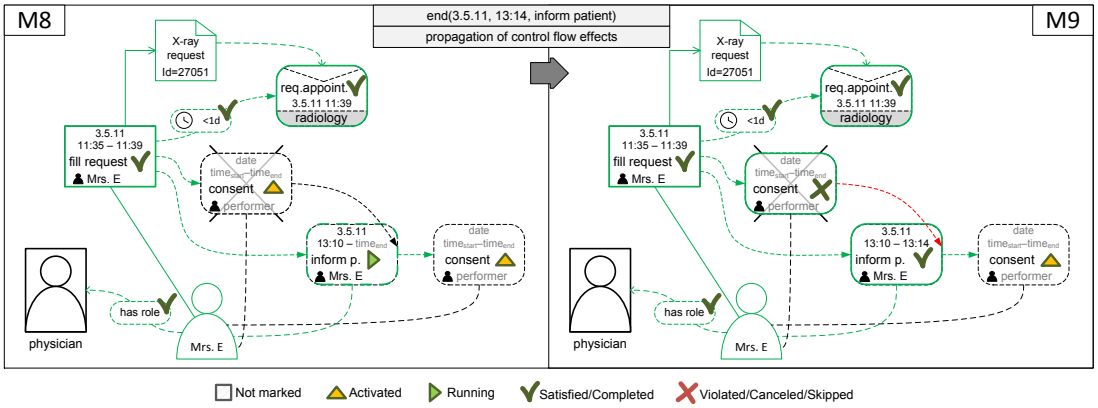


Figure 4.9: Control flow propagation (backwards)

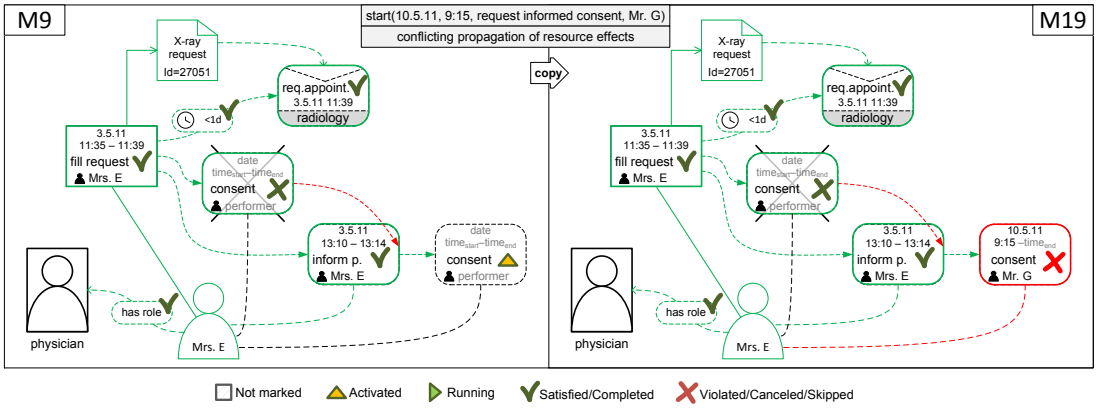


Figure 4.10: Resource conflict

activation are analyzed. These markings include the same annotations of the antecedence pattern as the activation as well as additional annotations for some of the elements that were skipped (×) or *not marked* (□) in the *activation* marking. In particular, the algorithms search for a marking that satisfies all conditions of the respective compliance rule (i.e. *fulfillment*). However, the non-deterministic processing of start events may ignore forbidden events. Consequently, a fulfillment is not sufficient to ensure compliance for a particular activation. In addition, it must be ensured that there is no *violation* extending the fulfillment. Thereby, a *violation* corresponds to a marking under which either required nodes (i.e. consequence occurrence) are skipped or forbidden ones (i.e. consequence absence) are completed. Note that the absence of any violation at a certain point in time does not guarantee for the absence of violations in the future. Accordingly, the compliance state of an activation only becomes *compliant* if a fulfillment exists that cannot be violated anymore. In turn, the compliance state remains *violable compliant* as long as respective fulfillments can be extended to violations. If the activation itself is directly extended by a *violation* (i.e., there exists no marking inbetween), the activation

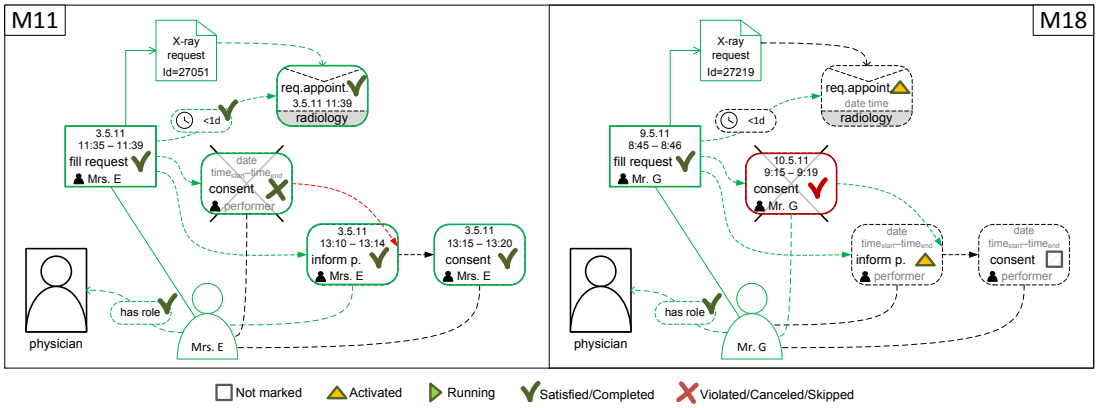


Figure 4.11: Fulfilling and violating markings

enters compliance state *violated*. Finally, Compliance state *pending* refers to *activations* that still require actions to reach a compliance, but have not been violated yet. In state *pending*, markings can be selected that highlight the specific actions demanded by the compliance rule and, in this way, proactively support compliance (cf. CMF8). In turn, conflict and violation markings may be utilized to highlight potential root causes of compliance violations (cf. CMF9).

Examples of a fulfillment (see marking *M11*) as well as a violation (see *M18*) are shown in Fig. 4.11. The fulfillment *M11* ensures that the activation *M4* can be considered as compliant. In turn, violation *M18* directly extends another activation, which then is considered as violated. When achieving marking *M9* after processing Event 12, however, the activation *M4* was still in state *pending*. In particular, *M9* highlights that activity `request informed consent` is required as next action to reach compliance because the respective consequence occurrence activity is activated.

Finally, the eCRG monitoring framework utilizes the compliance states of the different activations for calculating compliance metrics (cf. CMF10). As an example, the *compliance degree* is calculated as percentage of compliant activations of the total activations.

4.3 Evaluation

The eCRG monitoring framework was assessed in two respects. Section 4.3.1 summarizes evaluation results related to the compliance monitoring functionalities (CMF). In turn, the technical feasibility of the approach was demonstrated by implementing a prototype and assessing it (cf. Section 4.3.2).

Table 4.4: Discussion of CMF1 adopted from [KRK17, LMM+15]

CMF 1: Constraints referring to time. *Evaluation criteria*: "To fully support this functionality, the approach must be able to monitor qualitative and quantitative time-related conditions." [LMM+15] *Implementation:* The eCRG framework supports qualitative *and* quantitative time-related conditions. In particular, qualitative time constraints can be expressed through sequence flow edges. Time condition attachments, in turn, which may be attached to activities as well as sequence flows, enable the specification and monitoring of quantitative time constraints.

4.3.1 Evaluating eCRG Monitoring against CMFs

In [KRK17], the eCRG framework was evaluated against the compliance monitoring functionalities (CMF) introduced in [LMM+15]. In particular, [KRK17] discusses the conditions of each CMF. Finally, it shows that all CMFs are satisfied. As an example, Table 4.4 presents the evaluation of the monitoring framework against CMF1 [KRK17].

4.3.2 Proof-of-Concept Prototype

The technical feasibility of the eCRG monitoring framework is demonstrated by a proof-of-concept prototype, which implements the algorithms described in [KRK15b, KRK17]. These algorithms cover the processing of events, propagation of effects, and step-wise assessment of compliance. Finally, the eCRG markings generated during compliance monitoring are visualized.

Using the prototype as well as available process logs, we were able to monitor eCRGs (i.e. compliance rules) from different scenarios. First, eCRGs can be imported from the eCRG modeling component (cf. Section 3.3.4). Second, event logs can be imported and replayed, while concurrently monitoring the corresponding eCRG instances. Fig. 4.12 depicts a screenshot of the proof-of-concept prototype.

The prototype was applied to eCRGs and event logs of different size to obtain performance measures. We could show that the benefits of the eCRG monitoring framework come along with a high computational complexity [KRK17]. In particular, the monitoring of large eCRGs often results in large sets of markings and, hence, becomes time consuming. The results of the performance test runs are summarized in Table 4.5.

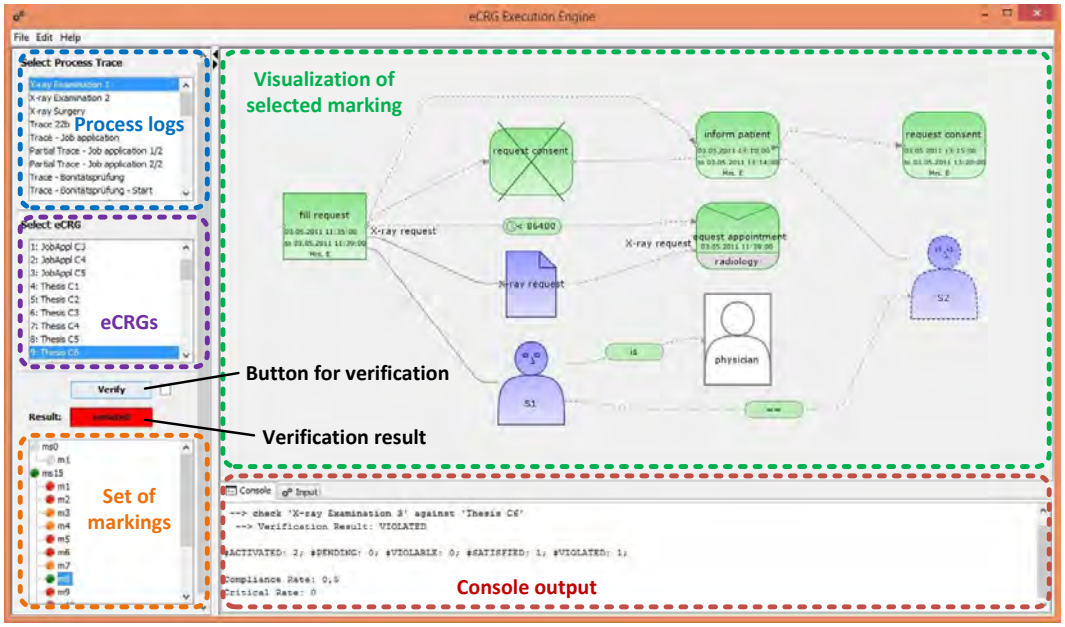


Figure 4.12: eCRG monitoring prototype

Table 4.5: Performance measurement (adopted from [KRK17])

eCRG size			100 events			200 events			300 events			400 events		
	nodes	edges & attachments	time (*ms*)	avg (*ms*)	size	time (*ms*)	avg (*ms*)	size	time (*ms*)	avg (*ms*)	size	time (*ms*)	avg (*ms*)	size
S	3 (2,1)[1]	3 (1,2)[2]	122	1.20	236	433	2.15	876	698	2.32	1,243	**6711**	**11.18**	**7,609**
M	10 (4,6)	15 (3,12)	702	7.16	1,049	9,724	48.6	11,519	**58,758**	**194.6**	**39,914**	-	-	-
L	15 (6,9)	27 (6,21)	3,729	36.9	3,014	**720,236**	**3583.3**	**101,978**	-	-	-	-	-	-

[1] The first entry of the tuple (i.e. 2) refers to activity or message nodes, the second one (i.e. 1) refers to data and resource nodes

[2] The first entry of the tuple (i.e. 1) refers to sequence flow, the second one (i.e. 2) refers to all other edges and attachments

4.4 Related Work

In literature, several frameworks for *compliance monitoring* and *continuous auditing* [AKV08] are discussed.

[NS07] presents a pattern-based approach that enriches process models with a semantic layer of *internal controls*, i.e. compliance checks. In turn, [TRS+11] proposes a framework based on *compliance checkpoints*, i.e., annotations of process models that trigger compliance checks during process execution. The approach presented in [NVN+08] calculates the optimal number and position of controls in a process. For this purpose, the trade-off between costs and risks, depending on the number of checks performed, is considered. Another compliance monitoring framework is presented in [GMP06]. It describes, how compliance can be monitored based on event management middleware.

In turn, [BM05] discusses the monitoring and enforcement of compliance in respect to process collaborations. The detailed architecture of an online auditing tool is described in [AHW+11, Acc12]. It allows monitoring the operations of an organization in detective, corrective and preventive modes. Finally, [ASE15] proposes an architecture enabling process compliance monitoring as a service.

There exists a variety of techniques for compliance monitoring: The compliance monitoring framework presented in [ABE+15] relies on *anti-patterns* and complex event processing. In particular, Esper and the *Event Processing Language* are used to identify situations in which compliance is violated. *Supervisory Control Theory* [SFV+12], in turn, prevents users from performing actions leading to compliance violations. [Seb12] introduces the BPath compliance monitoring approach; BPath expressions combine LTL with hybrid logic and can be transformed into XPath. Furthermore, the Dynamo framework provides a powerful logic for specifying compliance rules, including temporal operations [BBG+07]. Dynamo compensates compliance violations with pre-defined healing strategies [BGP07]. Other compliance monitoring approaches utilize first-order temporal logic [HV08, BKMP08, BHKZ12]. Finally, *Fuzzy Conformance Checking* [BCM+11] calculates an evaluation score instead of providing simple *yes/no* answers. The evaluation score reflects the degree, the considered process instances conform to compliance rules.

[MMA12, MMC+14] transform visual declarative constraints into *Event Calculus* and *Linear Temporal Logic* (LTL) to use them for compliance monitoring. In [MMWA11], colored automata are applied to enable detailed feedback and root cause analyses.

The eCRG monitoring framework [KRK15b, KRK17] developed in the context of this thesis has been adopted from the monitoring of *Compliance Rule Graphs* (CRG) [LRMKD11, Ly13]. CRGs enable fine-grained compliance diagnostics at run time, but focus on the control flow perspective. [GLGRM13] proactively ensures compliance with CRGs by translating them into numerical optimization problems.

[LMM+13, LMM+15] compare approaches for monitoring business process compliance taking 10 compliance monitoring functionalities (CMF) as benchmark. In particular, it is concluded that existing approaches do not provide a solution combining an expressive language (CMF 1-5) with full compliance traceability (CMF 8+9). Table 4.6 summarizes the comparison of compliance monitoring approaches [LMM+15]. In addition, Table 4.6 investigates in what respect the considered works provide a high-level visual language.

A-posteriori compliance checking is related to compliance monitoring as it verifies compliance rules against activity and event logs (cf. Section 2.4). However, corresponding approaches [BBMS12, RFA12, OS16] focus on completed logs. Thus, they need not consider *violable* and *pending* states of compliance rules.

Table 4.6: Classification of monitoring approaches based on CMF (adopted from [LMM+15] and [KRK17])

Approach	CMF 1 time	CMF 2 data	CMF 3 re-source	CMF4 non atomic	CMF 5 life-cycle	CMF 6 multi-inst	CMF 7 react mgmt	CMF 8 proact mgmt	CMF 9 root cause	CMF 10 compl degree	- visual lang
SCT [SFV+12]	+/–	–	+	+	+	–	–	+	–	–	–
ECE Rules [BCM+11]	+	+/–	+	+	–	–	+	–	+/–	+	–
BPath (Sebahi) [Seb12]	+	+	+	+	+/–	+	+	–	–	+/–	–
Gomez et al. [GLGRM13]	+	–	–	+	n.a.	+/–	+	+	–	–	+/–
Giblin et al. [GMP06]	+	n.a.	n.a.	n.a.	n.a.	n.a.	+	n.a.	n.a.	n.a.	–
Narendra et al. [NVN+08]	–	+	+	n.a.	–	+	+	–	–	+	–
Thullner et al. [TRS+11]	+	n.a.	n.a.	n.a.	n.a.	n.a.	+	–	–	n.a.	–
MONPOLY [BKMP08] [BHKZ12]	+	+	+	+/–	+/–	+	+	–	–	–	–
Halle et al. [HV08]	+/–	+	+/–	n.a.	n.a.	n.a.	+	n.a.	n.a.	n.a.	–
Dynamo [BG05, BGP07] [BBG+07]	+	+	+/–	+	n.a.	+	+	–	–	+/–	–
Namiri et al. [NS07]	+/–	+	+	+	–	+	+	–	–	–	–
MobuconEC [MMC+14]	+	+	+	+	+	+	+	–	–	+/–	+/–
MobuconLTL [MMA12] [MMWA11, MWMA12]	+/–	–	–	+	–	–	+	+	+	+/–	+/–
SeaFlows [LRMKD11]	+/–	+/–	+/–	+	+	+	+	+	+	+/–	+
eCRG monitoring	+	+	+	+	+	+	+	+	+	+	+

Caption: + supported, +/–partly supported, – not supported, n.a. cannot be assessed.

4.5 Summary

This chapter introduced the eCRG monitoring framework, which enables the monitoring of running process instances against compliance rules. The latter are visually expressed as extended Compliance Rule Graphs. The eCRG monitoring framework is not only able to continuously and reactively monitor multiple instances of control flow constraints, but additionally allows for compliance rules that address temporal, data and resource aspects plus restrictions on process interactions (i.e. messages exchanged) with business partners. Besides its high expressiveness, the eCRG monitoring framework supports all compliance monitoring functionalities introduced in [LMM+15]. For example, it provides compliance metrics as well as root cause analyses when compliance violations occur. The eCRG monitoring framework was evaluated through a proof-of-concept prototype, which was applied to different use cases and subject to performance measurements.

5

Compliance Checking for Cross-Organizational, Adaptive Processes

The content of this section has been published in the following papers:

[KRM+13]: **D. Knuplesch**, M. Reichert, J. Mangler, S. Rinderle-Ma, and W. Fdhila. Towards compliance of cross-organizational processes and their changes. In *Business Process Management (BPM'12) Workshops*, volume 132 of *LNBIP*, pages 649–661. Springer, 2013

[KRFRM13]: **D. Knuplesch**, M. Reichert, W. Fdhila, and S. Rinderle-Ma. On enabling compliance of cross-organizational business processes. In *11th International Conference on Business Process Management (BPM'13)*, volume 8094 of *LNCS*, pages 146–154. Springer, 2013

[KRP+13]: **D. Knuplesch**, M. Reichert, R. Pryss, W. Fdhila, and S. Rinderle-Ma. Ensuring compliance of distributed and collaborative workflows. In *9th International Conference on Collaborative Computing (CollaborateCom'13)*, pages 133–142. IEEE, 2013

[KFRRM15]: **D. Knuplesch**, W. Fdhila, M. Reichert, and S. Rinderle-Ma. Detecting the effects of changes on the compliance of cross-organizational business processes. In *34th International Conference on Conceptual Modeling (ER'15)*, volume 9381 of *LNCS*, pages 94–107. Springer, 2015

The original articles are provided in Appendices 1, 2, 3 and 7.

[FRMKR15] extended this work to be applicable in a wider and more general context.

5.1 Research Challenges

During the last decade, several approaches for the a-priori compliance checking of intra-organizational business process models emerged [GHSW09, ADW08, KLRM+10] (cf. Section 2.4.2).

Like intra-organizational processes, *cross-organizational processes* as well as the process models describing them are subject to semantic constraints that may stem from various sources [KRM+13]. As opposed to intra-organizational processes, cross-organizational ones involve several autonomous partners (e.g. manufacturer, supplier, and customer) and consist of partner-specific process models as well as interactions between them. Consequently, specifying cross-organizational processes is not a trivial task, but corresponds to a multi-staged process that includes the collaborative definition of an interaction model as well as the distributed modeling of public and private process models (cf. Chapter 2.2). In general, complete knowledge about all private and public models of the cross-organizational business process cannot be presumed. When specifying an interaction model, private and public process models might not have been defined yet. In general, privacy constraints prevent disclosing details on private process models [AW01].

Changes of a cross-organizational process, including its public and private process models [RWR06, FIRMR15], might affect its conformance with the given compliance rules. Consequently, compliance needs to be re-verified when evolving cross-organizational processes. As compliance checking constitutes a costly endeavor [ADW08, KLRM+10], the set of compliance rules to be re-verified should be as minimal as possible.

In general, the compliance of cross-organizational business processes cannot be ensured with existing compliance checking techniques. The latter presume fully known and pre-specified process models. Instead compliance checking techniques for cross-organizational processes need to cope with incomplete process information [KRM+13]:

- When specifying an interaction model, the information on private and public process models might be incomplete.
- Even if all public process models of a cross-organizational business process are known, details of the corresponding private processes remain hidden due to the privacy constraints of the partners. Note that this might even apply if private details are required for compliance assessment. Consequently, an approach is needed that allows partners to additionally make *assertions* on the behavior of their private processes, but without disclosing too many private process details.
- As cross-organizational processes might be subject to change [RBD99, RW12], one needs to be able to effectively determine the compliance rules potentially affected by a process change.

In general, compliance violations should be detected as early as possible. Consequently, techniques are needed for checking compliance of cross-organizational business processes in their different specification phases (cf. Section 2.2).

Example 5.1 (Cross-organizational compliance)
Consider the models of the cross-organizational healthcare process introduced in Chapter 2 as well as compliance rules C7 and C8 (cf. Fig. 5.1). According to C7, before any X-ray examination may take place the respective patient needs to be physically examined and

informed about risks. In turn, C8 disallows making an appointment before sending the informed consent.

The cross-organizational healthcare process complies with C7. This can be decided when considering the private models of the two partners (cf. Fig. 2.7). However, when only having access to the public process models, this cannot be proven as `inform patient` is not included in the public model of the ward unit (cf. Fig. 2.8).

Rule C8 is not obeyed by the cross-organizational process (cf. Fig. 2.7). Note that non-compliance can be derived based on the interaction model (cf. Fig. 2.9).

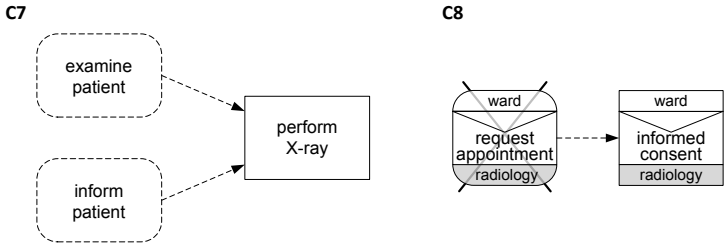


Figure 5.1: Compliance rules C7 and C8

5.2 Scientific Contribution

This section summarizes the contributions made by [KRM+13], [KRFRM13], [KRP+13] and [KFRRM15].

[KRM+13, KRFRM13, KRP+13] introduce an approach that allows specifying and verifying the compliance of a cross-organizational business process at different levels. As outlined in Fig. 5.2, the approach considers three kinds of compliance rules. *Global compliance rules* correspond to constraints that restrict the entire cross-organizational process. In turn, *local compliance rules* refer to the private process of a particular partner. Finally, a partner may make *assertions* to guarantee for a certain behavior of its private process that cannot be derived from its public model. Consequently, a partner has to verify the compliance of its private model against the assertions made. The latter may then be considered as public, but partner-specific (i.e. local) compliance rules.

There exist considerable works that deal with the a-priori compliance checking of intra-organizational processes [LMX07, ADW08, KLRM+10]. These approaches can be directly applied in order to verify both local compliance rules and partner-specific assertions against private (i.e. intra-organizational) process models. Consequently, in the context of this work, it is sufficient to focus on the verification of compliance at the interaction level and the public process models. [KRFRM13, KRP+13] introduced *compliability* as

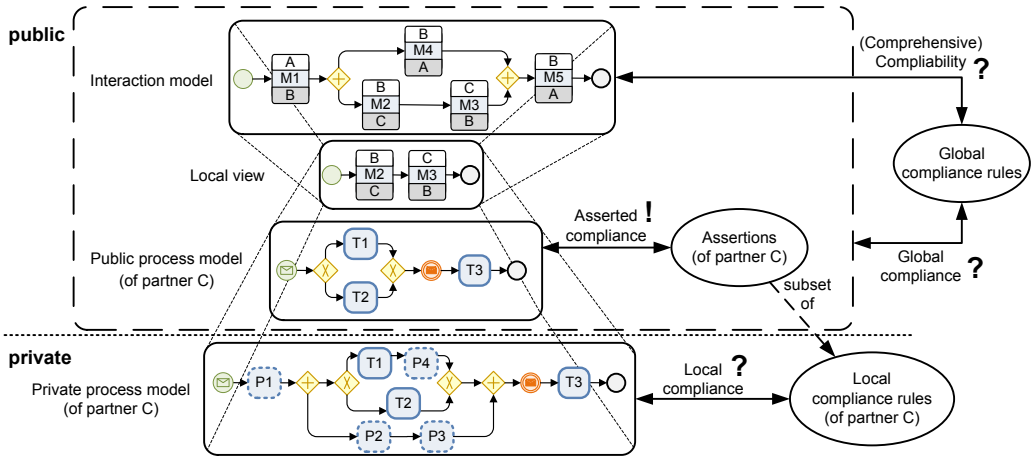


Figure 5.2: Levels of compliance in cross-organizational business processes (adopted from [KRFRM13, KRP+13])

compliance criterion at the interaction model level and *global compliance* as compliance criterion for the public process models.

5.2.1 Compliability

When designing the interaction model of a cross-organizational process, usually, no detailed information on the activities of the overall process has been specified yet. Consequently, when checking the compliance of an interaction model, it cannot be finally decided whether compliance rules are met. However, one can check whether or not the interaction model conflicts with the given compliance rules. In this context, *compliability* shall ensure that an interaction model does not violate any global compliance rule. Verifying compliability, therefore, requires the existence of a trace of interactions (i.e. message exchanges) and activities that complies with the set of global compliance rules on one hand and conforms with the interaction model on the other hand, i.e., the trace can be replayed on the interaction model after removing all activities from it.

Comprehensive compliability, in turn, refines compliability by additionally ensuring that both the entire interaction model and its corresponding interactions do not violate compliance. As opposed to compliability, for each message exchange set out by an interaction model, a corresponding trace is required that involves the selected interaction on one hand and complies with the set of global compliance rules on the other.

Note that compliability is a necessary, but not sufficient criterion for the existence of a compliant realization of an interaction model, i.e., it is still possible that there exists no collaboration of partner processes that implements a (comprehensive) compliable interaction model conforming to all global compliance rules.

5.2.2 Global Compliance

Public process models and their collaboration not only set out the interactions between business partners, but also refer to public activities and define their execution order. However, for privacy reasons, public process models do not mirror every activity of the corresponding private process model. Besides their public process models, however, partners may share additional information on their private processes in terms of semantic *assertions* their private process models comply with. Any assertion, therefore, can be considered as a *filter* selecting permissible traces.

Global compliance, in turn, shall ensure that the collaboration of public processes (with related assertions) meets all global compliance rules. Accordingly, global compliance requires that each trace conforming to the collaboration complies with all global compliance rules. Note that such traces consist of private and public activities as well as interactions. Consequently, conformance means that the trace conforms to all assertions on one hand and can be replayed on the collaboration of public processes after removing all private activities on the other.

As opposed to compliability, *global compliance* is a sufficient, but not necessary condition. It ensures that any possible implementation of a collaboration of public processes, together, with their public assertions, complies with global compliance rules. However, note that a collaboration of private processes might still be compliant with global compliance rules, even if global compliance cannot be ensured.

Example 5.2 (Compliability and global compliance)
Consider the models of the cross-organizational healthcare process (cf. Chapter 2) as well as compliance rules C7 and C8 (cf. Fig. 5.1). In every trace producible on the interaction model (cf. Fig. 2.9), the `X-ray request` is sent before the `transmit informed consent`. Consequently, compliability and comprehensive compliability are violated.

In turn, global compliance can be ensured based on the assertions A1 and A2 of the ward (cf. Fig. 5.3). According to A1, activity `examine patient` is always completed before activity `fill request form` is started. Moreover, A2 ensures that activity `inform patient` is always executed before starting activity `request informed consent`. Consequently, global compliance with C7 can be ensured based on the public process models (cf. Fig. 2.8).

[KRM+13] introduced the distinction between *global compliance rules*, *local compliance rules*, and *assertions*. *Compliability* of interaction models was described in [KRFRM13] for the first time. In turn, formal specifications of *compliability*, *comprehensive compliability*, and *global compliance* as well as techniques for their verification are provided in [KRP+13]. The latter work also contains two proofs showing that compliability constitutes a necessary

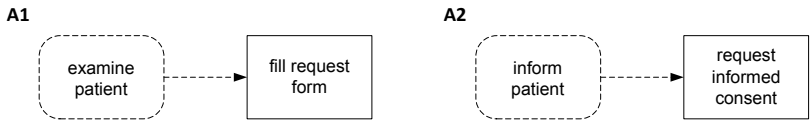


Figure 5.3: Assertions A1 and A2

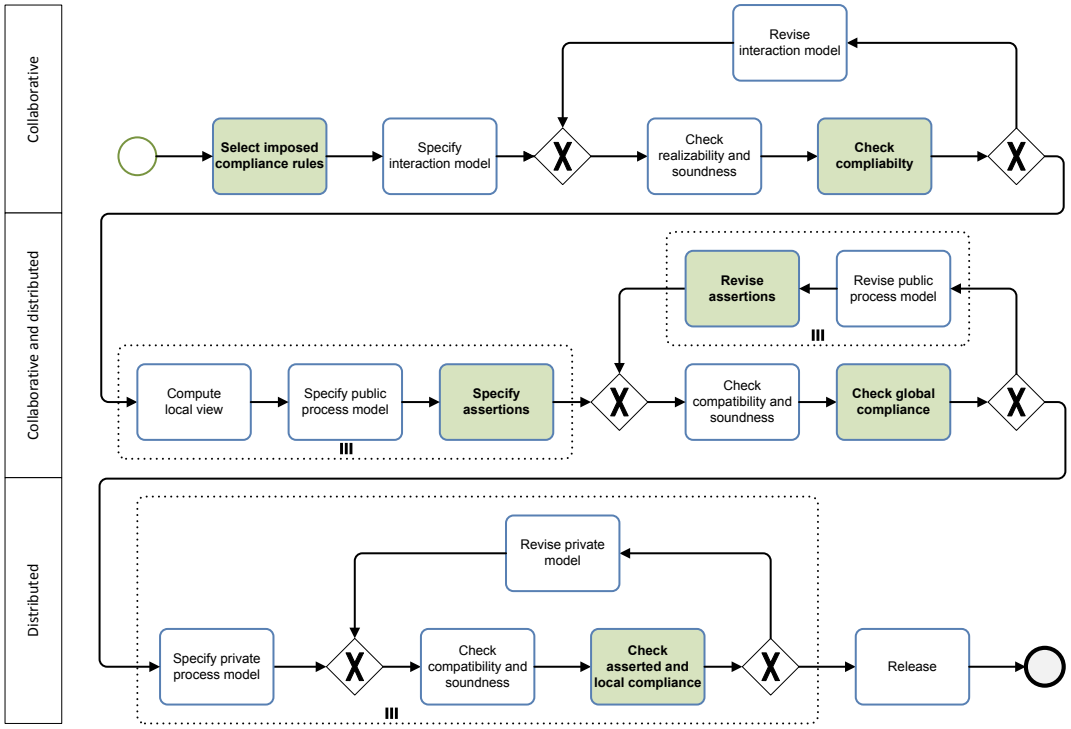


Figure 5.4: Process of modeling cross-organizational processes (adopted from [KRP+13])

condition and global compliance a sufficient one. The criteria were further embedded in the process of modeling cross-organizational processes (cf. Fig. 5.4).

5.2.3 Effects of Process Changes on Global Compliance

Changes of cross-organizational processes (i.e., collaborations of *public processes* with *assertions*) or parts of these processes might affect their compliance as well. Consequently, compliance has to be re-verified whenever changing a cross-organizational process or parts of it. Note that not every change bears the risk of violating global compliance rules. Consequently, one should limit the set of compliance rules that need to be re-verified again.

In general, solely re-verifying those compliance rules that refer to activities directly affected by the change is not sufficient. On one hand, this *naive approach* would recheck

false positive compliance rules that become overfulfilled due to a change. On the other, this approach can even result in *false negatives*, i.e., it is unable to identify all compliance rules that need to be rechecked. Note that changes of cross-organizational processes may not only affect public process models, but also alter private ones. However, only changes of the public models are known. In turn, private models may be arbitrarily modified as long as they conform to corresponding public models and assertions.

Example 5.3 (False Negatives)
Reconsider the scenario from Example 5.2. Though, Compliance rule C7 is not directly affected when removing activity `request informed consent`, this change releases assertion A2 such that global compliance with C7 cannot be ensured anymore. Note that after the change the hospital could remove private activity `inform patient` without violating its public model and assertions.

In [KFRRM15], the dependencies between activities and assertions on one hand and compliance rules on the other are expressed based on the qualified dependency graph (QDG). In particular, a QDG depicts which activities affect which compliance rules or assertions in which sense. Based on this information, the QDG allows us to reliably limit the compliance rules affected by a given change, which is expressed in terms of a change profile (cf. Section 2.1.3). In particular, the algorithms described in [KFRRM15] classify the direct effects of changes on global compliance rules and assertions as positive or negative. Then, the transitive effects on assertions are determined. Since assertions might affect each other, even small changes might have a major impact, i.e., require the rechecking of multiple *global compliance rules*. [KFRRM15] is not limited to activity changes, but considers the addition and deletion of assertions at the same time.

Finally, note that [KRM$^+$13, KRFRM13, KRP$^+$13, KFRRM15] focus on the control flow and interaction perspectives of business processes and do not consider the time, data, and resource perspectives.

5.3 Evaluation

5.3.1 Proof-of-Concept Prototype

To feasibility of the approach is demonstrated through a proof-of-concept prototype [KRFRM13, KRP$^+$13, KFRRM15]. The latter was implemented as plug-in of the AristaFlow BPM Suite [DRR$^+$09, DR09, LKRD10] on top of existing libraries provided by the SeaFlows toolset [KLRM$^+$10, LKR$^+$11]. Using the LTL Model Checker SAL [MOR$^+$04] the prototype enables *compliability* and *global compliance checking* with respect to global compliance rules and assertions expressed in linear temporal logic (LTL).

The prototype was further enhanced with features to properly deal with process changes and to specify compliance rules and assertions in terms of eCRGs, which are then transformed into finite automata. Furthermore, qualified dependency graphs are displayed and utilized to calculate those compliance rules that might be affected by a change.

5.3.2 Related Work

Only few approaches address a-priori compliance checking of cross-organizational business processes. [GMS06] checks the compliance of business process with business contracts specified in terms of the Formal Contract Language (FCL). In turn, [ACG+08] introduces a declarative approach for the rule-based specification of cross-organizational business processes. As opposed to our approach, the issue of privacy is not addressed [KRM+13, FRMKR15].

Another stream of related works addresses the interplay between changes and compliance of intra-organizational business processes. [KYC13] enables a comprehensive compliance checking based on Mixed-Integer Programming. Besides a-priori compliance checking, a minimal set of process change operations can be determined to heal detected compliance violations. [KSG13, SK15], in turn, prevent compliance violations through the addition of missing controls at run time. Several other frameworks deal with the compliance of intra-organizational processes after process changes [LRD08, LRMGD12, KR11a, KKR+13]. [LRD08] shows how to determine those compliance rules that might be violated due to the changes of an intra-organizational process. [TWR+15a, TWR+15b] combine model checking and *complex event processing* (CEP) to ensure compliance of adaptive processes at both design and run time.

Finally, [Com14] deals with the *monitorability* of compliance rules in the context of evolving cross-organizational processes. Monitoring requirements are derived from compliance rules and compared with the available information. As opposed to our approach, [Com14] does not check compliance *a priori*, but ensures that the compliance of cross-organizational processes can be monitored during run time.

Other frameworks allow monitoring the compliance of cross-organizational processes at run time as well, e.g., [BM05, MRB+14, Hof13, MBP10].

To the best of our knowledge, the approach presented in this chapter [KRFRM13, KRP+13, KFRRM15] is the first one ensuring the compliance of adaptive cross-organization processes, which takes privacy issues into account as well [FRMKR15].

5.4 Summary

Like intra-organizational business processes, cross-organizational processes are subject to domain-specific compliance rules. [KRFRM13, KRP+13] introduced an approach for

the a-priori compliance checking of cross-organizational, adaptive business processes. In particular, compliance criteria for interaction models and process collaborations as well as techniques for verifying these criteria were introduced. As opposed to intra-organizational compliance checking, the proposed techniques are able to cope with incomplete process information as in the case of cross-organizational processes. As compliance checking is known to be costly, [KFRRM15] limits the number of the compliance checks to be re-applied when changing a cross-organizational process. The presented concepts and techniques were demonstrated through a proof-of-concept prototype that was applied to different scenarios.

6

Summary and Outlook

Enterprises need to ensure the compliance of their business processes with a variety of semantic constraints, also denoted as *compliance rules* [GS09, Ly13]. In general, these rules not only constrain the occurrence and order of activities, but other process perspectives as well. For example, compliance rules may impose temporal constraints or refer to process resources and data [CRRC10]. In the context of cross-organizational business processes, in addition, compliance rules may refer to the interactions between the business partners as well.

To enable the automated verification of business process compliance, a formal specification of compliance rules and corresponding compliance checking techniques are required. As the available process information varies along the phases of the process life cycle, different compliance checking techniques are required [LRD08, KR11a, KRM+13].

This thesis adds three main contributions to the state-of-art on business process compliance. First, it introduces the *extended Compliance Rule Graph* (eCRG) language that enables the visual specification of multi-perspective compliance rules. Second, it presents a sophisticated framework for the monitoring (i.e. run-time compliance checking) of eCRG compliance rules. Third, it provides compliance checking techniques for cross-organizational, adaptive business processes.

The presented contributions can be extended in several respects. For example, the presented compliance checking techniques for cross-organizational business processes can be also applied to collections of process variants [HBR10, RW12, ATW+15]. Furthermore, multi-perspective eCRGs are not bound to activity-centric process models, but might be applied to data-centric processes, which are composed of various interacting processes (e.g., lifecycle and coordination processes) [MRH08, Mül09, KR11b, KWR11, SKAR17, SAR18], as well. Finally, the eCRG compliance monitoring may be combined with predictive analytics in order to predict compliance violations even before they occur during runtime [CFMR16, TDMD16].

Bibliography

[Aa18] H. van der Aa. Comparing and aligning process representations. PhD thesis, Vrije Universiteit Amsterdam, The Netherlands, 2018.

[ABE+15] A. Awad, A. Barnawi, A. Elgammal, R. E. Shawi, A. Almalaise, and S. Sakr. Runtime detection of business process compliance violations: an approach based on anti patterns. In *20th Symposium on Applied Computing (SAC'15)*, pages 1203–1210. ACM, 2015.

[Acc12] R. Accorsi. An approach to data-driven detective internal controls for processaware information systems. In *Workshop on Data Usage Management on the Web (DUMW'12)*, pages 29–33. 2012.

[ACG+08] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, M. Montali, and P. Torroni. Expressing and verifying business contracts with abductive logic programming. *International Journal of Electronic Commerce*, 12(4), pages 9–38, 2008.

[ADW08] A. Awad, G. Decker, and M. Weske. Efficient compliance checking using BPMN-Q and temporal logic. In *6th International Conference on Business Process Management (BPM'08)*, volume 5240 of *LNCS*, pages 326–341. Springer, 2008.

[AH05] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4), pages 245–275, 2005.

[AHW+11] W. M. P. van der Aalst, K. M. van Hee, J. M. E. M. van der Werf, A. Kumar, and M. Verdonk. Conceptual model for online auditing. *Decision Support Systems*, 50(3), pages 636–647, 2011.

[AKV08] M. Alles, A. Kogan, and M. Vasarhelyi. Putting continuous auditing theory into practice: Lessons from two pilot implementations. *Information Systems*, 22(2), pages 195–214, 2008.

[ASE15] A. Awad, S. Sakr, and A. Elgammal. Compliance monitoring as a service: Requirements, architecture and implementation. In *1st International Conference on Cloud Computing (ICCC)*, pages 1–7. IEEE, 2015.

[ATW+15] C. Ayora, V. Torres, B. Weber, M. Reichert, and V. Pelechano. VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems. *Information & Software Technology*, 57, pages 248–276, 2015.

[AW01] W. M. P. van der Aalst and M. Weske. The P2P approach to interorganizational workflows. In *13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068 of *LNCS*, pages 140–156. Springer, 2001.

[AWW09] A. Awad, M. Weidlich, and M. Weske. Specification, verification and explanation of violation for data aware compliance rules. In *7th International Conference on Service-Oriented Computing (ICSOC'09)*, volume 5900 of *LNCS*, pages 500–515. Springer, 2009.

[AWW11] A. Awad, M. Weidlich, and M. Weske. Visually specifying compliance rules and explaining their violations for business processes. *Journal of Visual Languages and Computing*, 22(1), pages 30–55, 2011.

[BBG+07] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini. A timed extension of WSCoL. In *5th International Conference on Web Services (ICWS'07)*, pages 663–670. IEEE, 2007.

[BBMS12] A. Baumgrass, T. Baier, J. Mendling, and M. Strembeck. Conformance checking of RBAC policies in process-aware information systems. In *Business Process Management (BPM'12) Workshops*, volume 100 of *LNBIP*, pages 435–446. Springer, 2012.

[BCM+11] S. Bragaglia, F. Chesani, P. Mello, M. Montali, and D. Sottara. Fuzzy conformance checking of observed behaviour with expectations. In *12th International Conference of the Italian Association for Artificial Intelligence (AI*IA'11)*, volume 6934 of *LNCS*, pages 80–91. Springer, 2011.

[BDH05] A. Barros, M. Dumas, and A. H. M. ter Hofstede. Service interaction patterns. In *3rd International Conference on Business Process Management (BPM'05)*, volume 3649 of *LNCS*, pages 302–318. Springer, 2005.

[BG05] L. Baresi and S. Guinea. Dynamo: Dynamic monitoring of WS-BPEL processes. In *3rd International Conference on Service-Oriented Computing (ICSOC'05)*, volume 3826 of *LNCS*, pages 478–483. Springer, 2005.

[BGP07] L. Baresi, S. Guinea, and L. Pasquale. Self-healing bpel processes with dynamo and the jboss rule engine. In *International Workshop on Engineering of Software Services for Pervasive Environments (ESSPE'07)*, pages 11–20. ACM, 2007.

[BHKZ12] D. Basin, M. Harvan, F. Klaedtke, and E. Zalinescu. MONPOLY: Monitoring usage-control policies. In *3rd International Conference on Runtime Verification (RV'12)*, volume 7186 of *LNCS*, pages 360–364. Springer, 2012.

[BKMP08] D. Basin, F. Klaedtke, S. Müller, and B. Pfitzmann. Runtime monitoring of metric first-order temporal properties. In *28th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'08)*, volume 2 of *LIPIcs*, pages 49–60. Schloss Dagstuhl, 2008.

[BM05] A. Berry and Z. Milosevic. Extending choreography with business contract constraints. *International Journal on Cooperative Information Systems*, 14(2-3), pages 131–179, 2005.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logics of Programs Workshop*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.

[CFMR16] R. Conforti, S. Fink, J. Manderscheid, and M. Röglinger. PRISM - A predictive risk monitoring approach for business processes. In *14th International Conference on Business Process Management (BPM'16)*, volume 9850 of *LNCS*, pages 383–400. Springer, 2016.

[CKCR13] S. Catalkaya, D. Knuplesch, C. M. Chiao, and M. Reichert. Enriching business process models with decision rules. In *Business Process Management (BPM'13) Workshops*, volume 171 of *LNBIP*, pages 198–211. Springer, 2013.

[CKR+15] C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, and A. Ruiz-Cortés. RALph: a graphical notation for resource assignments in business processes. In *27th International Conference on Advanced Information Systems Engineering (CAiSE'15)*, volume 9097 of *LNCS*, pages 53–68. Springer, 2015.

[Com14] M. Comuzzi. Aligning monitoring and compliance requirements in evolving business networks. In *22nd International Conference on Cooperative Information Systems (CoopIS'14)*, volume 8841 of *LNCS*, pages 166–183. Springer, 2014.

[CRC11] C. Cabanillas, M. Resinas, and A. R. Cortés. Defining and analysing resource assignments in business processes with RAL. In *9th International Conference on Service-Oriented Computing (ICSOC'11)*, volume 7084 of *LNCS*, pages 477–486. Springer, 2011.

[CRRC10] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Hints on how to face business process compliance. In *Jornadas de Ingeniería del Software y Bases de Datos (JISBD'10)*, pages 26–32. 2010.

[CTZ+16] C. Czepa, H. Tran, U. Zdun, T. T. T. Kim, E. Weiss, and C. Ruhsam. Plausibility checking of formal business process specifications in linear temporal logic. In *International Conference on Advanced Information Systems Engineering (CAiSE'16) Forum*, pages 1–8. CEUR-WS, 2016.

[DAC99] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *21st International Conference on Software Engineering (ICSE'99)*, pages 411–420. ACM, 1999.

[DR09] P. Dadam and M. Reichert. The ADEPT project: a decade of research and development for robust and flexible process support. *Computer Science - Research and Development*, 23(2), pages 81–97, 2009.

[DRMR18] M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.

[DRR+09] P. Dadam, M. Reichert, S. Rinderle-Ma, K. Göser, U. Kreher, and M. Jurisch. Von ADEPT zur AristaFlow BPM Suite - Eine Vision wird Realität: Correctness by construction und flexible, robuste Ausführung von Unternehmensprozessen. *EMISA Forum*, 29(1), pages 9–28, 2009.

[DSDB15] P. Delfmann, M. Steinhorst, H.-A. Dietrich, and J. Becker. The generic model query language GMQL – conceptual specification, implementation, and runtime evaluation. *Information Systems*, 47(0), pages 129 – 177, 2015.

[DW07] G. Decker and M. Weske. Behavioral consistency for B2B process integration. In *19th International Conference on Advanced Information Systems Engineering (CAiSE'07)*, volume 4495 of *LLNCS*, pages 81–95. 2007.

[FIRMR15] W. Fdhila, C. Indiono, S. Rinderle-Ma, and M. Reichert. Dealing with change in process choreographies: Design and implementation of propagation algorithms. *Information Systems*, 49, pages 1–24, 2015.

[FKR+14] W. Fdhila, D. Knuplesch, J. Reich, S. Rinderle-Ma, and M. Reichert. Message from the CeSCoP 2014 workshop chairs. In *International Enterprise Distributed Object Computing Conference (EDOC'14) Workshops*, page 356. IEEE, 2014.

[FKR+16] W. Fdhila, D. Knuplesch, M. Reichert, M. Fellmann, A. Koschmider, and R. Laue. Message from the CeSCop/ProMoS 2016 workshop chairs. In *International Enterprise Distributed Object Computing Conference (EDOC'16) Workshops*, pages 1–2. IEEE, 2016.

[FRMKR15] W. Fdhila, S. Rinderle-Ma, D. Knuplesch, and M. Reichert. Change and compliance in collaborative processes. In *12th International Conference on Services Computing (SCC'15)*, pages 162–169. IEEE, 2015.

[FSWS11] S. Feja, A. Speck, S. Witt, and M. Schulz. Checkable graphical business process representation. In *15th Symposium on Advances in Databases and Information Systems (ADBIS'11)*, volume 6295 of *LNCS*, pages 176–189. Springer, 2011.

[FZ14] M. Fellmann and A. Zasada. State-of-the-art of business process compliance approaches. In *22nd European Conference on Information Systems (ECIS'14)*. 2014.

[GH15] G. Governatori and M. Hashmi. No time for compliance. In *19th International Enterprise Distributed Object Computing Conference (EDOC'15)*, pages 9–18. IEEE, 2015.

[GHSW09] G. Governatori, J. Hoffmann, S. Sadiq, and I. Weber. Detecting regulatory compliance for business process models through semantic annotations. In *Business Process Management (BPM'08) Workshops*, volume 17 of *LNBIP*, pages 5–17. Springer, 2009.

[GK07] A. K. Ghose and G. Koliadis. Auditing business process compliance. In *5th International Conference on Service-Oriented Computing (ICSOC'07)*, volume 4749 of *LNCS*, pages 169–180. Springer, 2007.

[GKFJ17] P. Gong, D. Knuplesch, Z. Feng, and J. Jiang. bpCMon: A rule-based monitoring framework for business processes compliance. *Web Service Research*, 14(2), pages 81–103, 2017.

[GKR16] P. Gong, D. Knuplesch, and M. Reichert. Rule-based monitoring framework for business process compliance. Technical Report 2016-3, Ulm University, 2016.

[GL00] B. N. Grosof and Y. Labrou. An approach to using XML and a rule-based content language with an agent communication language. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 96–117. Springer, 2000.

[GLGRM13] M. Gomez-Lopez, R. Gasca, and S. Rinderle-Ma. Explaining the incorrect temporal events during business process monitoring by means of compliance rules and model-based diagnosis. In *International Enterprise Distributed Object Computing Conference (EDOC'13) Workshops*, pages 163–172. IEEE, 2013.

[GMP06] C. Giblin, S. Müller, and B. Pfitzmann. From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Technical Report RZ-3662, IBM, 2006.

[GMS06] G. Governatori, Z. Milosevic, and S. Sadiq. Compliance checking between business processes and business contracts. In *10th International Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 221–232. 2006.

[GRRA06] C. Günther, S. Rinderle, M. Reichert, and W. M. P. van der Aalst. Change mining in adaptive process management systems. In *14th International Conference on Cooperative Systems Conferences (CoopIS'06)*, pages 309–326. 2006.

[GS09] G. Governatori and S. Sadiq. The journey to business process compliance. In *Handbook of Research on BPM*, pages 426–454. IGI Global, 2009.

[HBR10] A. Hallerbach, T. Bauer, and M. Reichert. Capturing variability in business process models: the Provop approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7), pages 519–546, 2010.

[HBZ+16] C. Haisjackl, I. Barba, S. Zugal, P. Soffer, I. Hadar, M. Reichert, J. Pinggera, and B. Weber. Understanding declare models: strategies, pitfalls, empirical results. *Software and System Modeling*, 15(2), pages 325–352, 2016.

[Her96] H. Herbst. Business rules in systems analysis: A meta-model and repository system. *Information Systems*, 21(2), pages 147 – 166, 1996.

[HMS12] T. Hildebrandt, R. Mukkamala, and T. Slaats. Nested dynamic condition response graphs. In *4th International Conference on Fundamentals of Software Engineering (FSEN'12)*, volume 7141 of *LNCS*, pages 343–350. Springer, 2012.

[Hof13] W. Hofman. Compliance management by business event mining in supply chain networks. In *7th International Workshop on Value Modeling and Business Ontologies (VMBO'13)*. TU Delft, 2013.

[HR04] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press, 2004.

[Hul08] R. Hull. Artifact-centric business process models: Brief survey of research results and challenges. In *7th International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'08)*, volume 5332 of *LNCS*, pages 1152–1163. Springer, 2008.

[HV08] S. Hallé and R. Villemaire. Runtime monitoring of message-based workflows with data. In *12th International Enterprise Distributed Object Computing Conference (EDOC'08)*, pages 63–72. IEEE, 2008.

[HZ14] C. Haisjackl and S. Zugal. Investigating differences between graphical and textual declarative process models. In *Advanced Information Systems Engineering (CAiSE'14) Workshops*, volume 178 of *LNBIP*, pages 194–206. Springer, 2014.

[KFRRM15] D. Knuplesch, W. Fdhila, M. Reichert, and S. Rinderle-Ma. Detecting the effects of changes on the compliance of cross-organizational business processes. In *34th International Conference on Conceptual Modeling (ER'15)*, volume 9381 of *LNCS*, pages 94–107. Springer, 2015.

[KKR+13] F. Koetter, M. Kochanowski, T. Renner, C. Fehling, and F. Leymann. Unifying compliance management in adaptive environments through variability descriptors. In *6th International Conference on Service-Oriented Computing and Applications (SOCA'13)*, pages 214–219. 2013.

[KLRM+10] D. Knuplesch, L. T. Ly, S. Rinderle-Ma, H. Pfeifer, and P. Dadam. On enabling data-aware compliance checking of business process models. In *29th International Conference on Conceptual Modeling (ER'10)*, volume 6412 of *LNCS*, pages 332–346. Springer, 2010.

[KPR12a] D. Knuplesch, R. Pryss, and M. Reichert. Data-aware interaction in distributed and collaborative workflows: Modeling, semantics, correctness. In *8th International Conference on Collaborative Computing (CollaborateCom'12)*, pages 223–232. IEEE, 2012.

[KPR12b] D. Knuplesch, R. Pryss, and M. Reichert. A formal framework for data-aware process interaction models. Technical Report 2012-06, Ulm University, 2012.

[KR11a] D. Knuplesch and M. Reichert. Ensuring business process compliance along the process life cycle. Technical Report 2011-06, Ulm University, 2011.

[KR11b] V. Künzle and M. Reichert. PHILharmonicFlows: towards a framework for object-aware process management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4), pages 205–244, 2011.

[KR14] D. Knuplesch and M. Reichert. An operational semantics for the extended compliance rule graph language. Technical Report 2014-6, Ulm University, 2014.

[KR17] D. Knuplesch and M. Reichert. A visual language for modeling multiple perspectives of business process compliance rules. *Software and Systems Modeling*, 16(3), pages 715–736, 2017.

[KRFRM13] D. Knuplesch, M. Reichert, W. Fdhila, and S. Rinderle-Ma. On enabling compliance of cross-organizational business processes. In *11th International Conference on Business Process Management (BPM'13)*, volume 8094 of *LNCS*, pages 146–154. Springer, 2013.

[KRK15a] D. Knuplesch, M. Reichert, and A. Kumar. Towards visually monitoring multiple perspectives of business process compliance. In *Advanced Information Systems Engineering (CAiSE'15) Forum*, pages 41–48. CEUR-WS, 2015.

[KRK15b] D. Knuplesch, M. Reichert, and A. Kumar. Visually monitoring multiple perspectives of business process compliance. In *13th International Conference on Business Process Management (BPM'15)*, volume 9253 of *LNCS*, pages 263–279. Springer, 2015.

[KRK17] D. Knuplesch, M. Reichert, and A. Kumar. A framework for visually monitoring business process compliance. *Information Systems*, 64, pages 381 – 409, 2017.

[KRL+13a] D. Knuplesch, M. Reichert, L. T. Ly, A. Kumar, and S. Rinderle-Ma. On the formal semantics of the extended compliance rule graph. Technical Report 2013-05, Ulm University, 2013.

[KRL+13b] D. Knuplesch, M. Reichert, L. T. Ly, A. Kumar, and S. Rinderle-Ma. Visual modeling of business process compliance rules with the support of multiple perspectives. In *32nd International Conference on Conceptual Modeling (ER'13)*, volume 8217 of *LNCS*, pages 106–120. Springer, 2013.

[KRM+13] D. Knuplesch, M. Reichert, J. Mangler, S. Rinderle-Ma, and W. Fdhila. Towards compliance of cross-organizational processes and their changes. In *Business Process Management (BPM'12) Workshops*, volume 132 of *LNBIP*, pages 649–661. Springer, 2013.

[KRP+13] D. Knuplesch, M. Reichert, R. Pryss, W. Fdhila, and S. Rinderle-Ma. Ensuring compliance of distributed and collaborative workflows. In *9th International Conference on Collaborative Computing (CollaborateCom'13)*, pages 133–142. IEEE, 2013.

[KRS96a] I. Konyen, M. Reichert, and B. Schultheiß. Organisationsstrukturen einer Universitätsklinik am Beispiel der Uni-Frauenklinik Ulm. Technical Report DBIS-18, University of Ulm, 1996.

[KRS+96b] I. Konyen, M. Reichert, B. Schultheiß, S. Frank, and R. Mangold. Prozeßentwurf für den Bereich der minimal invasiven Chirurgie. Technical Report DBIS-14, University of Ulm, 1996.

[KSB15] A. Kumar, S. R. Sabbella, and R. R. Barton. Managing controlled violation of temporal process constraints. In *13th International Conference on Business Process Management (BPM'15)*, volume 9253 of *LNCS*, pages 280–296. Springer, 2015.

[KSG13] K. Kittel, S. Sackmann, and K. Göser. Flexibility and compliance in workflow systems - the KitCom prototype. In *Advanced Information Systems Engineering (CAiSE'13) Forum*, pages 154–160. CEUR-WS, 2013.

[KSR96a] I. Konyen, B. Schultheiß, and M. Reichert. Prozeßentwurf eines Ablaufs im Labor. Technical Report DBIS-16, University of Ulm, 1996.

[KSR96b] I. Konyen, B. Schultheiß, and M. Reichert. Prozeßentwurf für den Ablauf einer radiologischen Untersuchung. Technical Report DBIS-15, University of Ulm, 1996.

[KWR11] V. Künzle, B. Weber, and M. Reichert. Object-aware business processes: Fundamental requirements and their support in existing approaches. *International Journal of Information System Modeling and Design*, 2(2), pages 19–46, 2011.

[KYC13] A. Kumar, W. Yao, and C. Chu. Flexible process compliance with semantic constraints using mixed-integer programming. *INFORMS Journal on Computing*, 25(3), pages 543–559, 2013.

[LKR+10] L. T. Ly, D. Knuplesch, S. Rinderle-Ma, K. Göser, M. Reichert, and P. Dadam. SeaFlows toolset - compliance verification made easy. In *Advanced Information Systems Engineering (CAiSE'10) Forum*, pages 26–33. CEUR-WS, 2010.

[LKR+11] L. T. Ly, D. Knuplesch, S. Rinderle-Ma, K. Göser, H. Pfeifer, M. Reichert, and P. Dadam. SeaFlows toolset – compliance verification made easy for process-aware information systems. In *Information Systems Evolution: Advanced Information Systems Engineering (CAiSE'10) Forum – Selected Extended Papers*, volume 72 of *LNBIP*, pages 76–91. Springer, 2011.

[LKRD10] A. Lanz, U. Kreher, M. Reichert, and P. Dadam. Enabling process support for advanced applications with the AristaFlow BPM Suite. In *Business Process Management (BPM'10) Demonstration Track*. CEUR-WS, 2010.

[LMM+13] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. P. van der Aalst. A framework for the systematic comparison and evaluation of compliance monitoring approaches. In *17th International Enterprise Distributed Object Computing Conference (EDOC'13)*, pages 7–16. IEEE, 2013.

[LMM+15] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information Systems*, 54, pages 209–234, 2015.

[LMX07] Y. Liu, S. Müller, and K. Xu. A static compliance-checking framework for business process models. *IBM Systems Journal*, 46(2), pages 335–261, 2007.

[LR07] R. Lenz and M. Reichert. IT support for healthcare processes - premises, challenges, perspectives. *Data and Knowledge Engineering*, 61(1), pages 39–58, 2007.

[LR16] M. Lohrmann and M. Reichert. Effective application of process improvement patterns to business processes. *Software and System Modeling*, 15(2), pages 353–375, 2016.

[LRD08] L. T. Ly, S. Rinderle, and P. Dadam. Integration and verification of semantic constraints in adaptive process management systems. *Data and Knowledge Engineering*, 64(1), pages 3–23, 2008.

[LRMD10] L. T. Ly, S. Rinderle-Ma, and P. Dadam. Design and verification of instantiable compliance rule graphs in process-aware information systems. In *22nd International Conference on Advanced Information Systems*

Engineering (CAiSE'10), volume 6051 of *LNCS*, pages 9–23. Springer, 2010.

[LRMGD12] L. T. Ly, S. Rinderle-Ma, K. Göser, and P. Dadam. On enabling integrated process compliance with semantic constraints in process management systems. *Information Systems Frontiers*, 14(2), pages 195–219, 2012.

[LRMKD11] L. T. Ly, S. Rinderle-Ma, D. Knuplesch, and P. Dadam. Monitoring business process compliance using compliance rule graphs. In *International Conference on Cooperative Systems Conferences (CoopIS'11)*, volume 7044 of *LNCS*, pages 82–99. Springer, 2011.

[LRW16] A. Lanz, M. Reichert, and B. Weber. Process time patterns: A formal foundation. *Information Systems*, 57, pages 38–68, 2016.

[LWR14] A. Lanz, B. Weber, and M. Reichert. Time patterns for process-aware information systems. *Requirements Engineering*, 19(2), pages 113–141, 2014.

[Ly13] L. T. Ly. SeaFlows - a compliance checking framework for supporting the process lifecycle. PhD Thesis, Ulm University, Germany, 2013.

[MBP10] W. MingXue, K. Bandara, and C. Pahl. Distributed aspect-oriented service composition for business compliance governance with public service processes. In *5th International Conference on Internet and Web Applications and Services (ICIW'10)*, pages 339–344. IEEE, 2010.

[MGKR15] N. Mundbrod, G. Grambow, J. Kolb, and M. Reichert. Context-aware process injection: Enhancing process flexibility by late extension of process instances. In *23rd International Conference on Cooperative Information Systems (CoopIS'15)*, volume 9415 of *LNCS*, pages 127–145. Springer, 2015.

[MHHR06] D. Müller, J. Herbst, M. Hammori, and M. Reichert. IT support for release management processes in the automotive industry. In *4th International Conference on Business Process Management (BPM'06)*, volume 4102 of *LNCS*, pages 368–377. Springer, 2006.

[MMA12] F. Maggi, M. Montali, and W. M. P. van der Aalst. An operational decision support framework for monitoring business constraints. In *15th International Conference on Fundamental Approaches to Software Engineering (FASE'12)*, volume 7212 of *LNCS*, pages 146–162. Springer, 2012.

[MMC+14] M. Montali, F. M. Maggi, F. Chesani, P. Mello, and W. M. P. van der Aalst. Monitoring business constraints with the event calculus. *ACM Transactions on Intelligent Systems and Technology*, 5(1), pages 17:1–17:30, 2014.

[MMWA11] F. Maggi, M. Montali, M. Westergaard, and W. M. P. van der Aalst. Monitoring business constraints with linear temporal logic: an approach based on colored automata. In *International Conference on Business Process Management (BPM'11)*, volume 6896 of *LNCS*, pages 132–147. Springer, 2011.

[Mon10] M. Montali. *Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach*, volume 56 of *LNBIP*. Springer, 2010.

[Moo09] D. L. Moody. The physics of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6), pages 756–779, 2009.

[MOR+04] L. M. de Moura, S. Owre, H. Rueß, J. M. Rushby, N. Shankar, M. Sorea, and A. Tiwari. SAL 2. In *16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 496–500. Springer, 2004.

[MRB+14] T. Metzke, A. Rogge-Solti, A. Baumgrass, J. Mendling, and M. Weske. Enabling semantic complex event processing in the domain of logistics. In *Service-Oriented Computing (ICSOC'13) Workshops - Revised Selected Papers*, volume 8377 of *LNCS*, pages 419–431. Springer, 2014.

[MRH08] D. Müller, M. Reichert, and J. Herbst. A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In *20th International Conference on Advanced Information Systems Engineering (CAiSE'08)*, volume 5074 of *LNCS*, pages 48–63. Springer, 2008.

[Mül09] D. Müller. Management datengetriebener Prozessstrukturen. PhD thesis, Ulm University, Germany, 2009.

[MWMA12] F. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst. Runtime verification of ltl-based declarative process models. In *2nd International Conference on Runtime Verification (RV'11)*, volume 7186 of *LNCS*, pages 131–146. Springer, 2012.

[Ngu13] M. T. Nguyen. Modellierung, Pflege und Compliance-Regeln von Studienprozessen. Bachelor thesis, Ulm University, Germany, 2013.

[NK16] M. T. Nguyen and D. Knuplesch. eCRG Evaluation. Project Report, Institute of Databases and Information Systems, Ulm University, 2016.

[NS07] K. Namiri and N. Stojanovic. Pattern-Based design and validation of business process compliance. In *International Conference on Cooperative Systems Conferences (CoopIS'07)*, volume 4803 of *LNCS*, pages 59–76. Springer, 2007.

[NVN+08] N. Narendra, V. Varshney, S. Nagar, M. Vasa, and A. Bhamidipaty. Optimal control point selection for continuous business process compliance monitoring. In *3rd International Conference on Service Operations and Logistics, and Informatics (SOLI'08)*, pages 2536–2541. IEEE, 2008.

[OFR+12] A. Ottensooser, A. Fekete, H. A. Reijers, J. Mendling, and C. Menictas. Making sense of business process descriptions: An experimental comparison of graphical and textual notations. *Journal of Systems and Software*, 85(3), pages 596–606, 2012.

[OMG14] OMG. BPMN 2.0. Recommendation, OMG, 2014.

[OMG17] OMG. SBVR 1.4. Recommendation, OMG, 2017.

[OS16] N. Outmazgin and P. Soffer. A process mining-based analysis of business process work-arounds. *Software and Systems Modeling*, 15(2), pages 309–323, 2016.

[Pnu77] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE, 1977.

[PSA07] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst. DECLARE: full support for loosely-structured processes. In *11th International Enterprise Distributed Object Computing Conference (EDOC'07)*, pages 287–300. IEEE, 2007.

[RAHE05] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In *17th International Conference on Advanced Information Systems Engineering (CAiSE'05)*, volume 3520 of *LNCS*, pages 216–232. Springer, 2005.

[RBD99] M. Reichert, T. Bauer, and P. Dadam. Enterprise-wide and cross-enterprise workflow management: Challenges and research issues for adaptive workflows. In *Workshop Informatik '99*, pages 56–64. CEUR-WS, 1999.

[RD97] M. Reichert and P. Dadam. A framework for dynamic changes in workflow management systems. In *8th International Workshop on Database and Expert Systems Applications (DEXA '97)*, pages 42–48. IEEE, 1997.

[RD98] M. Reichert and P. Dadam. ADEPT$_{flex}$ – supporting dynamic changes of workflows without losing control. *Intelligent Information Systems*, 10(2), pages 93–129, 1998.

[Rei00] M. Reichert. Dynamische Ablaufänderungen in Workflow-Management-Systemen. PhD thesis, Ulm University, Germany, 2000.

[Rei12] M. Reichert. Process and data: Two sides of the same coin. In *20th International Conference on Cooperative Information Systems (CoopIS'12)*, volume 7565 of *LNCS*, pages 2–19. Springer, 2012.

[RFA12] E. Ramezani, D. Fahland, and W. M. P. van der Aalst. Where did I misbehave? Diagnostic information in compliance checking. In *International Conference on Business Process Management (BPM'12)*, volume 7481 of *LNCS*, pages 262–278. Springer, 2012.

[RFA14] E. Ramezani, D. Fahland, and W. M. P. van der Aalst. Supporting domain experts to select and configure precise compliance rules. In *Business Process Management (BPM'13) Workshops*, volume 171 of *LNCS*, pages 498–512. Springer, 2014.

[RFDA13] E. Ramezani Taghiabadi, D. Fahland, B. F. van Dongen, and W. M. P. van der Aalst. Diagnostic information for compliance checking of temporal compliance requirements. In *25th International Conference on Advanced Information Systems Engineering (CAiSE'13)*, volume 7908 of *LNCS*, pages 304–320. Springer, 2013.

[RFWM12] E. Ramezani, D. Fahland, J. M. van der Werf, and P. Mattheis. Separating compliance management and business process management. In *Business Process Management (BPM'11) Workshops*, LNBIP, pages 459–464. Springer, 2012.

[RHD98] M. Reichert, C. Hensinger, and P. Dadam. Supporting adaptive workflows in advanced application environments. In *EDBT Workshop on Workflow Management Systems*, pages 100–109. 1998.

[RJR07] S. Rinderle, M. Jurisch, and M. Reichert. On deriving net change information from change logs - the DELTALAYER-algorithm. In *GI-Fachtagung Datenbanksysteme in Business, Technologie und Web (BTW'07)*, volume 103 of *LNI*, pages 364–381. GI, 2007.

[RRD04] S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems – a survey. *Data and Knowledge Engineering*, 50(1), pages 9–34, 2004.

[RRKD05] M. Reichert, S. Rinderle, U. Kreher, and P. Dadam. Adaptive process management with ADEPT2. In *International Conference on Data Engineering (ICDE'05)*, pages 1113–1114. 2005.

[RRMD09] M. Reichert, S. Rinderle-Ma, and P. Dadam. Flexibility in process-aware information systems. In *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, volume 5460 of *LNCS*, pages 115–135. Springer, 2009.

[RW12] M. Reichert and B. Weber. *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer, 2012.

[RWR06] S. Rinderle, A. Wombacher, and M. Reichert. Evolution of process choreographies in DYCHOR. In *14th International Conference on Cooperative Information Systems (CoopIS'06)*, volume 4275 of *LNCS*, pages 273–290. Springer, 2006.

[SAR18] S. Steinau, K. Andrews, and M. Reichert. The relational process structure. In *30th International Conference on Advanced Information Systems Engineering (CAiSE'18)*, volume 10816 of *LNCS*, pages 53–67. Springer, 2018.

[Seb12] S. Sebahi. Monitoring business process compliance : a view based approach. PhD thesis, University Claude Bernard, Lyon, France, 2012.

[Sem13] F. Semmelrodt. Modellierung klinischer Prozesse und Compliance Regeln mittels BPMN 2.0 und eCRG. Master thesis, Ulm University, Germany, 2013.

[SFV+12] E. A. Santos, R. Francisco, A. Vieira, E. de F.R. Loures, and M. Busetti. Modeling business rules for supervisory control of process-aware information systems. In *Business Process Management (BPM'11) Workshops*, volume 100 of *LNBIP*, pages 447–458. Springer, 2012.

[SGN07] S. Sadiq, G. Governatori, and K. Naimiri. Modeling control objectives for business process compliance. In *5th International Conference on Business Process Management (BPM'07)*, volume 4714 of *LNCS*, pages 149–164. Springer, 2007.

[SK15] S. Sackmann and K. Kittel. Flexible workflows and compliance: A solvable contradiction?! In *BPM - Driving Innovation in a Digital World*, Management for Professionals, pages 247–258. Springer, 2015.

[SKAR17] S. Steinau, V. Künzle, K. Andrews, and M. Reichert. Coordinating business processes using semantic relationships. In *19th International Conference on Business Informatics (CBI'17)*, pages 33–42. IEEE, 2017.

[SKR14] F. Semmelrodt, D. Knuplesch, and M. Reichert. Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In *Enterprise, Business-Process and Information Systems Modeling (BPMDS / EMMSAD'14)*, volume 175 of *LNBIP*, pages 48–63. Springer, 2014.

[SMM+96a] B. Schultheiß, J. Meyer, R. Mangold, T. Zemmler, and M. Reichert. Prozeßentwurf am Beispiel eines Ablaufs aus dem OP-Bereich - Ergebnisse einer Analyse an der Universitätsfrauenklinik Ulm. Technical Report DBIS-6, University of Ulm, 1996.

[SMM+96b] B. Schultheiß, J. Meyer, R. Mangold, T. Zemmler, and M. Reichert. Prozeßentwurf für den Ablauf einer ambulanten Chemotherapie. Technical Report DBIS-7, University of Ulm, 1996.

[SMM+96c] B. Schultheiß, J. Meyer, R. Mangold, T. Zemmler, and M. Reichert. Prozeßentwurf für den Ablauf einer stationären Chemotherapie. Technical Report DBIS-5, University of Ulm, 1996.

[TDMD16] I. Teinemaa, M. Dumas, F. M. Maggi, and C. Di Francescomarino. Predictive business process monitoring with structured and unstructured data. In *14th International Conference on Business Process Management (BPM'16)*, volume 9850 of *LNCS*, pages 401–417. Springer, 2016.

[TEvP12] O. Turetken, A. Elgammal, W.-J. van den Heuvel, and M. Papazoglou. Capturing compliance requirements: A pattern-based approach. *IEEE Software*, 29(3), pages 29–36, 2012.

[TRS+11] R. Thullner, S. Rozsnyai, J. Schiefer, H. Obweger, and M. Suntinger. Proactive business process compliance monitoring with event-based systems. In *International Enterprise Distributed Object Computing Conference (EDOC'11) Workshops*, pages 429–437. IEEE, 2011.

[TWR+15a] T. Tran, E. Weiss, C. Ruhsam, C. Czepa, H. Tran, and U. Zdun. Embracing process compliance and flexibility through behavioral consistency checking in ACM: A repair service management case. In *Business Process Management (BPM'15) Workshops*. 2015.

[TWR+15b] T. Tran, E. Weiss, C. Ruhsam, C. Czepa, H. Tran, and U. Zdun. Enabling flexibility of business processes by compliance rules – a case study from the insurance industry. In *13th International Conference on Business Process Management (BPM'15) Industry Track*, pages 30–43. CEUR-WS, 2015.

[Wes12] M. Weske. *Business Process Management: Concepts, Languages, Architectures – 2nd Edition*. Springer, 2012.

[WPTR13] B. Weber, J. Pinggera, V. Torres, and M. Reichert. Change patterns in use: A critical evaluation. In *Enterprise, Business-Process and Information Systems Modeling (BPMDS/EMMSAD'13)*, volume 147 of *LNBIP*, pages 261–276. 2013.

[WRRM08] B. Weber, M. Reichert, and S. Rinderle-Ma. Change patterns and change support features - Enhancing flexibility in process-aware information systems. *Data and Knowledge Engineering*, 66(3), pages 438–466, 2008.

[WSR09] B. Weber, S. Sadiq, and M. Reichert. Beyond rigidity–dynamic process lifecycle support. *Computer Science - Research and Development*, 23(2), pages 47–65, 2009.

[ZF15] A. Zasada and M. Fellmann. A pattern-based approach to transform natural text from laws into compliance controls in the food industry. In *Lernen, Wissen, Adaption (LWA'15) Workshops*, pages 230–238. CEUR-WS, 2015.

[ZSH+15] S. Zugal, P. Soffer, C. Haisjackl, J. Pinggera, M. Reichert, and B. Weber. Investigating expressiveness and understandability of hierarchy in declarative business process models. *Software and System Modeling*, 14(3), pages 1081–1103, 2015.