software

MDPI

Article

Evaluation of Compliance Rule Languages for Modelling Regulatory Compliance Requirements

Andrea Zasada [1], Mustafa Hashmi [2,3,*], Michael Fellmann [1] and David Knuplesch [4]

[1] Institude of Computer Science, University of Rostock, 18057 Rostock, Germany
[2] La Trobe LawTech, La Trobe Law School, La Trobe University, Melbourne, VIC 3086, Australia
[3] Institute of Law and Technology, Autonomous University of Barcelona (IDT-UAB), 08193 Bellaterra, Spain
[4] alphaQuest GmbH, 89077 Ulm, Germany
* Correspondence: m.hashmi@latrobe.edu.au

Abstract: Compliance in business processes has become a fundamental requirement given the constant rise in regulatory requirements and competitive pressures that have emerged in recent decades. While in other areas of business process modelling and execution, considerable progress towards automation has been made (e.g., process discovery, executable process models), the interpretation and implementation of compliance requirements is still a highly complex task requiring human effort and time. To increase the level of "mechanization" when implementing regulations in business processes, compliance research seeks to formalize compliance requirements. Formal representations of compliance requirements should, then, be leveraged to design correct process models and, ideally, would also serve for the automated detection of violations. To formally specify compliance requirements, however, multiple process perspectives, such as control flow, data, time and resources, have to be considered. This leads to the challenge of representing such complex constraints which affect different process perspectives. To this end, current approaches in business process compliance make use of a varied set of languages. However, every approach has been devised based on different assumptions and motivating scenarios. In addition, these languages and their presentation usually abstract from real-world requirements which often would imply introducing a substantial amount of domain knowledge and interpretation, thus hampering the evaluation of their expressiveness. This is a serious problem, since comparisons of different formal languages based on real-world compliance requirements are lacking, meaning that users of such languages are not able to make informed decisions about which language to choose. To close this gap and to establish a uniform evaluation basis, we introduce a running example for evaluating the expressiveness and complexity of compliance rule languages. For language selection, we conducted a literature review. Next, we briefly introduce and demonstrate the languages' grammars and vocabularies based on the representation of a number of legal requirements. In doing so, we pay attention to semantic subtleties which we evaluate by adopting a normative classification framework which differentiates between different deontic assignments. Finally, on top of that, we apply Halstead's well-known metrics for calculating the relevant characteristics of the different languages in our comparison, such as the volume, difficulty and effort for each language. With this, we are finally able to better understand the lexical complexity of the languages in relation to their expressiveness. In sum, we provide a systematic comparison of different compliance rule languages based on real-world compliance requirements which may inform future users and developers of these languages. Finally, we advocate for a more user-aware development of compliance languages which should consider a trade off between expressiveness, complexity and usability.

Keywords: conceptual modelling; compliance rules modelling; regulatory compliance; business processes; expressiveness; language complexity

Citation: Zasada, A.; Hashmi, M.; Fellmann, M.; Knuplesch, D Evaluation of Compliance Rule Languages for Modelling Regulatory Compliance Requirements. *Software* **2023**, 2, 71–120. https://doi.org/10.3390/software2010004

Academic Editors: Silvia Bonfanti and Juri Di Rocco

Received: 3 August 2022
Accepted: 25 November 2022
Published: 28 January 2023

1. Introduction

Companies are required to comply with an increasing number of regulations of different origins and complexity. Some rules are required by law, while others are rooted in best-practice frameworks, company policies and inter-company business contracts. One of the first legal initiatives to assure compliance was the American Sarbanes-Oxley Act (SOX) of 2002, a U.S. federal law which had a great impact on all financial reporting processes [1]. Not only did it increase the transparency of financial reporting, it also paved the way for auditors to identify ineffective control mechanisms [2]. Other regulations such as BASEL series of acts I-III or MiFID (Markets in Financial Institute Directive) BASEL or followed, to address different aspects of financial reporting, including the disclosure of potential risks [3]. Moreover, compliance requirements not only arise from legislation, but have also developed from codes of practice and standards such as ISO 9000 ff, SCOR, ISO/IEC 27018:2014, GDPR [4], guidelines and business partner contracts, and internal control objectives [5,6]. Ensuring that all these compliance requirements are met is a major challenge for many enterprises, since it requires a labour-intense and error-prone review of all business operations.

This is all the more important, as enterprises increasingly rely on business process models and execution environments to manage and automate their business processes [7]. The collected data about processes and their execution greatly increases opportunities for automation. Therefore, approaches to ensure compliance in business process modelling and execution are regarded as highly relevant in the literature [6,8–22]. However, the existence of a plethora of sophisticated languages and approaches, in turn, creates a new challenge: *how do we find a suitable compliance language to capture real-world compliance requirements?* In order to decide which language fits best, two criteria are of utmost importance: (i) the language needs to be *expressive* enough to intuitively capture the respective compliance requirements of a business process; and (ii) the language has to be usable by compliance experts viz. it should be as *complex* as necessary and, at the same time, as simple as possible. Notice that, in this article, ur focus is not on the usability of modelling language, but rather on the expressive power and complexity of the compliance rule modelling languages. The usability analysis of modelling languages is another concern and deserves to be addressed separately as is achieved by Becker et al. [23].

Some works address this challenge by comparing different compliance languages. For example, the authors of COMPAS-Project [3] report on available models, languages and architectures for assuring compliance. Whereas Becker et al. [23] analyzes existing approaches for compliance checking regarding their applicability to arbitrary modelling techniques and their ability to address a wide range of compliance rules. Fellmann and Zasada [24], on the other hand, provide a comprehensive overview and classification of compliance languages. More recently, Hashmi and Governatori [25] mapped deontic modalities using compliance patterns proposed in different modeling languages. However, to the best of our knowledge, no comprehensive study has been reported in the literature that formally investigates the complexity and expresiveness of existing compliance requirements modelling languages. To obviate this shortcoming, in this article, we address the following research questions: to what extent can compliance requirements be expressed by existing languages? Thus, the contribution of this paper is a detailed review of the expressiveness of compliance rules languages. In addition, we also evaluate the complexity of such languages.

The remainder of the paper is structured as follows. In the next section, we discuss the overall research procedure adopted for this paper (Section 2). After that, we elaborate on our language selection process (Section 3) and introduce the compliance requirements of a financial-services process (Section 4) that we use for the rule formalisation and evaluation of the languages' semantic expressiveness (Section 5). We then consolidate the results by conducting a comparative analysis, pointing out representable deontic effects on the one hand, and the textual complexity of the formalisation on the other (Section 6). Thereafter,

we discuss related work (Section 7) and the analysis' results (Section 8), before we conclude the paper with some final remarks and indications for future research (Section 9).

2. Research Approach

This paper provides a systematic procedure to compare and evaluate the expressiveness and complexity of compliance rule languages. Embedded in techniques for process modelling analysis, business process compliance (BPC) aims to design, analyse and monitor requirements that are imposed by laws, regulations and standards [26]. Existing approaches in this well-developed field of research allow for compliance checking from different perspectives, i.e., declarative or normative, using a textual and/or graphical representation to formally express compliance rules for subsequent model checks. Being able to elicit relevant process information, by abstracting from the natural language in which regulatory requirements are encoded, is one of the key challenges of BPC research.

Naturally, existing methods deviate in the exactness and granularity in which compliance requirements are modelled, due to the applied formalism. Focussing on technical implementation, it often remains unclear to what extent the formalism captures the meaning of a given compliance rule accurately. The chosen formalism also entails a certain degree of complexity, which has a great impact on the understandability of the respective language. A number of approaches, therefore, suggest a graphical layer, or the use of patterns, to hide the complexity of the underlying fomalism. However, neither of these approaches distinguishes between the introduced "high-level" language and the formalism itself, when discussing its expressiveness and complexity. In fact, most approaches focus on the modelling and verification of compliance rules without conducting usability studies [23].

In addition, using different running examples tailored to a specific problem often hampers the fair comparison between approaches and impairs their generalisability. Hence, we refine and extend the discussion of the applicability of compliance rule languages to:

1. The *expressiveness* in terms of the different scope (i.e., control flow, etc.) and deontic effects and modalities (i.e., obligations) of a compliance requirement as a normative construct.
2. The *complexity* measured by metrics that help untangle the components of formulas (i.e., operands, operators) as well as the elements of graphs (i.e., nodes, edges).

With the proposed methodology, we aim to determine the differences with which existing compliance rule languages can reflect the intuition of a given set of compliance rules and take measures to quantify the complexity of the applied formalism. We thereby provide an integrated approach to compare the expressiveness and complexity of compliance rule formalisations at design-time. The main contributions of this paper are:

- A systematic literature review and classification of business-process compliance languages.
- A practical application of typical compliance rule languages found in the literature.
- A comparison of the expressiveness and complexity for textual and visual compliance rule languages.

The overall approach consists of three phases: survey, design and evaluation (cf. Figure 1). The phases were adopted from Ly et al. [27], who derived a framework for Compliance Monitoring Functionalities (CMF) which facilitates a systematic comparison of approaches to compliance monitoring at run-time.

Phase 1—Survey: We start our survey with the definition of search phrases and criteria to set the goals and boundaries for the following literature review. We focus on approaches that are explicitly described as compliance approaches and refrain from technical-oriented approaches that view compliance as a peripheral aspect and solve rather specific problems. The approach had to be designed for end users of IT tools, which implies the availability of a meta-model or grammar which enables the automated execution of compliance checks (cf. Section 3).

The comparison of the approaches is driven by a realistic process from the finance industry which incorporates different types of compliance requirements. The requirements are divided into four process perspectives [28]: control flow, data flow, time and resources. In addition, we include the concept of a compensation to demonstrate how temporary compliance violations can be resolved later.

Phase 2—Modeling: After selecting and classifying the compliance approaches, we briefly introduce each language used in the respective approaches, before modelling the requirements derived from the given example process. As we distinguish between a high-level language (if applicable) and the underlying formalism, we provide a large number of details for the subsequent evaluation (involve [29] in Section 5.5). Moreover, we juxtapose textual and visual approaches to simplify the direct comparison between similar approaches (cf. Section 5).

Phase 3—Evaluation: In the last phase, the approaches are compared regarding their expressiveness and complexity. First, the expressiveness is measured by the level of completion with which the requirements introduced in *Phase 1* have been modelled in *Phase 2*. To extend the discussion to the correct interpretation and modelling of each compliance requirement, we characterize the semantics of the formalised rules with regard to a number of modalities (i.e., obligations, permissions and prohibitions) which are based on legal theory [25] (cf. Section 6.1).

Second, the complexity is measured by means of metrics which were originally used to analyse the complexity of software (modules). In this context, they serve as a simple but effective instrument to compare the lexical complexity of the different modelling languages.

In line with metrics for programming languages, both the variety and the volume of the used language constructs is considered when lexical complexity is determined. To do this, we first discuss the applicability of software metrics, before we apply the Halstead metrics to the rule formalisation developed in *Phase 2* (cf. Section 6.2).

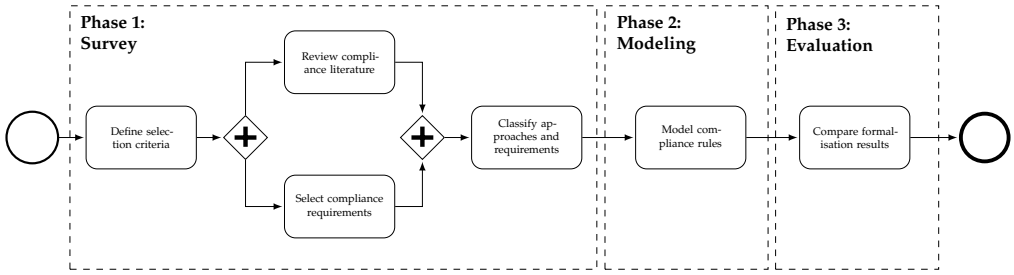


Figure 1. Phases of applied research procedure.

3. Language Selection Process

The literature review was conducted to identify relevant approaches to BPC. To structure our literature search and analysis, we followed the systematic review process proposed by Webster and Watson [30]. Since we were especially interested in compliance languages, we first selected suitable keywords and queried prominent scholarly databases such as Google Scholar, Science Direct, Scoupus, Web of Science to find scholarly references. To attain highly relevant articles, we continuously refined our searches in each iteration. To restrict our search, right from the beginning, to compliance publications in close relation to process management, we chose the "all in title" prefix. The final query was composed of the keywords *compliance (rule OR language OR pattern OR pattern-based) (business OR process OR workflow)*, which we also provided in their plural forms. We then obtained all 76 results of the literature search, and sorted out 2 duplicates, before we reviewed and discussed the remaining 74 hits based on title and abstract. The list of identified articles can be retrieved from https://www.dropbox.com/s/buyju4d6zjyfjdm/Library.pdf?dl=0,

accessed on 3 August 2022. Note that the list contains only 58 entries due to the temporal availability of some reference details.

The pre-selection revealed 67 relevant approaches, for which a forward/backward search was conducted. The main criteria for selecting a paper was that it addressed compliance checking and focused on instruments such as languages or patterns to express compliance requirements in order to allow for compliance checking. Thereby, we identified 177 additional references which were reviewed by criteria introduced in the next section.

In order to be able to conduct an in-depth comparison of languages and also to report the results in sufficient detail, we had to reduce the large number of possible approaches further. In addition, this is also reasonable since, despite the large number of publications, there exists only a limited number of distinct compliance approaches that make use of an even smaller set of compliance languages. With "approach", we denote the fundamental strategy of compliance rule representation and/or checking which, in turn, can rely on (one or multiple) compliance languages. We, hence, applied additional filtering criteria [31] for the inclusion of compliance approaches (cf. Figure 2): (a) We selected approaches that are explicitly described as a compliance language or pattern catalogue. With this, we required some versatility of the approach and sorted out approaches that deal with compliance to a certain degree, but primarily solve other (or very specific) problems; (b) We preferred "high-level" languages or patterns intended for the business analyst or compliance expert over lower level languages and patterns. With this, we selected approaches that address end users of IT tools rather than IT experts and eliminated approaches that primarily focus on implementation aspects; (c) We required that a meta-model or grammar is provided in one of the publications of the approach. With this, we ensure that the approach, though being versatile, can nevertheless be applied in a systematic way, i.e., that there are principles that govern the composition from primitive language or patterns to complex expressions; (d) We required the machine processability of expressions created with the compliance language or pattern system, at least in theory. With this, we ensured that the approach is amenable to IT support and, thus, eliminated management frameworks and approaches with little to no automation potential.

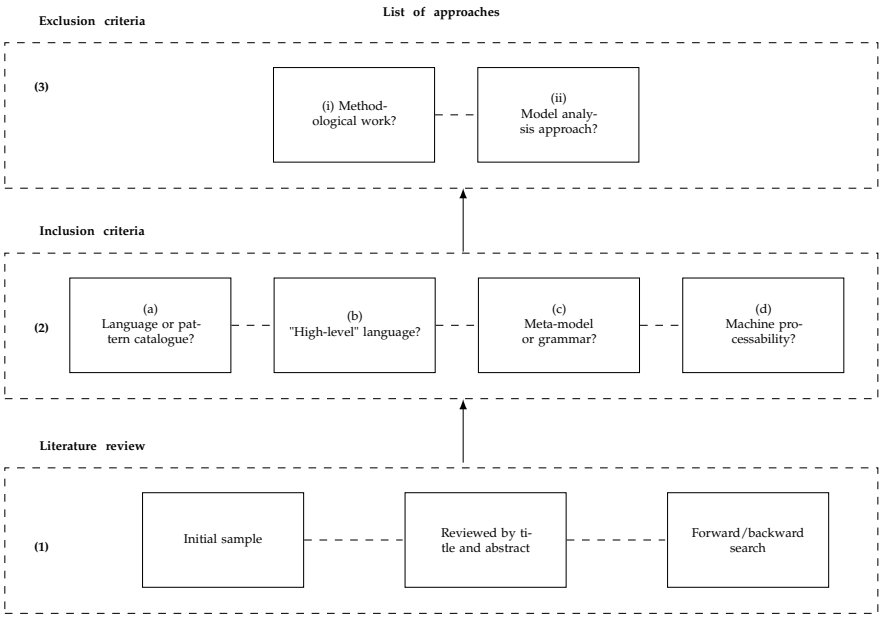


Figure 2. Filtering criteria.

Besides these inclusion criteria, we additionally applied exclusion criteria. We did not further consider an approach: (a) If it consisted of pure methodological work—e.g., procedures for how to deal with compliance problems; (b) If existing (formal) languages are applied in a transformation scenario where the initial model does not capture compliance

rules explicitly and compliance rules are predominantly hard coded into the compliance checking tool. Thus, we classify these languages as model analysis approaches.

The described procedure resulted in 44 approaches out of 208 reviewed papers. Although all 44 approaches were considered as relevant, they were still overlapping in content. Table 1 contains the results. In addition to the short description we give on the particular approach, we also decided to assign each language to one or more categories, namely: *graph*, *pattern*, *query* and *logic*. Graph-based approaches typically use visual elements to model compliance rules, while pattern, query and logic-based approaches frequently rely on textual formulas. Often, these concepts are combined to model a compliance rule as, for example, in [32], where queries are represented as visual patterns and mapped to a formal expression in *computational tree logic* (CTL).

By comparing a class of languages instead of single unlinked approaches, we aim to generalize the results and provide a substantial knowledge base to better understand the complexity and expressiveness of compliance rule languages. For the language comparison, we finally chose two languages of each category, as indicated in Table 1. The seven languages are formally introduced in Section 5, before being compared in Section 6 based on an example process and its compliance requirements, which are presented in the next section.

Table 1. List of selected approaches.

Language	Approach	Classification			
		Graph	Pattern	Query	Logic
BPMN-Q [32] *	Pattern-based language expressing compliance requirements as BPMN graphs with a formal representation in CTL.	X	X	X	
CRL [33] *	Pattern-based language expressing compliance requirements as atomic and composite pattern with a formal representation in LTL and MTL.		X		
Declare [28] *	Framework supporting several declarative languages (i.e., ConDec, ConDec++, DecSerFlow) for modelling graphical constraints with a formal representation in LTL.	X	X		
DCR [34]	Combination of two logics for verifying compliance via model checking. Process and organisational view are expressed as graphs before being translated into description logic. The data view is implemented as hybrid logic.	X	X		X
DMQL [35] *	Query language based on graph theory for matching patterns in conceptual models represented in arbitrary modelling languages.	X	X	X	
eCRG [36] *	Visual language for modelling compliance rules with a formal representation in FOL. Specified compliance rules are verified against event logs.	X			
PCL [37] *	Combination of deontic and defeasable logic to capture the intuition of normative requirements. Compliance rules are formally represented as obligations, prohibitions and permissions.				X
Petri-net Pattern [38]	Repository of frequent compliance patterns formally represented as Petri-nets. Conformance checks are performed on the basis of event logs.	X	X		
PENELOPE [39] *	Declarative language for expressing temporal deontic assignments. Compliance rules are formally represented as obligations and permissions.				X
PROPOLS [40]	Pattern-based property specification language based on OWL. Specified properties are used to verify compliance in BPEL service composition schemas.	X	X		
PPSL [41]	Pattern-based property specification language based on UML activities. Constraints are modelled as visual patterns with a formal representation in LTL.	X	X		
Rule Pattern [42]	Rule-based process mining approach enclosing patterns formally grounded and divided into static (FOL), dynamic (LTL) and composed pattern.	X			

Abbreviations: Business Process Model and Notation Query Language (BPMN-Q), Business Property Specification Language (BPSL), Compliance Request Language (CRL), Computational Tree Logic (CTL), Diagramed

Model Query Language (DMQL), Dynamic Condition Response Graph (DCR), extended Compliance Rule Graph (eCRG), Process Compliance Language (PCL), Linear Temporal Logic (LTL), Metrical Temporal Logic (MTL), Process Entailment from the Elicitation of Obligations and Permissions (PENELOPE), Process Pattern Specification Language (PPSL), Property Specification Pattern Ontology Language for Service Composition (PROPOLS), Unified Modeling Language (UML), Web Ontology Language (OWL). **Comment**: Compared approaches are marked with *.

4. Running Example

Since our main motivation is to compare compliance languages, we begin with establishing a common ground for comparison. We do this in this section by providing a running example from the financial industry. In Section 4.1, we present a securities purchasing process which we use to derive types of compliance requirements in Section 4.2.

4.1. Process Model

Compliance has a long tradition in the finance sector. Examples have been drawn, e.g., from sales and consultation [43], loan application [33,35] and account opening [7], and studied with respect to the different constraints that occur in a process [44]. In this paper, we introduce a financial advisory process. The compliance requirements of this process reflect financial regulations of the *German Securities Trading Act* (WpHG), the *EU Markets in Financial Instruments Directive* (MiFID II) as well as the *US Foreign Account Tax Compliance Act* (FACTA). Basically, these regulations establish principles and criteria to take appropriate measures against insider trading and money laundering [45] The classification of these requirements is shown in Table 2.

Table 2. Classification of compliance requirements based on process aspects.

Control flow	R1: Order R2: Occurence
Data flow	R3: Data value R4: Interaction
Resources	R5: Employee role R6: Segregation of duties
Time	R7: Point in time R8: Period of time (interval)

The process model shown in Figure 3 resulted from a process workshop and a number of interviews conducted with the compliance officer and three customer advisors of a German savings bank [46]. During the first iteration, we focused on the general structure of the process including the legal regulations that apply. The second iteration helped us to identify compliance practices and corresponding requirements. The original purpose of the qualitative interviews was to discover how IT helps employees to ensure compliance. Hence, the compliance requirements had not been the focus until that point.

The process exemplifies a securities purchase for new and existing customers. It was selected due to its relevance for day-to-day operations and its transferability to other asset classes. The process starts with a new customer request and branches off into activities that are either related to the approval of a new customer, or the maintenance of an existing customer. Based on the customer's knowledge base and investment profile, the advisor discloses the individual investment risks to the customer before the investment advice is given and sufficiently documented. The process is completed by the execution of the customer order and some administrative tasks related to the securities purchase. In the last step, the invoice is sent to the customer.

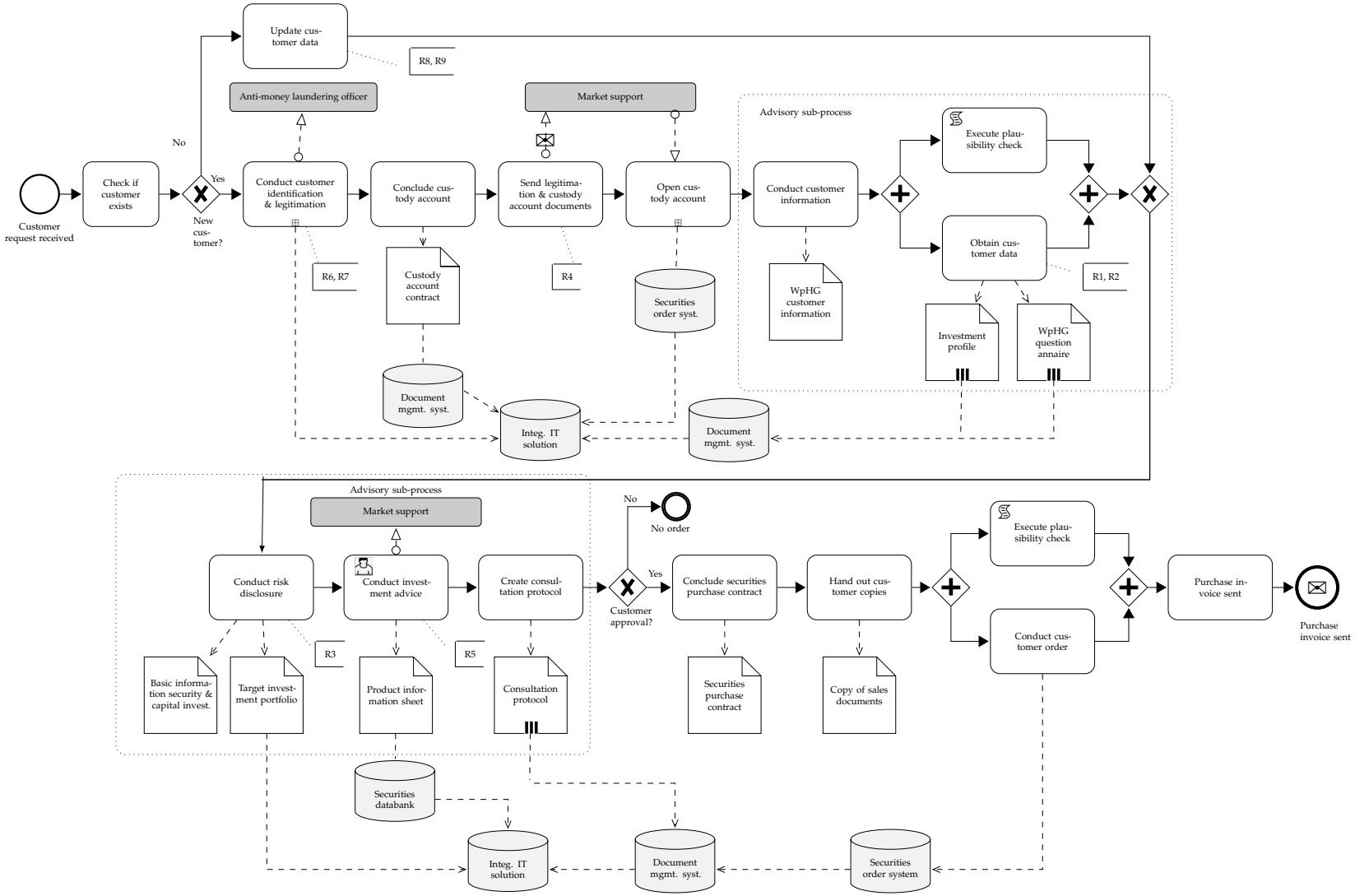


Figure 3. BPMN model of the securities purchase process.

4.2. Compliance Requirements

With the adoption of legal standards such as SOX [1] and BASEL III [47], financial institutes have become obliged to establish and maintain an internal control system that facilitates an effective risk and compliance management.

Over the course of this process, compliance requirements are compiled from legal releases and distributed through internal channels which are run by the compliance department. The department is headed by legal experts, who deal with the complexity of externally imposed regulations and manage the implementation of compliance requirements. Their work is supported by information systems which are specialised, for example, for the retrieval of information (securities databank) or business operations (order system) which have a relation to compliance regulations (see Figure 3). The electronic document-management system is the foundation for distributing compliance information, with links to concrete working instructions, questionnaires, protocols and contracts. Working instructions included the legal references that served us as template to formulate nine compliance requirements (see Table 3).

The first requirement *R1* stipulates which information must be obtained from the customer before a subsequent activity can be performed. *R2* simply states which information needs to be shared with the customer. The two requirements are related to *R3*, which requires the customer to acknowledge the receipt of this information. *R4* specifies which documents need to be forwarded to another business unit after the customer has been identified and legitimised. *R5* determines the competence level (role) of the consultant during the advisory process, while *R6* stipulates which two activities must be performed by employees with different roles (four-eyes principle). *R7* and *R8* indicate whether a task is related to a specific point in time or a time interval.

Table 3. Compliance requirements of the securities purchase process.

ID	Compliance Requirement
R1	The customer data must be received before the individual risk assessment can take place.
R2	The customer advisor must provide the two obligatory brochures *WpHG Customer Information* and the *Basic Information Securities and Capital Investment*.
R3	The customer advisor must ensure that the customer acknowledges receipt of the two brochures
R4	After concluding the custody account contract, the customer legitimation and the account documents need to be send to market support.
R5	The investment advice needs to be conducted by a customer advisor with a securities competence of level C or above.
R6	The customer identification and legitimation must be handled by the customer advisor, while suspected cases of money laundering must be checked by an anti-money-laundering officer.
R7	Before concluding a custody-account contract, the customer advisor needs to wait until the suspected case of anti-money laundering is resolved.
R8	The customer information must be updated with every future customer contact.
R9	Stockless custody accounts are charged with a fee of EUR 5 per year. If the fee is not paid, the account is terminated by market support. The account is reactivated by a new securities purchase.

The classification of these requirements is shown in Table 2. It is comprised of the four process perspectives: control flow, data flow, resources and time, which have been addressed in many compliance approaches. In addition, in *R9*, we included the concept of a compensation for managing exceptions from a rule [27]. A compensation usually consists of an *if-then-else* statement [33] which defines what happens when the first condition cannot be met, and a second repairing action has to be implemented in order to comply. Based

on these five requirement types, we develop a formal representation of the rules in the next section.

5. Language-Specific Compliance Rule Formalisation

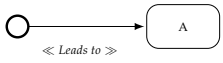
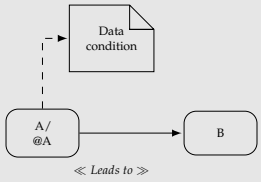
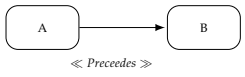
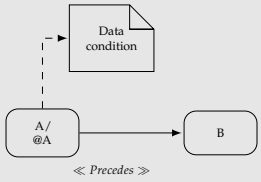
In this section, we present the formalisation of the selected languages. For this, we first introduce the basic constructs of the language. We then demonstrate how well these languages can express the previously categorised compliance requirements. Finally, we specify the formal semantics of the expressions by providing a mapping to a formal language.

5.1. BPMN-Q

BPMN-Q (*Business Process Model and Notation Query Language* [32]) is a visual query language for specifying control-flow and data-flow rules as a pattern which can be used to query a repository of BPMN process models. Each pattern has a formal representation in CTL to check for compliance violations, and a set of anti-patterns [48] to query the part of the process model causing the violation. The approach builds on existing pattern classifications such as [49] describing the presence, absence and/or ordering of activities [9]. BPMN-Q comprises two types of patterns, namely, control-flow and conditional control-flow patterns. The latter are used to express data conditions on activities. For the rule formalisation in BPMN-Q, we adopted four (conditional) control-flow patterns (see Table 4). Anti-patterns, which are generated from the original patterns to describe potential violations, have not been considered (they can be used to, e.g., detect violations at the runtime of the process, i.e., queries for violations).

A BPMN-Q pattern is basically a rudimentary BPMN graph composed of activities and events and a path edge connecting the flow objects [32]. If a data condition is consigned in the requirement, the pattern is annotated by an implicit data input–output.

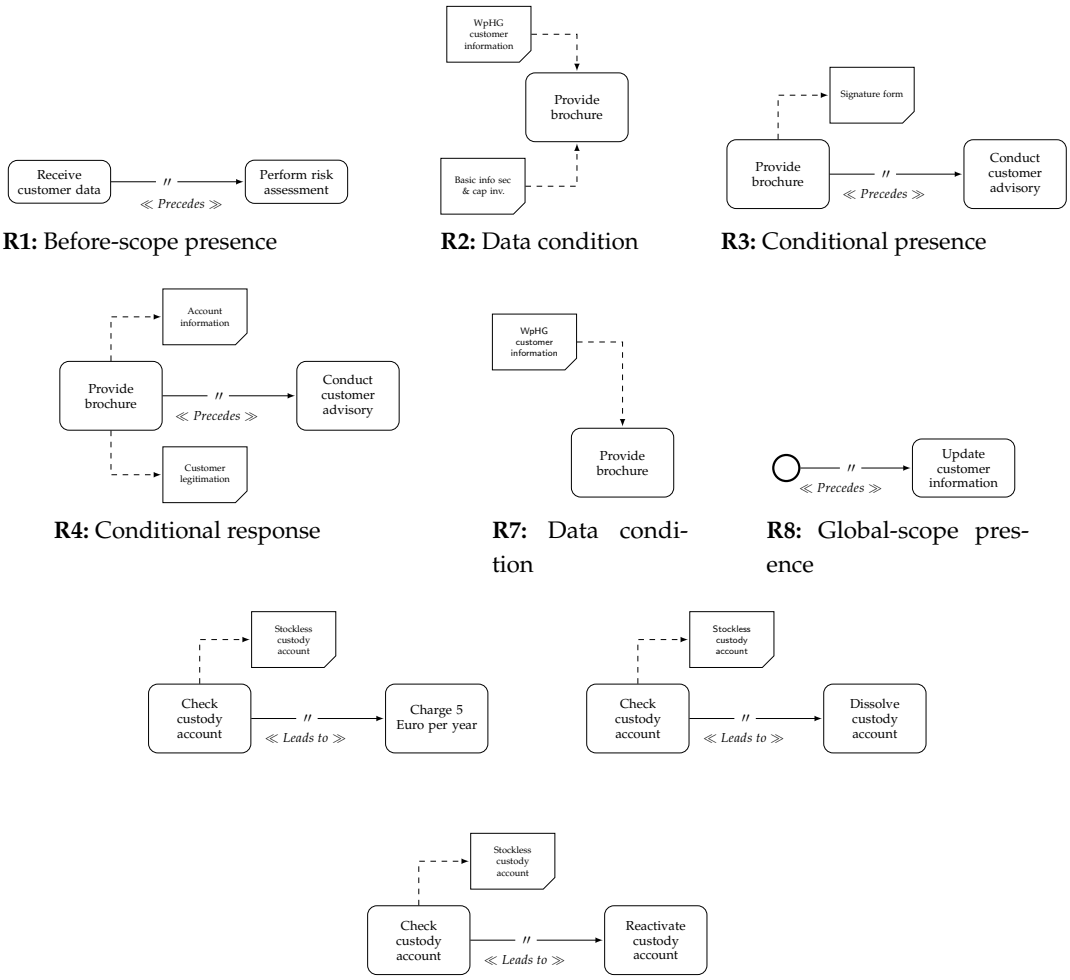
Table 4. Excerpt of BPMN-Q patterns (adopted from [32]).

Pattern	Type	CTL Mapping
	Global-scope Presence (Leads To)	AG(start → AF(executed(A)))
	Conditional Response (Leads to)	AG((executed(A) ∧ stableDataCondition)→ AF(executed(B)))
	Before-scope Presence (Precedence)	¬E[¬executed(A) U ready(B)]
	Conditional Presence (Precedence)	¬E[¬(executed(A) ∧ stableDataCondition) U ready(B)]b

Naming convention used in the mappings: logical operators (¬ not, ∧ and, → implies); activities *A* and *B*, anonymous activity @*A*; predicates start (process), end (process), ready (activity), executed (activity); temporal operators: A (every), E (some), F (final), G (global) and U (until).

The CTL semantics for the visual patterns (cf. Figure 4) are listed in Table 5. *R1* has been modelled using the *Before-scope Presence* (*Precedence*) pattern (i.e., ¬*E*[¬*executed*(*A*) *U ready*(*B*)]) indicating that the individual risks cannot be assessed (*B*) before the customer data has been received (*A*). In contrast to *R1*, the update of the customer information (*A*) required in *R8* applies to the whole process (i.e., *AG*(*start* → *AF*(*executed*(*A*)))) and can,

thus, be modelled with the *Global-scope Presence* (*Leads to*) pattern. Note that the graphical notation of the *Precedence* and *Leads to* pattern does not change the diagram except for the name of the pattern that is assigned to the path edge [9].



R1: Before-scope presence **R2:** Data condition **R3:** Conditional presence

R4: Conditional response **R7:** Data condition **R8:** Global-scope presence

R9: Conditional response

Figure 4. Formalised requirements in BPMN-Q patterns.

R3 and *R4* visualize the dependency between an activity and a data condition. For both patterns, the data condition is aligned with the first activity denoted by the name of the data object and its state at the time the activity is executed. For the formal definition of the *stableDataCondition*, see [32]. *R2* and *R7* depict two more data conditions; however, due to the absence of a second activity, it is modelled only with a single activity. Alternatively, these rules can be realized with an anonymous activity (@*A*), but the corresponding textual requirements do not specify the relation to a second activity. Through the lack of an explicit compensation pattern, *R9* is modelled by repeating the initial activity and data condition.

As the formalisation shows, BPMN-Q patterns are easy to read and interpret, although the complexity of the corresponding CTL formula increases with respect to conditional patterns and to compliance rules which are composed of more than one pattern. To apply BPMN-Q, however, all process models have to be specified in BPMN, and, although there is the possibility to extend the language, other process perspectives such as resources (*R5* and *R6*), time and data are currently not supported.

Table 5. Mapping of formalised requirements from BPMN-Q to CTL.

ID	CTL Representation
R1	¬E[¬executed(*Receive customer data*) U ready(*Perform risk assessment*)]
R2	AG(ready(*Check new customer*) → state(*Customer information brochure, provided*)) AG(ready(*Check new customer*) → state(*Basic information securities and capital investment brochure, provided*))
R3	¬E[¬(executed(*Conduct customer advisory*) ∧ state(*Signature form, signed*) U ready(*Provide brochures*)]
R4	AG((executed(*Conclude custody account contract*) ∧ state(*Documents, received*)) → AF(executed(*Send account information and legitimation documents*)))
R5	*R5 cannot be modelled with BPMN-Q semantics.*
R6	*R6 cannot be modelled with BPMN-Q semantics.*
R7	AG(ready(*Conclude custody account*) → state(*Money-laundering case, solved*))
R8	AG(start → AF(executed(*Update customer information*)))
R9	*R9 cannot be modelled with BPMN-Q semantics.*

5.2. CRL

CRL (*Compliance Request Language*, [33]) is a systemised collection of typical complaince patterns. CRL provides four classes of rule patterns as well as a mapping to temporal logic to verify compliance rules annotated in a process model. Atomic patterns (i.e., order and occurrence, [50]) are aligned with the property specification patterns developed by Dwyer [49]. This pattern collection has been extended for compliance checking introducing resource patterns, timed patterns and a notion for composite patterns [51]. A special feature of CRL is that the approach combines two logics for the formal presentation of patterns. Beside *Linear Temporal Logic* (LTL), *Metric Temporal Logic* (MTL) is applied to compensate for the missing support of timed patterns in LTL [52]. An excerpt of the patterns and their formal specification in LTL is given in Table 6.

A compliance rule is modelled by choosing the correct pattern from the classes of the CRL meta-model [33]. The first two classes reflect the control flow (i.e., atomic patterns) and resource perspective of a process. The patterns of the other two classes can be used to depict time conditions (i.e., timed patterns) and to model dependencies between operands (i.e., composite patterns). Operands represent business objects as well as activities and events, and their attributes or conditions [53]. As a general rule, an operand begins with an activity (or event), followed by one or more business objects describing the particular compliance task. The formulation of pattern-based expressions is comparable to the use of activities and events of Event-driven Process Chains. Atomic patterns can be combined into more complex and even nested expressions using Boolean operators (i.e., AND, OR) and parentheses. An expression is built from patterns and operands, which have a direct mapping to either LTL or MTL.

Table 7 shows how the patterns introduced in Table 6 are applied to model compliance rules. Note that the example rules do not require an original timed pattern, which is why we rely exclusively on LTL semantics [54]. *R1*, *R4* and *R8* are modelled with the atomic patterns *Precedes* and *LeadsTo*, respectively. Both patterns imply that the rule has to hold for every occurrence of the activity. In the case of *R1*, one task has to be performed before another task ($\neg Q$ W P); that is, the risk analysis cannot take place ($\neg Q$) until the customer data have been received (P).

Table 6. Excerpt of CRL patterns (adopted from [33]).

Pattern	Type	LTL Mapping
P Exists	Atomic Pattern	$\mathsf{F}(P)$
P LeadsTo Q	Atomic Pattern	$\mathsf{G}(P \rightarrow \mathsf{F}(Q))$
P Precedes Q	Atomic Pattern	$\neg Q \ \mathsf{W} \ P$
t PerformedBy R	Resource Pattern	$\mathsf{G}(t \rightarrow t.Role(R))$
t_1 SegregatedFrom t_2	Resource Pattern	$\mathsf{G}(t_1.Role(R) \rightarrow \neg(t_2.Role(R)) \wedge \mathsf{G}(t_1.User(U) \rightarrow \neg(t_2.User(U))$
P Frees Q	Atomic Pattern	$P \ \mathsf{R} \ Q$
P (LeadsTo | DirectlyFollowedBy) P_1 (Else | ElseNext) $P_2, \ldots,$ (Else | ElseNext) P_n	Atomic Pattern	$\mathsf{G}(p \rightarrow \mathsf{F} \mid \mathsf{X}(p_{1 \wedge 1 \leq i < n-1}(\mathsf{F} \mid \mathsf{X}(p_i \ NotSucceed) \wedge (p_i) \ NotSucceed \rightarrow \mathsf{F} \mid \mathsf{X}(p_i + 1)))))$

Naming convention used in the mappings: logical operators ($\neg$ not, $\wedge$ and, $\rightarrow$ implies); operands P, Q, R, T, p_i and t_i with $i, n \in \aleph$; temporal operators F (final), G (global), R (release), W(weak until) and X (next).

On the contrary, *R4* and *R8* are modelled assuming that an activity, such as concluding a custody-account contract (P), must be followed by sending the account documents and the customer legitimation to market support (Q). Here, the focus shifts from activity Q to P as the triggering element ($\mathsf{G}(P \rightarrow \mathsf{F}(Q))$. However, the exact time period, in which these rules have to be fulfilled, cannot be specified until more information about the process is given. For the same reason, interactions in terms of message exchanges between business partners are not modelled explicitly [55].

Table 7. Mapping of formalised requirements using CRL patterns to LTL.

ID	Pattern and LTL Representation
R1	*ReceiveCustomerData* Precedes *PerformRiskAssessment* $\neg$*ReceiveCustomerData* W(*PerformRiskAssessment*)
R2	(*CustomerInformationBrochure* And *SecuritiesAndCapitalInvestmentBrochure*) Exists F(*CustomerInformationBrochure* $\wedge$ *SecuritiesAndCapitalInvestmentBrochure*)
R3	*NewCustomer.AcknowledgeInformationBrochures* = *'Yes'* Frees*ConductCustomerAdvisory* *NewCustomer.AcknowledgeInformationBrochures* = *'Yes'* R(*ConductCustomerAdvisory*)
R4	*ConcludeCustodyAccountContract* LeadsTo (*SendAccountDocuments* And *SendCustomerLegitimation*) G(*ConcludeCustodyAccountContract* $\rightarrow$ F(*SendAccountDocuments* $\wedge$ *SendCustomerLegitimation*))
R5	*ConductInvestmentAdvice* PerformedBy *Role.CustomerAdvisorSecuritiesCompetence* $\geq$ 'C' G(*ConductInvestmentAdvice* $\rightarrow$ *ConductInvestmentAdvice.Role*(*'CustomerAdvisor'* $\wedge$ *SecuritiesCompetence* $\geq$ 'C'))
R6	(*ConductCustomerIdentification* And *ConductCustomerLegitimation*) SegregatedFrom *CheckSuspectedMoneyLaunderingCase* G((*ConductCustomerIndentification.Role*(*'CustomerAdvisor'*) $\wedge$ *ConductCustomerLegitimation.Role*(*'CustomerAdvisor'*)) $\rightarrow$ $\neg$(*CheckSuspectedMoneyLaunderingCase.Role*(*'AntiMoneyLaunderingOfficer'*)) $\wedge$ G((*ConductCustomerIndentification.User*('U') $\wedge$ Conduct-CustomerLegitimation.User('U')) $\rightarrow$ $\neg$(*CheckSuspectedMoneyLaunderingCase.User*('U'))
R7	*MoneyLaunderingCase.Solved* = *'Yes'* Frees *ConcludeCustodyAccount* *MoneyLaunderingCase.Solved* = *'Yes'* R(*ConcludeCustodyAccount*)
R8	*CustomerContact* LeadsTo *UpdateCustomerInformation* G(*CustomerContact* $\rightarrow$ F(*UpdateCustomerInformation*))
R9	((*CheckCustodyAccount* = *'Stockless'*) LeadsTo (*CustomerPayFee* = 5 *'Euro'*)) Else *DissolveCustodyAccount* Else *ReactivateDissolvedCustodyAccount* G((*CheckCustodyAccount* = *'Stockless'*) $\rightarrow$ (F(*CustomerPayFee* = 5 *'Euro'*) $\wedge$ F(*CustomerPayFeeNotSuccceed*) $\wedge$ (*CustomerPayFeeNot-Succceed*) $\rightarrow$ X(*DissolveCustodyAccountNotSucceed*) $\wedge$ F(*DissolveCustodyAccountNotSucceed*) $\wedge$ (*DissolveCustodyAccountNotSucceed*) F(*ReactivateDissolvedCustodyAccount*))))

The atomic pattern *P Exists* indicates that the activity P has to occur at least once during the process. In *R2*, this pattern is used to ensure that legal information in the form of two brochures is passed to the customer (*F(P)*). To model the requirements *R3* and *R7*, we used the *Frees* pattern. It includes a data condition (cf. *R3*) that checks, for a new customer (P), whether s/he has acknowledged the receipt of the obligatory brochures (Q). That means that the second operand Q must be true up to and including the point where

the operand P first becomes true ($P\ R\ Q$) [33]. In this way, most control-flow rules can be expressed using only atomic patterns, which coincide with the control-flow patterns found in the literature [44].

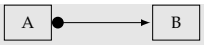
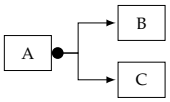
R5 and *R6* are examples of two resource patterns. They can be used to assign a task directly to a role, which can then be executed by several users (cf. *R5*). For the segregation of duties (cf. *R6*), role and user are separated from each other. An alternative concept of modelling the relation between activities whose execution might conflict is a compensation. A compensation (cf. *R9*) is realised in CRL as an *if-then-else* statement [33]. Initiated by a *LeadsTo* (or *DirectlyFollowedBy*) pattern, the *Else* (or *ElseNext*) pattern signifies that a primary action (P) can be compensated by defining at least one repairing action ($P_1 \ldots P_n$).

While the presented patterns require, in general, only little interpretation, the complexity of the underlying LTL formulas clearly increases with the complexity of the pattern expression, as observed for resource patterns, and compensations and composite pattern (cf. Table 7).

5.3. Declare

Declare [28,56] is a constraint-based language for the specification, verification, and monitoring of processes and process orchestrations. Declare is based on a set of constraint templates (i.e., visual control-flow patterns) which allow the specifying of the order in which tasks should be executed. Constraints are visualised as connectors or annotations to activities which, in turn, are drawn as boxes. A formal translation to LTL specifies the semantics of Declare constraints. Table 8 shows examples of Declare templates and their mapping to LTL.

Table 8. Excerpt of Declare constraints (adopted from [57]).

Visual Constraint	Control Flow Pattern	LTL Mapping
	Existence(A)	$\mathbf{F}(A)$
	Responded_Existence(A,B)	$\mathbf{F}(A) \rightarrow \mathbf{F}(B)$
	Co_Existence(A,B)	$\mathbf{F}(A) \leftrightarrow \mathbf{F}(B)$
	Response(A,B)	$\mathbf{F}(A) \leftrightarrow \mathbf{F}(B)$
	Branched_Response(A,(B,C))	$\mathbf{G}(A \rightarrow \mathbf{F}(B \vee C))$
	Precedence(A,B)	$(\mathbf{F}\ B) \rightarrow (\neg B\ U\ A)$
	Exclusive_1_Of_2(A,B)	$((\mathbf{F}\ A) \wedge (\neg \mathbf{F}\ B)) \vee ((\neg \mathbf{F}\ A) \wedge (\mathbf{F}\ B))$

Naming convention used in the mappings: logical operators ($\neg$ not, $\wedge$ and, $\vee$ or, $\rightarrow$ implies, $\leftrightarrow$ coincides); operands A and B; temporal operators F(final) and G (global)..

Declare templates can be partitioned into *existence templates*, *relation templates*, *choice templates*, and *negation templates*. *Existence templates* (e.g., Existence(A)) restrict the number of occurrences for a given activity A. Dependencies and sequence orders between activities are expressed by *relation templates*. For example, the constraint Responded_Existence(A,B) expresses that activity A must not occur without activity B. However, activity B may occur without activity A. The small filled circle attached to activity A visualises this difference, i.e., that only activity A triggers the constraint. In turn, the constraint Co_Existence(A,B) states that neither activity A nor activity B can occur without the other activity. The constraints Response(A,B) and Precedence(A,B) restrict the possible execution orders of activities A and B. In particular, Response(A,B) expresses that after each occurrence of activity A, activity B must occur as well (but B may occur without or before A). In turn, Precedence(A,B) requires that activity B may only occur after activity A.

The constraint Exclusive_1_Of_2(*A*,*B*) is a *choice constraint* which expresses that only one of the connected activities must be executed, i.e., either *A* or *B*. Note that Table 8 depicts only a small subset of the Declare constraints. Other types of constraints enable, for example, the specification of direct successors (e.g., Chain_Response(*A*,*B*) meaning that *A* must be followed by *B*). *Negation constraints* can force the absence of an activity in a certain scope (e.g., Not_Response(*A*, *B*)—*B* must not follow *A*). Finally, there are *branched* constraints indicating a disjunction of the branched activities.

Figure 5 depicts the rules relevant to the process model as Declare constraints and their mapping onto LTL. Declare focuses on the process control flow and, hence, is not able to cover details of compliance rules that refer to data and resources. Although the original purpose of Declare is the specification and verification of process models, Declare is also applicable to the specification and verification of compliance rules (e.g., [58]). In particular, Declare is able to express compliance rules *R1* and *R8* by using a Precedence and a Responded_Existence constraint, respectively. An Existence constraint can express that the activity *provide brochures* must occur at least once (cf. *R2*).

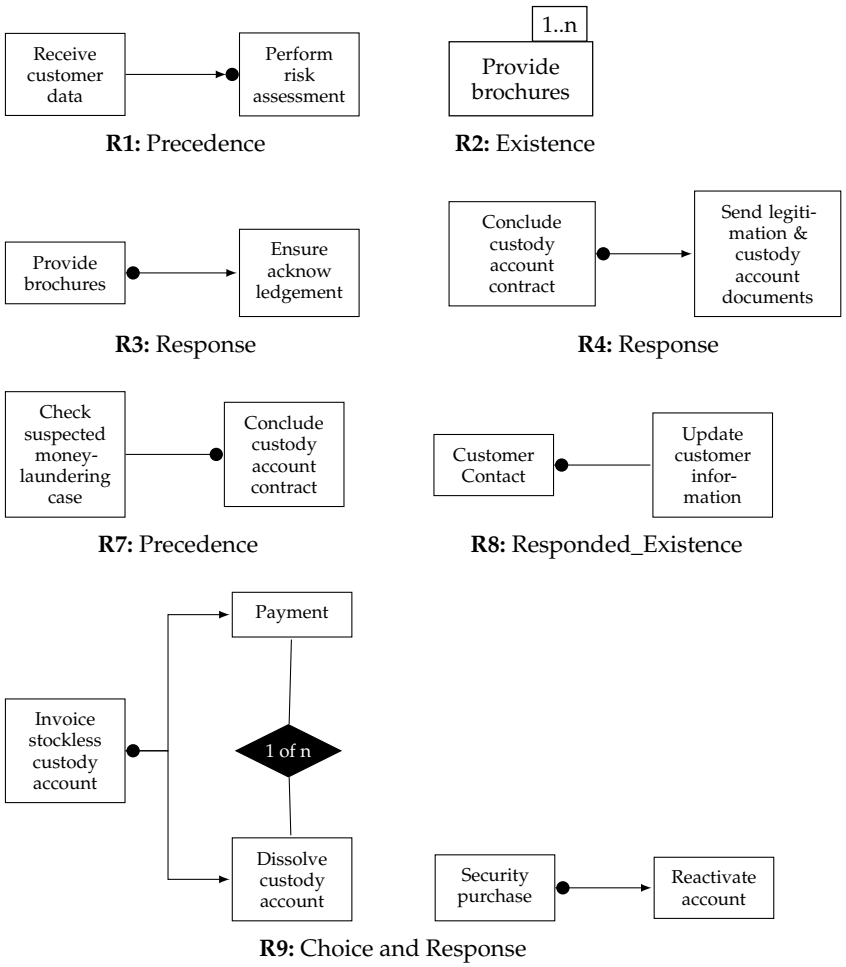


Figure 5. Formalised requirements in Declare patterns.

The Response constraint in *R3* and *R4* specifies that the activities *provide brochures* and *conclude custody-account contract* require a corresponding successor. In turn, the Precedence constraint depicted in *R7* ensures that the activity *check suspected money-laundering case* must be executed before activity *conclude custody-account contract*. However, Declare only partially covers the meaning of compliance rules *R2*–*R4* and *R7*. In particular, information about resources and data involved cannot be modelled (e.g., *brochure WpHG customer information* in *R2* and *R3* or the resource *customer adviser* in *R2*, *R3* and *R7*). In the context of *R9*, temporal constraints (e.g., *per year*) could not be considered. However, combining

a Branching_Response and an Exclusive constraint, Declare is capable of expressing that either activity *dissolve custody account* or activity *payment* must occur after activity *invoice stockless custody account*. Finally, *R5* and *R6* constitute pure resource constraints which cannot be represented by Declare. Table 9 shows Mapping of formalised requirements using Declare constraints to LTL

Table 9. Mapping of formalised requirements using Declare constraints to LTL.

ID	Pattern and LTL Representation
R1	Precedence(Receive customer data,Perform risk assessment)
	(**F** *Receivecustomerdata*)→ (¬*Receivecustomerdata U Performriskassessment*)
R2	Existence(*ProvideBrochures*)
	F(*ProvideBrochures*)
R3	Response(*ProvideBrochures*,Ensure Acknowledgement)
	F(*ProvideBrochures*)↔**F**(*EnsureAcknowledgement*)
R4	Response(*ConcludeContract*,Send Documents)
	F(*ConcludeContract*)↔**F**(*SendDocuments*)
R5	*R5 cannot be modelled with Declare semantics.*
R6	*R6 cannot be modelled with Declare semantics.*
R7	Precedence(Check Suspected Case,Conclude Contract)
	(**F** *CheckSuspectedCase*)→ (¬*CheckSuspectedCase U ConcludeContract*)
R8	Responded_Existence(*CustomerContact*,Update Customer Information)
	F(*CustomerContact*)→**F**(*UpdateCustomerInformation*)
R9	Branched_Response(*InvoiceStocklessCustodyAccount*,(*Payment*,*DissolveCustodyAccount*))
	G(*InvoiceStocklessCustodyAccount*→**F**(*Payment*∨*DissolveCustodyAccount*))
	Exclusive_1_Of_2(*Payment*,*DissolveCustodyAccount*)
	Exclusive_1_Of_2(*Payment*,*DissolveCustodyAccount*)
	Response(*SecurityPurchase*,Reactivate Account)
	F(*SecurityPurchase*)↔**F**(*ReactivateAccount*)

5.4. eCRG

The *extended Compliance Rule Graph* (eCRG) language is a graph-based notation for compliance rules [26,55]. It builds on the Compliance Rule Graph (CRG) language [59,60] and provides support for various process perspectives. eCRG follows an *if-then-else* semantics and, accordingly, is composed of an *antecedence pattern* and one or multiple *consequence pattern*:

- The antecedence pattern corresponds to the *if* part and describes the scope of a compliance rule, i.e., to which situations the rule is applied or when it is triggered.
- The consequence pattern corresponds to the *then* and *else* parts and describes how the rule can be satisfied once it is triggered.

In other words, each match of the antecedence pattern requires a corresponding match of at least one of the related consequence patterns. Both patterns can be specified using different kinds of nodes, connectors, and attachments (cf. Figure 6). In general, nodes correspond to the occurrence or absence of activities or messages. Resource, data, and point-in-time nodes are used as place holders for resources, data objects, and points in time or as references to existing or well-defined data objects, resources, or dates. Connectors allow the constraining of the sequence flow, data flow, or the allocation of resources. Other connectors support expressing relations between data objects or between resources. Most eCRG elements can be further refined through the attachment of conditions (e.g., temporal conditions or data conditions).

To allow distinguishing between elements of the antecedence and consequence patterns, nodes and connectors of the former pattern are drawn as solid, cornered boxes and solid lines, respectively, whereas nodes and connectors of the latter pattern are drawn as dashed, rounded boxes and dashed lines respectively. To express the absence of a certain entity, respective nodes are crossed out. Multiple consequence patterns are distinguished by annotating their elements with numbers that identify the respective pattern. The nodes referring to particular resources or data objects are drawn as thick, solid, cornered boxes.

The formal semantics of the eCRG language is defined by a transformation eCRG on *First Order Logic* (FOL). Further, the eCRG monitoring framework [26] provides the execution semantics which enables compliance monitoring with eCRG and supports all CMFs from [27]. The eCRG execution semantics annotates the nodes, connectors and attachments of an eCRG with colours, texts and symbols. To deal with multiple, potentially concurrent activations of a single compliance rule, not only one but multiples copies of an eCRG might be used in parallel.

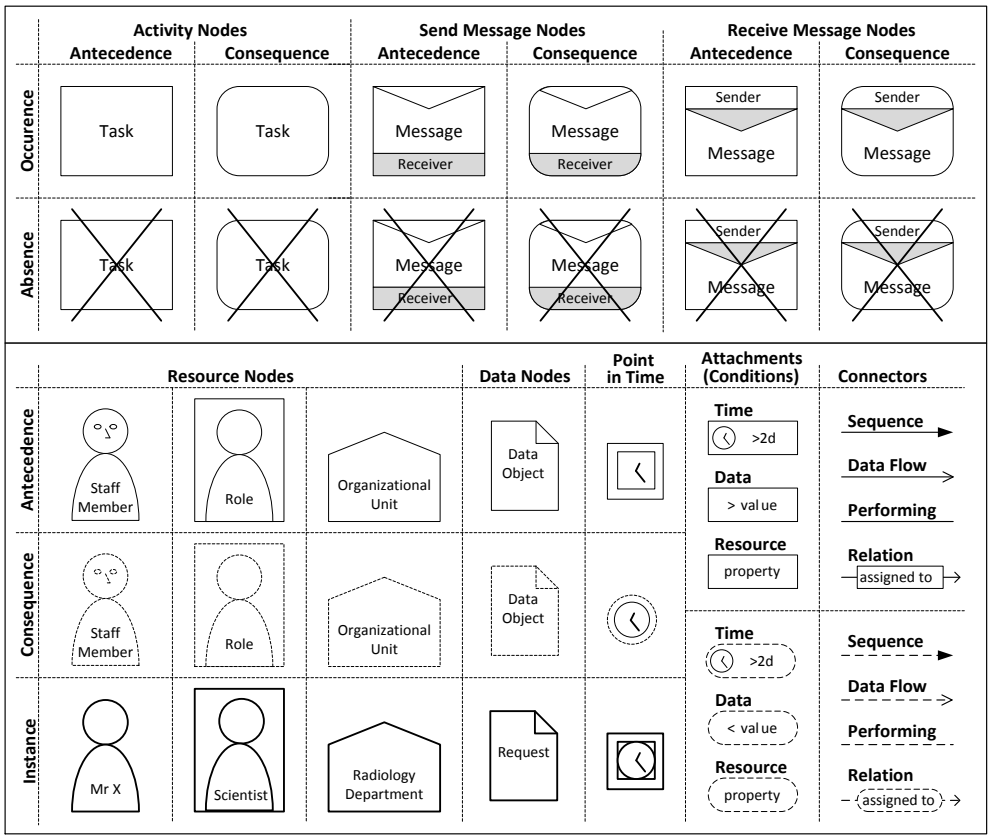


Figure 6. Excerpt of the eCRG language.

Figure 7 depicts the rules relevant to the process model as eCRGs. Table 10 provides the FOL expressions that result from these eCRGs. The eCRG language incorporates a large set of different elements. On the one hand, this large set of different elements can overstrain users in the beginning; on the other hand, it also enables nearly straightforward modelling of the example rules. Thereby, *R1–R5*, *R7* and *R8* can be modelled as a single eCRG expression, whereas *R6* and *R9* require two eCRGs. In particular, *R1* uses an antecedence occurrence node to indicate that activity *perform risk assessment* triggers *R1* and requires the previous receipt of *customer data*, as expressed by the corresponding consequence occurrence message node and the connecting consequence sequence-flow connector. This means that whenever the (antecedence occurrence) activity *perform risk assessment* occurs, the consequence has to be fulfilled as well, i.e., message *customer data* has to be received beforehand. The need for activity *provide brochure* in *R2* is modelled by a

consequence occurrence node. Note that *R2* does not contain any antecedence elements, as *R2* is active in any case. The activity should use the brochures that are specified by two data object instance nodes plus consequence data-flow connectors. The role of the performer is specified by the resource instance node that refers to role *customer advisor* and is connected by a consequence performing connector.

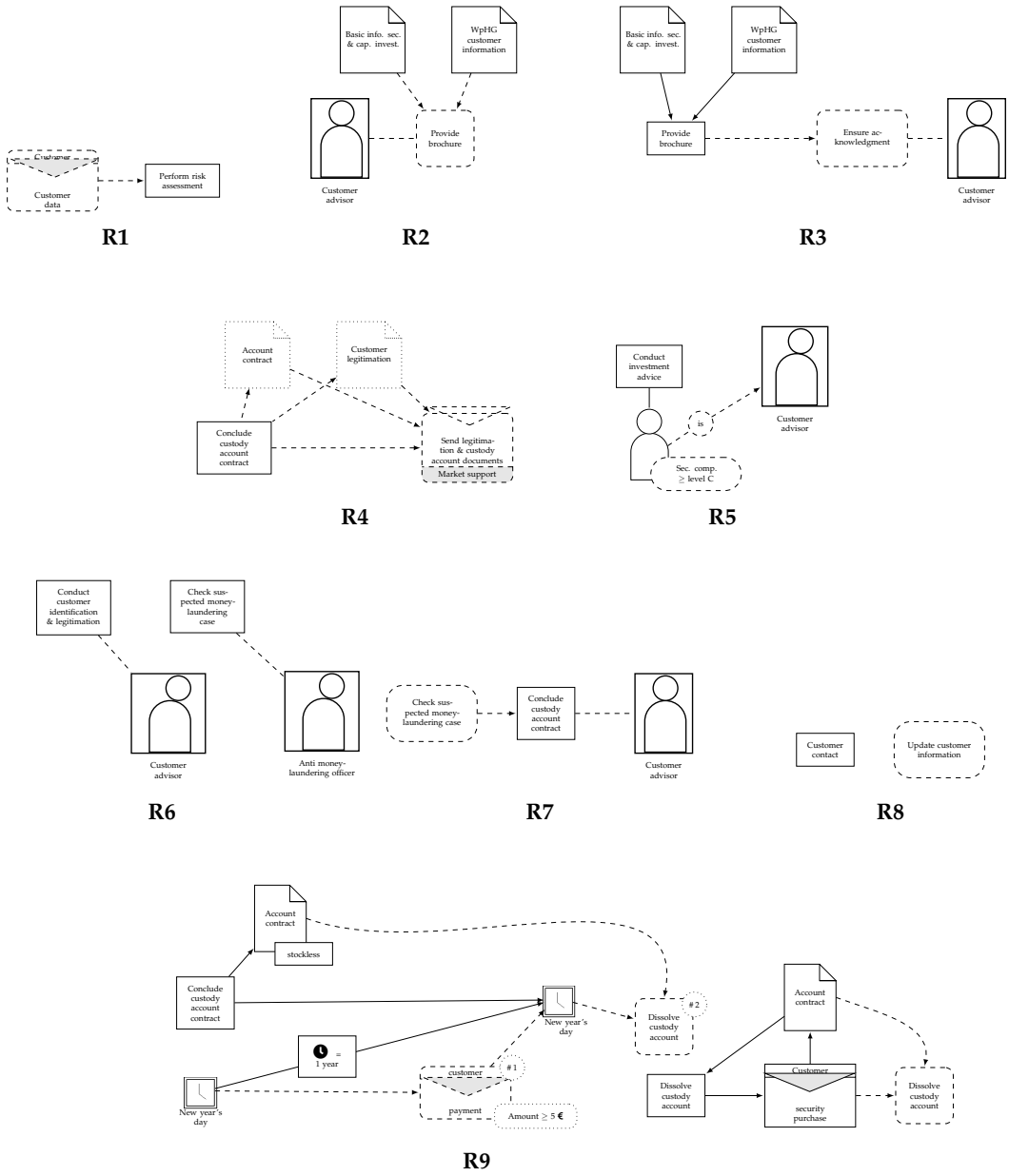


Figure 7. Formalised requirements in eCRG.

Table 10. FOL expressions of eCRG patterns.

R1	$\forall v_1^i, v_{s1}^t, v_{e1}^t : \Big(\big(Start(v_{s1}^t, v_1^i, perform\ risk\ assessment) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \big) \Rightarrow \exists v_2^i, v_2^t : \big(Receive(v_2^t, v_2^i, customer\ data, customer) \wedge (v_2^t < v_{s1}^t) \big) \Big)$
R2	$\exists v_1^i, v_{s1}^t, v_{e1}^t, v_1^{dv}, v_2^{dv}, v_1^r : \big(Start(v_{s1}^t, v_1^i, provide\ brochure) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \wedge Read(\cdot, v_1^i, \cdot, v_1^{dv}, \cdot) \wedge (v_1^{dv} = basic\ info.\ sec.\ \&\ cap.\ invest.) \wedge Read(\cdot, v_1^i, \cdot, v_2^{dv}, \cdot) \wedge (v_2^{dv} = WpHG\ customer\ information) \wedge Perform(v_1^i, v_1^r) \wedge InStaff(customer\ advisor, v_1^r) \big)$
R3	$\forall v_1^i, v_{s1}^t, v_{e1}^t, v_1^{dv}, v_2^{dv} : \Big(\big(Start(v_{s1}^t, v_1^i, provide\ brochure) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \wedge Read(\cdot, v_1^i, \cdot, v_1^{dv}, \cdot) \wedge (v_1^{dv} = basic\ info.\ sec.\ \&\ cap.\ invest.) \wedge Read(\cdot, v_1^i, \cdot, v_2^{dv}, \cdot) \wedge (v_2^{dv} = WpHG\ customer\ information) \big) \Rightarrow \exists v_2^i, v_{s2}^t, v_{e2}^t, v_1^r : \big(Start(v_{s2}^t, v_2^i, ensure\ acknowledgement) \wedge End(v_{e2}^t, v_2^i) \wedge (v_{s2}^t \leq v_{e2}^t) \wedge Write(\cdot, v_1^i, customer\ legitimation, v_2^{dv}, \cdot) \wedge Perform(v_2^i, v_1^r) \wedge InStaff(customer\ advisor, v_1^r) \wedge (v_{e1}^t < v_{s2}^t) \big) \Big)$
R4	$\forall v_1^i, v_{s1}^t, v_{e1}^t : \Big(\big(Start(v_{s1}^t, v_1^i, conduct\ custody\ account\ contract) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \big) \Rightarrow \exists v_2^i, v_2^t, v_1^{dv}, v_2^{dv}, v_3^{dv}, v_4^{dv}, v_5^{dv}, v_6^{dv} : \big(Send(v_2^t, v_2^i, send\ legitimation\ and\ custody\ account\ documents, market\ support) \wedge (v_{e1}^t < v_2^t) \wedge Write(\cdot, v_1^i, account\ contract, v_1^{dv}, \cdot) \wedge (v_1^{dv} = v_3^{dv}) \wedge Write(\cdot, v_1^i, customer\ legitimation, v_2^{dv}, \cdot) \wedge (v_2^{dv} = v_4^{dv}) \wedge Read(\cdot, v_2^i, account\ contract, v_5^{dv}, \cdot) \wedge (v_5^{dv} = v_3^{dv}) \wedge Read(\cdot, v_2^i, customer\ legitimation, v_6^{dv}, \cdot) \wedge (v_6^{dv} = v_4^{dv}) \big) \Big)$
R5	$\forall v_1^i, v_{s1}^t, v_{e1}^t, v_1^r, v_1^r : \Big(\big(Start(v_{s1}^t, v_1^i, conduct\ investment\ advise) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \wedge Perform(v_1^i, v_1^r) \wedge InStaff(v_2^r, v_1^r) \big) \Rightarrow \big(con_{sec\_comp \geq level\ C}(v_2^r) \wedge rel_{is}(v_2^r, customer\ advisor) \big) \Big)$
R6-1	$\forall v_1^i, v_{s1}^t, v_{e1}^t : \Big(\big(Start(v_{s1}^t, v_1^i, conduct\ customer\ identification\ and\ legitimation) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \big) \Rightarrow \exists v_1^r : \big(Perform(v_1^i, v_1^r) \wedge InStaff(customer\ advisor, v_1^r) \big) \Big)$
R6-2	$\forall v_1^i, v_{s1}^t, v_{e1}^t : \Big(\big(Start(v_{s1}^t, v_1^i, check\ suspected\ money\ laundering\ case) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \big) \Rightarrow \exists v_1^r : \big(Perform(v_1^i, v_1^r) \wedge InStaff(anti\text{-}money\ laundering\ officer, v_1^r) \big) \Big)$
R7	$\forall v_1^i, v_{s1}^t, v_{e1}^t : \Big(\big(Start(v_{s1}^t, v_1^i, conclude\ custody\ account\ contract) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \big) \Rightarrow \exists v_2^i, v_{s2}^t, v_{e2}^t, v_1^r : \big(Start(v_{s2}^t, v_2^i, check\ suspected\ money\ laundering\ case) \wedge End(v_{e2}^t, v_2^i) \wedge (v_{s2}^t \leq v_{e2}^t) \wedge (v_2^t < v_{s1}^t) \wedge Perform(v_1^i, v_1^r) \wedge InStaff(customer\ advisor, v_1^r) \big) \Big)$
R8	$\forall v_1^i, v_{s1}^t, v_{e1}^t : \Big(Start(v_{s1}^t, v_1^i, customer\ contact) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \Rightarrow \exists v_2^i, v_{s2}^t, v_{e2}^t : \big(Start(v_{s2}^t, v_2^i, update\ customer\ information) \wedge End(v_{e2}^t, v_2^i) \wedge (v_{s2}^t \leq v_{e2}^t) \big) \Big)$
R9-1	$\forall v_1^i, v_{s1}^t, v_{e1}^t, v_2^i, v_{s2}^t, v_{e2}^t, v_3^t, v_4^t, v_1^{dv}, v_2^{dv} : \Big(\big(Start(v_{s1}^t, v_1^i, conclude\ custody\ account\ contract) \wedge End(v_{e1}^t, v_1^i) \wedge (v_{s1}^t \leq v_{e1}^t) \wedge Write(\cdot, v_1^i, account\ contract, v_1^{dv}, \cdot) \wedge (v_1^{dv} = v_2^{dv}) \wedge con_{stockless}(v_2^{dv}) \wedge rel_{is}(v_3^t, New\ Year's\ Day) \wedge rel_{is}(v_4^t, New\ Year's\ Day) \wedge (v_{e1}^t < v_4^t) \wedge (v_3^t < v_4^t) \wedge (v_4^t - v_3^t = 1y) \big) \Rightarrow \Big(\big(\exists v_5^i, v_5^t, v_3^{dv} : \big(Receive(v_5^t, v_5^i, payment, customer) \wedge Parameter(v_5^i, amount, v_3^{dv}) \wedge con_{\geq 5Eur}(v_3^{dv}) \wedge (v_3^t < v_5^t) \wedge (v_5^t < v_4^t) \wedge (v_3^t < v_5^t) \big) \Big) \vee \Big(\exists v_6^i, v_{s6}^t, v_{e6}^t, v_4^{dv} : \big(Start(v_{s6}^t, v_6^i, dissolve\ custody\ account) \wedge End(v_{e6}^t, v_6^i) \wedge (v_{s6}^t \leq v_{e6}^t) \wedge Read(\cdot, v_6^i, account\ contract, v_4^{dv}, \cdot) \wedge (v_2^{dv} = v_4^{dv}) \wedge (v_3^t < v_{s6}^t) \big) \Big) \Big) \Big)$

Table 10. *Cont.*

R9-2	$\forall \nu_1^i, \nu_{s1}^t, \nu_{e1}^t, \nu_2^i, \nu_2^t, \nu_1^{dv}, \nu_2^{dv}, \nu_3^{dv} : \Big(\big(Start(\nu_{s1}^t, \nu_1^i, \textit{dissolve custody account}) \wedge End(\nu_{e1}^t, \nu_1^i) \wedge (\nu_{s1}^t \leq \nu_{e1}^t)$ $\wedge\, Read(\cdot, \nu_1^i, \textit{account contract}, \nu_1^{dv}, \cdot) \wedge (\nu_1^{dv} = \nu_2^{dv}) \wedge Receive(\nu_2^t, \nu_2^i, \textit{security purchase}, \textit{customer})$ $\wedge\, Write(\cdot, \nu_2^i, \textit{account contract}, \nu_3^{dv}, \cdot) \wedge (\nu_3^{dv} = \nu_2^{dv}) \wedge (\nu_{e1}^t < \nu_2^t)\big)$ $\Rightarrow \exists \nu_3^i, \nu_{s3}^t, \nu_{e3}^t, \nu_4^{dv} : \big(Start(\nu_{s3}^t, \nu_3^i, \textit{reactivate account}) \wedge End(\nu_{e3}^t, \nu_3^i) \wedge (\nu_{s3}^t \leq \nu_{e3}^t)$ $\wedge\, Read(\cdot, \nu_3^i, \textit{account contract}, \nu_4^{dv}, \cdot) \wedge (\nu_2^{dv} = \nu_4^{dv}) \wedge (\nu_2^t < \nu_{s3}^t)\ \big)\ \Big)$

As opposed to *R2*, rule *R3* is triggered whenever the two brochures are provided. Thus, an antecedence occurrence node as well as antecedence data-flow connectors are used for the activity *provide brochure*. A consequence occurrence node and a sequence-flow connector specify the activity that has to follow. Similarly to *R2*, the latter activity has to be performed by the *customer advisor* as expressed by the consequence performing connector and instance resource node. The condition (i.e., antecedence pattern) of *R4* consists of the activity *conclude custody account contract*. Whenever this activity occurs, it has to create or write two documents as indicated by the consequence data-flow connectors and data objects. Both documents should be sent to the *market support* afterwards, as expressed by the consequence occurrence message that accesses the data objects. In *R5*, the performer of the triggering activity *conduct investment advice* is modelled as an antecedence resource node and constrained by an attached consequence resource condition and a consequence resource relation to the role *customer advisor*.

In brief, the two eCRGs for *R6* and *R7* express which roles should be held by performers of activities *conduct customer identification and legitimation*, *check suspected money-laundering case*, and *conclude custody-account contract*. In particular, only consequence performing connectors link the activities with respective roles. In addition, the consequence occurrence node and consequence sequence flow in *R7* require the activity *check suspected money-laundering case* to occur before any occurrence of the antecedence, i.e., activity *conclude custody-account contract*. *R8* is described by an eCRG whose nodes are not connected because *customer contact* (i.e., antecedence) requires an *update customer information* (i.e., consequence). However, the order of both activities does not matter, so that they are not connected by any sequence-flow connector.

Finally, two eCRGs are needed to describe *R9*. The antecedence (i.e., trigger) of the first eCRG consists of activity *conclude custody-account contract* which writes or creates a *stockless account contract* as expressed by the data flow edge, data object, and the attached data condition (all antecedence).

Furthermore, the two antecedent points in time describe any legal year due to their fixed distance of one year. Note the time condition attachment on the sequence-flow edge (both antecedence). Accordingly, this eCRG applies to every legal year that ends after a stockless account contract was concluded. In this context, at least one of the two consequences #1 or #2 must apply: either message *payment* with an *amount* ≥ 5 must be received within the year (consequence receive-message node with data attachment) or the custody account must be dissolved after the year has passed (consequence occurrence node).

The second eCRG of *R9* contains an antecedent-occurrence activity and receive-message node which both are connected to the same data object, which is part of the antecedence as well. Thus, when both activity *dissolve custody account* and message *security purchase* occur in the specified order and access the same data object, then activity *reactivate account* has to occur afterwards and access the same data object specified by the consequence-occurrence node and connectors.

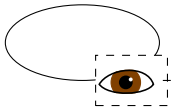
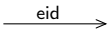
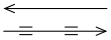
Two conceptual issues arise in the context of eCRG. First, the eCRG language needs explicit specifications of time intervals through separate start and end points in time. In order to express the simple phrase *per year*, more than five elements are required (cf. *R9*). Second, eCRGs lack explicit support for the reuse of permissions.

5.5. DMQL

The DMQL (*Diagramed Model Query Language* [35]) is a query language for process models. In the context of this language, conceptual models are interpreted as graphs composed of nodes and edges. Queries are specified using a graphical notation. In doing so, queries can be formulated independently of the modelling language in use. With regard to the theoretical background, DMQL is based on graph theory and combines various graph algorithms. Fundamentally, algorithms for subgraph isomorphism and subgraph homeomorphism have been combined and extended to accommodate the specific requirements of a model query language. Among the requirements are the ability to analyse the properties of nodes and to account for directed/undirected edges which may form a path in a model. Regarding paths, an important feature that is implemented in DMQL is the possibility to specify whether paths are allowed to overlap in cases where multiple paths are specified in a query. This problem refers to subgraph isomorphism, which, in theory, is known to be NP-complete [29]. However, DMQL makes use of algorithms that, nevertheless, can return results for real-world models. For more detailed information about the language (e.g., its syntax and semantics) as well as its prototypical implementation, we refer to [35].

The basic notations of DMQL consist of *nodes* (or vertices), *attributes* and *edges*, as illustrated in Table 11. Nodes correspond to constructs of a modelling language, such as a task or gateway in the *Business Process Model and Notation* (BPMN). For each node in the query, the range of allowed constructs of the underlying modelling language can be specified and a small symbol is added at the bottom right of the node shape. In cases where one construct is allowed, a small shape signifying this construct is added. In cases where multiple constructs are allowed, an eye symbol is added instead (cf., also, Table 11). Attributes correspond to properties of nodes, such as the name, description and other attributes that can be captured by a construct of the modelling language in use. Attributes are treated much like nodes, but for the sake of clarity, they are displayed using a rectangle with rounded corners. Edges can represent a large number of relations between the constructs of the modelling language in use. If BPMN is used, then edges may correspond to, e.g., sequence flow or information flow. In addition, the direction of the edges can be specified in relation to the direction of the edge in the original model. If the direction in the original model is from left to right, then the meaning of edge directions can be specified as depicted in Table 11. Moreover, multiple edges can also form paths.

Table 11. DMQL graphical notations.

Nodes		**Edge Directions**	
	Node		node
	Node type(s) indication		original
			opposite
			original, opposite
			none, opposite
	Attributes		none, original, opp

To signify a path, the solid line of the edges displayed in the concrete syntax in Table 11 would be dotted (the direction options remain the same). For paths, various properties can be specified, among them

1. The minimum and maximum path length;
2. The minimum and maximum number of overlapping edges and nodes between different paths contained in a query;
3. Required, allowed and forbidden nodes and edge types on the path, and
4. required, allowed or forbidden patterns on the path.

Since DMQL is built on top of GMQL (*Generic Model Query Language*) [61], it relies on the formalisms of GMQL when the query pattern is processed. In the following, the formalisation is explained by referring to GMQL. The basic idea behind GMQL is that a model consists of two sets of nodes, the set O of its objects and the set R of its relationships. Queries are then expressed in terms of set-modifying functions and operators, which can be processed efficiently by the developed algorithms implementing GMQL. Due to space limitations, we will not explain all sets and operators available but instead refer to [61] for an overview. Using these sets and operators, a formalisation of the first compliance rule *R1* can be given as follows.

```
DirectedPaths(
    ElementsWithTypeAttributeOfValue(
        ElementsOfType(O, IntermediateEventReceiveMessage),
        label, "Receive Customer Data")
    ElementsWithTypeAttributeOfValue(
        ElementsOfType(O, Activity),
        label, "Risk Assignment"))
```

Since the query pattern essentially consists of specifying a path between two known nodes, the GMQL-function `DirectedPaths()`, which matches paths between two sets of nodes, is used. These two sets, in turn, are specified via the function `ElementsWithTypeAttributeOfValue()`. This function takes a type definition given by the function `ElementOfType()` as well as an attribute (label) and its value as input parameter. GMQL also offers special functions for detecting the immediate neighbours and the relationship(s) between elements as well as standard set operators such as union and intersection. This is applied in the formalisation of compliance rule *R2*, given below.

```
ElementsDirectlyRelated(
    ElementsWithTypeAttributeOfValue(
        ElementsOfType(O, Activity), label, "Provide Brochure")
    ElementsWithTypeAttributeOfValue(
        ElementsOfType(O, Lane), label, "Customer Advisor"))
DirectSuccessors(
    Union(
        ElementsWithTypeAttributeOfValue(
            ElementsOfType(O, Document), label, "WpHG Customer Information"),
        ElementsWithTypeAttributeOfValue(
            ElementsOfType(O, Document), label, "Basic Info. Sec. & Cap. Inv.")),
    ElementsWithTypeAttributeOfValue(
        ElementsOfType(O, Lane), label, "Provide Brochure")))
```

Whereas the function `ElementsDirectlyRelated()` is used to match sets of connected nodes with undirected connections, `DirectSuccessors()` is used to match directed graph elements. Note that the two different documents which are handed over to the customer are aggregated to a set of nodes with the Union-operator. Figure 8 illustrates the DMQL mapping of the given compliance rules. Since DMQL is geared towards querying conceptual models, capturing compliance rules with DMQL also connotes selecting a modelling language. For this reason, we select BPMN as a standardized, widely used process modelling language. The chosen modelling language additionally specifies the semantics of the nodes used to construct the patterns.

Most requirements are easy to model in DMQL, except for *R3*, *R4* and *R9*. The receive-event in *R1* is modelled with a node representing a BPMN intermediate event of the type "catch", which has an outgoing path to the subsequent activity. An alternative model of this rule would also allow activities that can receive messages, not just events. In this case, the eye-symbol would be added to the node of the first pattern to express that multiple types of model elements are allowed. For the following rules, we do not comment on these sorts of decision since they are not relevant viewed from the perspective of semantics.

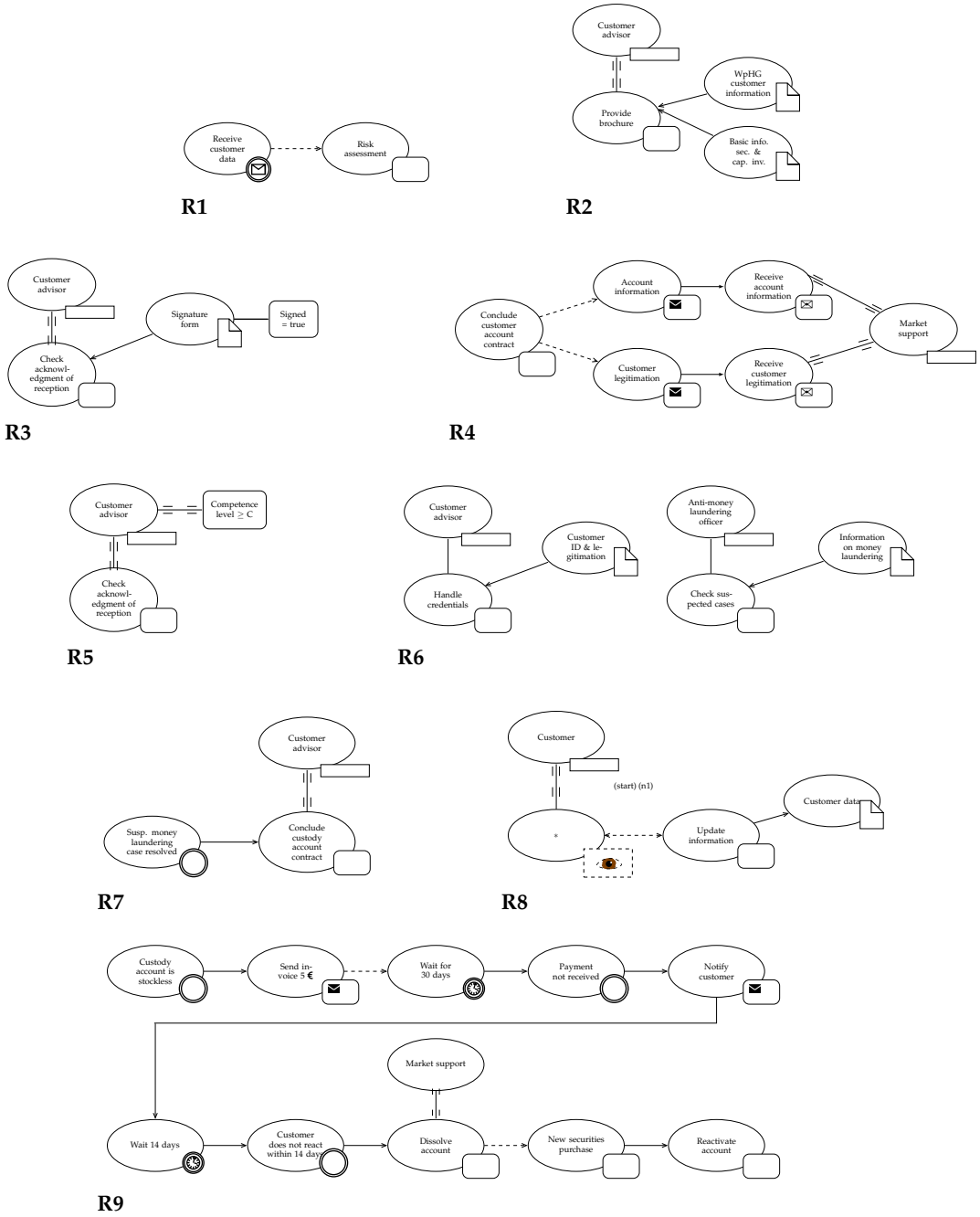


Figure 8. Formalised requirements in DMQL

The provision of the brochures in *R2* is modelled as an activity involving the two brochures. This activity is executed by a lane named *customer advisor*—since BPMN does not directly support the concept of a role or employee. Modelling it in this way is possible because DMQL allows implicit edges, i.e., the fact that an activity resides in a specific pool or lane and can be interpreted as an edge. *R3* cannot be modelled in a straight-forward manner, since it is unclear where and in which form the information that the customer has acknowledged the reception of the two brochures is stored. It can, however, be concluded that an acknowledgement may imply a signature on some form. The state of being signed, in turn, can be modelled as an attribute of the signature form. Similarly, *R4* cannot be modelled directly, because sending information to an organizational unit is not possible in BPMN (it may be possible in other languages). Instead, the required rule is modelled by two sending and corresponding receiving activities, whereby the latter is associated with

the pool *market support* via an edge. In this way, information can be sent to a department. *R5* can be easily modelled. The fact that the competence level of the *customer advisor* is above or equal to C can be expressed by simply relating an attribute to the node representing the lane *customer advisor*. In order to model *R6*, we need to interpret the word "while" in the natural language description as "and", i.e., as conjunction, and not in a temporal sense. Then, the rule can be represented by two activities processing the relevant information which are associated with the correct lanes. *R7* is modelled using a BPMN intermediate event of the type "catch", which waits until the process state is updated to *suspected money-laundering case is resolved*. *R8* is relatively easy to capture, since the description *every future customer interaction* is translated to a place-holder node (i.e., a node of any type) which is connected to a *customer* lane and which is on a path including an activity *update information* which writes to a *customer data* object. *R9* is more difficult to model due to the abstract formulation of the rule. For example, instructions such as "charge with a fee of EUR 5 per year" leave open if there is any invoice and a time frame for the payment. The modelled solution of this compliance query could even be extended—for example, the customer should be notified upon account dissolution or reactivation.

5.6. PCL

PCL (*Process Compliance Language*, [37]) is a formal logic for the specification of regulatory norms. The logic is based on defeasible logic [62,63] and A deontic logic of violations [64] which enables PCL to represent exceptions as well as modelling the violations of obligations.

It consists of a set of atomic symbols: a numerable set of propositional letters $a, b, c, \ldots$ representing state variables and the tasks of a process. The logic formulas are constructed using the deontic operators: $\neg$ (negative) and $\otimes$ (a non-Boolean connective). The deontic operators provide the support to specify the type of an obligation using subscripts and superscripts, i.e., O^x_y where the superscript x is used to specify the obligation modality (i.e., a specific type of obligation), and subscript y can be empty depending on the specific type of an obligation.

Table 12 illustrates the various obligation operators PCL provides for representing different types of obligations. Notice that PCL formulas can be can be transformed into RuleML or its advanced normative version LegalRuleML which can be used for automated processing [65,66]. The PCL formulas are written using the following construction rules:

(i) If every propositional letter is a literal (i.e., l), then its negation $\neg l$ is also a literal;
(ii) If X is a deontic operator (i.e., a specific type of an obligation) and l is a literal, then Xl and $\neg Xl$ are deontic literals.

More intuitively, (an atomic) p and $\neg p$ (its negation) is a proposition, while $\sim p$ to model negation p i.e., if $p = l$, then its complement $\sim p = \neg l$, and vice versa, if $p = \neg l$ then $\sim p = l$. Accordingly, if p is a proposition, then O_p (for obligation), P_p (for permissions) and F_p (for prohibition) propositions, respectively. In addition, the logic also introduces the notion of $\otimes$-expressions, (called reparation chains) so that every literal is an $\otimes$ expression; if $l_1, \ldots, l_n$ are literals, then $l_1 \otimes, \ldots, \otimes l_n$ is a reparation expression (see [37] for detailed semantics of the logic).

Table 13 illustrates the PCL mapping of compliance rules. It is trivial to capture the intuition of rules *R1* through *R4*. Rule *R2* provides the data condition where the investment consultant must provide two mandatory brochures to the customer.

PCL does not provide operators (or patterns) that can be used to explicitly model data conditions. However, constraints involving the process aspects (e.g., data, time, etc.) can be defined as literals in the PCL formulas.

Table 12. PCL obligation operators (adopted from [37]).

Operator	Intuitive Reading
$O^{a,\pi}_{pr}$	achievement, persistent, preemptive [OAPP]
$O^{a,\pi}_{n-pr}$	achievement, persistent, non-preemptive [OAPNP]
$O^{a,\tau}_{pr}$	achievement, non-persistent, preemptive [OANPP]
$O^{a,\tau}_{n-pr}$	achievement, non-persistent, non-preemptive [OANPNP]
O^{m}	maintenance [M]
O^{p}	punctual [P]

Table 13. Formalised requirements in PCL [37].

ID	Compliance Requirements
R1	$PerformRiskAssessment \Longrightarrow_{[OAPNP]} ObtainCustomerData$
R2	$NewCustomerAccount \Longrightarrow_{[OAPNP]} ProvideBrochure1, ProvideBrochure2$
R3	$NewCustomerAccount, ReceivedBrochures \Longrightarrow_{[OAPNP]} AcknowledgeReceptionofBrochures$
R4	$NewCustomerAccount, ConcludeCustodyAccountContract \Longrightarrow_{[OAPNP]} SendAccountDocuments,$ $SendCustomerLegitimation$
R5	$SecuritiesStaff, CompetenceLevelCorAbove \Longrightarrow_{[P]} ConductInvestmentAdvice$
R6	(1) $NewCustomer, CustomerAdvisor \Longrightarrow_{[OM]} ConductCustomerIdentification, ConductCustomerLegitimation$
	(2) $NewCustomer, AntiMoneyLaunderingOfficer \Longrightarrow_{[OM]} CheckSuspectedMoneyLaunderingCase$
R7	(1) $NewCustomer, ResolvedMoneyLaunderingCase \Longrightarrow_{[P]} ConcludeCustodyAccount$
Alternatively	
	(2) $NewCustomer, UnresolvedMoneyLaunderingCase \Longrightarrow_{[OM]} \neg ConcludeCustodyAccount$
R8	$ExistingCustomer, CustomerContact \Longrightarrow_{[OM]} UpdateCustomerInformation$
R9	(1) $StocklessCustodyAccount \Longrightarrow_{[OM]} Pay5EuroFee \otimes DissolveCustodyAccount$
	(2) $DissolvedCustodyAccount, NewSecuritiesPurchase \Longrightarrow_{[OAPNP]} ReactivateAccount$
Alternatively	
	(1) $StocklessCustodyAccount \Longrightarrow_{[OAPNP]} Pay5EuroFee$
	(2) $StocklessCustodyAccount, \neg Pay5EuroFee \Longrightarrow_{[OM]} DissolveCustodyAccount$
	(3) $DissolvedCustodyAccount, NewSecuritiesPurchase \Longrightarrow_{[P]} ReactivateCustodyAccount$

R5, on the other hand, implicitly stipulates a prohibition preventing staff lower than level C from providing securities advice. It is not possible to directly model prohibitions with PCL semantics as it does not provide patterns (or operators) to express prohibitions. Instead, they can be modelled as permissions, because deontic logic assume a strong relationship between obligations and permissions, where permissions are considered as the lack of obligations. Hence, they can be modelled as the dual of each other i.e., $O\psi \stackrel{\text{def}}{=} \neg P\neg\psi$ or $\neg O\neg\psi \stackrel{\text{def}}{=} P\psi$ (most variants of deontic logic largely assume that the duality relation between permissions and obligations ensures the consistency between the sets of norms [67,68].), and O and P are operators for obligations and permissions. This relationship can be written as: $O\psi \longrightarrow P\psi$. Essentially, the formula is equivalent to $O\psi \longrightarrow \neg O\neg\psi$, meaning that the prohibition of obligation $O\psi$ is not to do $O\psi$ i.e., $F\psi = \neg O\psi$. Notice that for a permission to be effective, a prohibition must exist.

R6 prescribes the typical segregation of duty (SoD) constraint where a suspected money-laundering case is handled by the anti-money-laundering officer. Essentially, the SoD constraints ensure that the relevant activities are assigned to the right/appropriate personnel. While *R6* is a structural rule, we need to model it separately because PCL does not provide structural patterns (or operators) to cover such constraints. Instead, it can be broken down into two distinct PCL formulas, each covering a case for both customer advisor and the anti-money-laundering officer.

In contrast, R_7 expresses an exception in which customers can conclude a custody account if a resolution of an unresolved money-laundering case is achieved. Essentially, the language must have the ability to model exceptions, which is important to correctly understand the distinction between strong and weak permissions. In particular, in law, strong permissions express exceptions to obligation while a permission provides an exception to a prohibition [69,70]. Hence, *R7* can be modelled as either a permission or a maintenance obligation, where the permission represents the exception to the obligation. In contrast, modeling the rule as a maintenance obligation of not concluding a custody account until a money-laundering case is solved represents a prohibition. It has been argued [25] that maintenance obligations are suitable for representing prohibitions since maintenance obligations must be complied with for all the instants of the interval in which the obligation is in force.

Rules *R8* and *R9* specify temporal and data conditions. The rules present maintenance obligations, which must be complied with at all instances. As discussed previously, it is not possible to explicitly model the conditions of the rule relevant to process aspects; instead, such conditions can be represented as literal directly within the PCL formulas by means of control tags [5]. While control tags facilitate annotating various constraints, the data for the control tags can be provided either by the analysts or they can be directly extracted from the databases attached to the process [71]. While PCL is able to properly capture majority of rules, the language is not able to capture perdurant obligations [72], which was later addressed. In addition, nested rules and recursive compensations cannot be modelled in the current variant of PCL.

5.7. PENELOPE

PENELOPE (*Process Entailment from Elicitation of Obligations and Permissions*, [39]) is a framework which captures compliance requirements relevant to tasks of business processes. The framework validates the compliant behaviour of processes at design-time, for which it uses a proprietary algorithm which computes all active deontic assignments to generate control flow and state space. The generated state space is a set of deontic assignments which are active in a particular task. The interaction between the generated process models flows from state to state, and all the states are computed until no obligation or permission holds at a state, or a violation cannot be compensated. Upon computing all states, the algorithm draws the BPMN model in which tasks represent a set of obligations fulfilled by a role, and errors and end events model violations of an obligation or permission by a role at a state.

Table 14 illustrates that the deontic assignments in PENELOPE are modelled using a well-known event-based logic, EC (Event-Calculus, [73,74]). The logic, which provides a rich set of predicates, expresses various types of states of event occurrences e.g., Happens (occurrence of an event at a time point), Initiates (an event triggers the property of a system), Terminates (an event terminates the property of the system), and HoldsAt (that the property of a system holds at a time-point). In addition, some auxiliary predicates express the premature termination (Clipped) and resumption (Declipped) of an event at a particular point in time between the interval. The InitiallyTrue and InitiallyFalse allows the modelling of the system's states where only partial information about the domain is available. In contrast, the domain-independent axioms describe the states in which a variable (fluent) holds or does not hold at a particular point in time; see [74] for detailed semantics of EC.

Table 14. PENELOPE's deontic properties [39].

Terms	Meanings
$Oblig(\pi, \alpha, \delta)$	*agent* π *must perform the activity* α *by due date* δ
$Per(\pi, \alpha, \delta)$	*agent* π *can perform the activity* α *prior to due date* δ
$CC(\pi, \alpha_1, \delta_1, \alpha_2, \delta_2)$	*agent* π *must perform activity* α_2 *by due date* δ_2 *after activity* α_1 *is performed prior to the due date* δ_1
$(A)\ Terminates(\alpha, Oblig(\pi, \alpha, \delta), \tau) \longleftarrow \tau \leq \delta$	
$(B)\ Terminates(\alpha, Per(\pi, \alpha, \delta), \tau) \longleftarrow \tau \leq \delta$	
$(C)\ Happens(violation(Oblig(\pi, \alpha, \delta)), \delta) \longleftarrow HoldsAt(Oblig(\pi, \alpha, \delta)) \wedge \sim Happens(\alpha, \delta)$	
$(D)\ Initiates(\alpha_1, Oblig(\pi, \alpha_2, \delta_2), \tau) \longleftarrow \tau \leq \delta_1 \wedge HoldsAt(CC(\pi, \alpha_1, \delta_1, \alpha_2, \delta_2)), \tau)$	

Table 15 depicts the mapping of rules relevant to the process model using PENELOPE properties. (In presenting the mappings, we use PENELOPE properties that are based on the standard variant of EC; however, there are other variants of EC giving different set of predicates and events; see [75] for details.) There are a number of issues with this representation—in particular, from the representation and reasoning perspective of norms. For example, modeling rules *R1* through *R3* is straight forward but *R5*, which prescribes an explicit permission, cannot be properly represented because PENELOPE's permission predicate $Per(\pi, \alpha, \delta)$ provides a parameter for specifying deadlines, i.e., δ. Since permissions cannot be violated [72], no deadlines are required. Essentially, modelling *R5* as permission is semantically correct, but it does not correctly capture the intuition of the rule, and is, thus, incorrect from a reasoning perspective.

In contrast, rules *R6* and *R8* prescribe a maintenance obligation which must be fulfilled for all instances of the interval in which the obligation is in force. Currently, PENELOPE semantics only provide temporal information, i.e., a deadline to detect the violations which cannot be used for reasoning about the maintenance obligation. A deadline for a maintenance obligation would signal the obligation is no longer in force [76], but a maintenance obligation remains in force for the whole duration of the interval in which it is in force.

R4 and *R7* can be simply modelled as conditional commitments, for which PENELOPE provides conditional patterns, i.e., CC, whose main idea, as in most business scenarios and concrete applications, is that an entity commits itself to another entity so that it can produce some effects required to correctly execute that entity. Essentially, conditional patterns can be useful from a structural perspective, since a conditional commitment might model the absence of an occurrence of an activity, meaning that entity *B* cannot occur until the occurrence of entity *A* [25]. In addition, *R7* is a maintenance obligation, which cannot be explicitly modelled with PENELOPE. However, in PENELOPE, prohibitions are not considered under the close world assumption [77], and are implicitly assumed when no obligations or permissions can be derived. PENELOPE does not provide deontic properties for modelling prohibitions. However, one possible way to model prohibitions with PENELOPE semantics could be modelling them as negative permissions, since permissions and obligations have a duality relation (cf. Section 5.6). However, modelling *R7* as a negative permission would not properly capture the intuition of the rule. Finally, modelling *R9* is easy as it prescribes an achievement obligation which PENELOPE can effectively model.

Table 15. Formalised requirements using PENELOPE patterns in EC.

ID	Compliance Requirements
R1	$Initiates(ObtainedCustomerData(Advisor, NewCust), Oblig(Advisor, PerformRiskAssessment, \delta), \tau) \longleftarrow$ $Happens(ObtainedCustomerData, \tau) \wedge HoldsAt(ObtainedCustomerData, \tau) \wedge \tau \leq \delta$
R2	$Initiates(NewCustAccount, Oblig(Advisor, Provide(Broch1, Broch2^{\dagger}), \delta), \tau) \longleftarrow$ $Happens(NewCustAccount, \tau) \wedge HoldsAt(NewCustAccount, \tau) \wedge \tau \leq \delta$
R3	$Initiates(ProvidedInfoBrochures, Oblig(Advisor, EnsureAcknolwedgement, \delta), \tau) \longleftarrow$ $Happens(ProvidedInfoBrochures, \tau) \wedge HoldsAt(ProvidedInfoBrochures, \tau) \wedge \tau \leq \delta$
R4	$Initiates(ConcludedCAC, Oblig(Advisor, SendMarketSupport(ADs, CL), \delta_2), \tau) \longleftarrow$ $HoldsAt(CC(Advisor, ConcludedCAC, \delta_1, SendMarketSupport(ADs, CS), \delta_2), \tau) \wedge \tau \leq \delta$
R5	$Initiates(NewIA, Per(Advisor, ConductNewIA, \delta), \tau) \longleftarrow$ $HoldsAt(NewIA, \tau) \wedge HoldsAt(Advisor(ComptenceLevelCorAbove), \tau) \wedge \tau \leq \delta$
R6	R_6 provides a *maintenance* obligation which cannot be modeled with PENELOPE semantics.
R7	$Initiates(ResolvedAMLCase, Oblig(Advisor, ConcludeCustodyAccount, \delta_2), \tau) \longleftarrow$ $HoldsAt(CC(Advisor, ResolvedAMLCase, \delta_1, ConcludeCustodyAccount, \delta_2), \tau) \ \wedge \ \tau \leq \delta_1$
R8	R_8 is a *maintenance* obligation, which cannot be modeled with PENELOPE semantics.
R9	$Initiates(UnpaidFee, Oblig(MarketSupport, DissolveAccount, \delta), \tau) \longleftarrow$ $HoldsAt(StocklessAccount, \tau) \wedge (Happens(UnpaidFee, \tau) \wedge HoldsAt(UnpaidFee, \tau)) \wedge \tau \leq \delta$

Naming convention used in the mappings: Brochure1 (WpHG Customer Information), Brochure2 (Basic Information Securities and Capital Investment), ADs (account documents), CAC (custody-account contract), CS (customer legitimation), IA (investment advice), AML (anti-money laundering), PI (portfolio investment).

6. Evaluation of Compliance Languages Based on the Formalisation Results

In this section, we adopt the normative classification framework from [72] to compare the different languages from a legal reasoning perspective. In the second part of this section, we investigate the textual complexity of compliance rules which were discussed in Section 5. As this aspect has not been addressed by compliance research so far, we start with an outline and reflection on the different metrics before evaluating the complexity of language patterns based on Halstead's complexity metrics [78].

6.1. Expressiveness

Norms prescribe how individuals ideally should or should not behave, or, in other words, what they are permitted to do and what they have a right to do [79]. This idea of human behaviour includes the possibility that actual behaviour may deviate from the ideal from time to time and, thus, result in a violation of duties or, more precisely, obligations. Deontic logic is an area of logic that is concerned with representing and reasoning about the distinction between the actual and the ideal [80]. In linguistics, the word "deontic" is a modality that indicates how desirable, believable, obligatory, or veritable a proposition is [81]. The difference between a norm and normative proposition is that the former is prescriptive while the latter can be seen as descriptive [82].

Deontic logic has, hence, evolved into a formal system where normative concepts apply according to which a compliance rule can be forged from a number of theoretical constructs, i.e., obligation, permission or prohibition, logical connectors and constants for representing certain deontic concepts [83]. Defeasible logic extends the view of deontic modalities to reason about non-monotonic, i.e., defeasible, inferences, which allow for a distinction between cause and effect and the withdrawal of contradicting conclusions in light of new evidence [84,85]. In terms of compliance, the use of this particular family of logics implies that rules can be checked independently from any other modelling language using only the proof theory itself [86]. The approach proposed in [72] combines deontic

and defeasible logic to elicit differences in the legal understanding of norms that cannot be easily captured by temporal logic [87]. By adopting the approach, we aim to explore to what extent the selected languages support the modelling of these normative concepts.

Figure 9 illustrates the relevant classes and conditional relationships of deontic modalities. The framework builds on the three deontic rule types—obligation, prohibition, and permission—proposed by [88]. It is extended by a notion for compensations derived in [62]. Obligations are always considered in the context of a possible violation. A process is non-compliant if an obligation cannot be compensated. This means a previous specified action can replace and, hereby, compensate a violated obligation which would otherwise result in a violation. Prohibitions are the inverse of an obligation, meaning a prohibition of A is the obligation of ¬A and vice versa [89]. Permissions state what someone is entitled to do, and, thus, a permission can never be violated. In the following, we run through several examples to illustrate the different deontic effects.

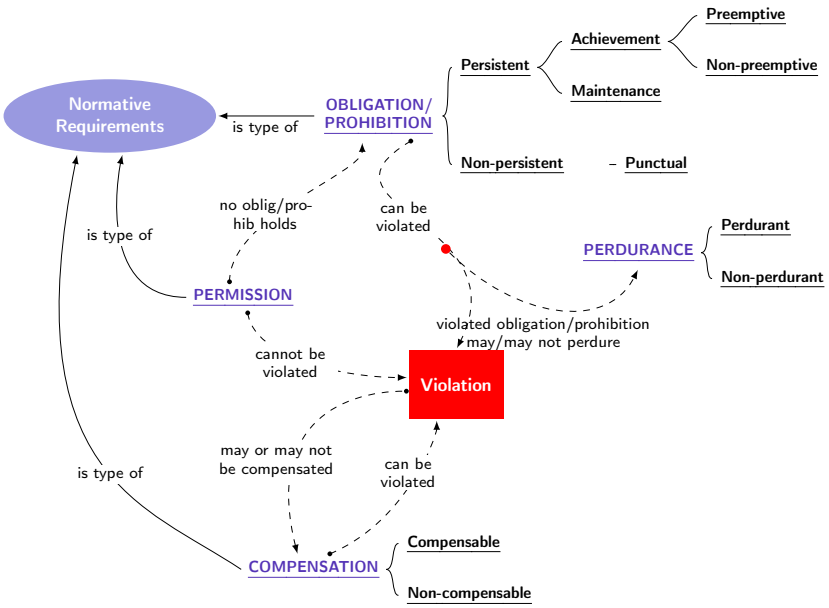


Figure 9. Classification of normative requirements (adopted from [72]).

Persistence: An important distinction is whether an obligation is terminated or removed. Consider an update of your online banking account. As soon as you have filled out all mandatory text fields, the *punctual* obligation expires and your session is terminated (*non-persistent*). We speak of a *persistent* obligation if the action linked to the obligation is removed, such as the standing order for a credit being terminated when it is completely paid back. Both achievement and maintenance are persistent obligations.

Achievement: This obligation type refers to a period of time in which the obligation must be met *at least once*. It can be further distinguished into preemptive and non-preemptive obligations. In general, it is obligatory for a train conductor to identify her/himself by means of a service card.

If the conductor presented a service card before conducting the ticket inspection, then the obligation to show the service card requested by the passenger during the ticket inspection has already been fulfilled and is categorised as a *preemptive* obligation.

Maintenance: This obligation type refers to a period of time in which the obligation must be *continually* met. Paying the monthly flat rate is an example of a periodical expenditure, whether it be the apartment rent or the high-speed internet.

Perdurance: A perdurant obligation persists after being violated. The payment of an invoice for an online purchase is, first of all, a *perdurant* obligation, but in case the purchase has to be returned to the retailer, this obligation becomes *non-perdurant*.

The formal definitions of all obligations types are given in [25]. Based on this formal distinction, we evaluate the rules modelled in Section 5. The results of the comparison are summarized in Table 16. A '+' symbol indicates which language intuitively captures the notion and provides full modelling support. Parentheses '(+)' are used to distinguish languages that do not explicitly support the deontic modality but feature a conceptual solution to the modelling task. The '−' symbol suggests that the deontic modality is either not supported, or not considered by the respective language.

The evaluation shows that the majority of languages is able to represent, essentially, three deontic modalities, namely, achievement, obligation, prohibition and violations of these. Beyond that, the comparison underlines that particularly punctual and perdurant obligations as well as permissions are only supported by a few languages. This shortcoming relates primarily to approaches using a language of the temporal logics family. These are being criticised as divagating from the norms' actual meaning and, thus, are not suitable to model certain normative requirements such as permissions [87]. The relevance of permissions, though, can be levelled in comparison to the other deontic modalities as they cannot be violated, and, therefore, would not hint at any compliance violation. Instead, they can be used to indicate that there are no obligations or prohibitions in effect [90]. A rather unexpected result of this evaluation is that, even though a corresponding deontic modality exists [25], in some cases a deviation seems to be unavoidable in order to grasp all the information encoded in the requirements. Consequently, an exact mapping between the deontic effects and a language' properties is not only a formal matter, but also a language-dependent choice.

The two pattern-based approaches BPMN-Q and CRL support achievement and maintenance obligations as well as prohibitions, violations and compensations.

For modelling the sample requirements in CRL, we use the *LeadsTo*, *Precedes* and *Frees* patterns; the resource patterns *PerformedBy* and *SegregatedFrom* as well as a *LeadsTo-Else* compensation. Contrary to the normative representation suggested in Table 13, we model *R1*, *R3* and *R4* not as achievement obligation but with patterns that depend on the relation between two activities. *R5* constitutes a permission, which is substituted by a resource pattern. Likewise, the two maintenance obligations described in *R6* are summarised under one resource pattern. *R7* can either be modelled as negated permission or maintenance obligation (cf. Table 13). However, the rule has been treated as an exception as it seems to have a similar effect in this particular case. *R8* describes a maintenance obligation, but, due to its implied ordering relation, we decided to model the rule by an order pattern instead.

Table 16. Expressiveness from a legal-reasoning perspective.

Language	**Deontic Modalities**									
	Punctual	**Achievement**	**Preemptive**	**Non-preemptive**	**Maintenance**	**Perdurant**	**Permission**	**Prohibition**	**Violation**	**Compensation**
BPMN-Q	−	+	−	−	+	−	−	+	+	(+)
CRL	(+)	+	−	−	+	−	−	+	+	+
Declare	(+)	+	+	+	+	−	−	+	+	(+)
DMQL	+	+	+	+	−	−	−	+	(+)	(+)
eCRG	+	+	+	+	+	+	(+)	+	+	+
PCL	+	+	+	+	+	+	+	+	+	+
PENELOPE	−	+	−	−	−	−	+	−	−	−

In the case of BPMN-Q, we chose between the *Before-scope/Global-scope Presence* and *Conditional Presence/Response* patterns to capture the given compliance rules. Owing to the absence of an activity or a start/end event in the textual requirement, the data conditions modelled in *R2* and *R7* follow no explicit BPMN-Q pattern. Hence, a mapping to the deontic modalities is not possible. Unlike CRL, the resource perspective is not supported by BPMN-Q resulting in the omission of *R5* and *R6*. With regard to their normative meaning, we deviated only in *R4*, which is modelled with a *Conditional Response* pattern instead of an achievement obligation.

Violations are handled differently by the two languages. BPMN-Q defines anti-patterns of the existing pattern set that are structurally matched to the process model and verified after relevant areas have been transformed into a Petri-net.

CRL patterns, on the other hand, are annotated to the process model and then verified on the basis of the event log. To model compensations, such as *R9*, CRL proposes a composite pattern, composed of the *LeadsTo* (or *DirectlyFollowedBy*) pattern and the conditional *Else*, which is followed by one or more repairing actions [33]. The repairing actions are enabled if the predecessor is violated. In BPMN-Q, compensations can only be imitated by repeating the respective pattern. Beyond that, CRL verbalizes exceptions to explicate under which condition an activity holds [91]. Such exceptions are initiated by the *Frees* pattern. In addition to the discussed deontic modalities, BPMN-Q and CRL also propose several intuitive control-flow patterns which are commonly used for the ordering of activities [28,44]. However, these patterns target the behavioural relation between two activities and do not correspond to any of the described deontic modalities.

In DMQL, punctual requirements can be modelled as directly connected nodes (or nodes connected with a path of specified maximum length). Achievements and prohibitions can be modelled via a combination of paths, and allowed or forbidden elements on paths. If, thereby, a path is established between a specified node and an arbitrary (i.e., place-holder) node, then this implicitly equals a before-scope or after-scope pattern known from other languages, such as BPMN-Q. In this way, preemptive obligations can be modelled by a path going from the start event of a process model to the activity that fulfils the obligation, followed by a path of either DMQL or BPMN constructs. Violations and compensations, in contrast, can be represented implicitly using the BPMN constructs of attached events for capturing violations occurring at run-time or compensating activities for compensation.

Declare supports achievement and maintenance obligations as well as prohibitions, violations and compensations. Achievement obligations are specified by the Responded_Existence constraint (cf. *R8*) and other similar constraints. In particular, preemptive obligations are expressed by Precedence constraints (e.g., in *R1*) and non-preemptive obligations are expressed by Response constraints (e.g., in *R4*). Further, Existence constraints can be used to specify maintenance obligations (cf. *R2*). Prohibitions, on the other hand, are represented by negations of other constraints, such as the Responded_Absence (i.e., negation of Responded_Existence). The Chain_Succession constraint supports specifying punctual obligations. In order to detect violations, Declare is mapped onto LTL expressions that can be verified using model checking and related techniques. Note that Declare provides only limited support for compensations that can only be realized by using choice templates. However, an order between the different solutions cannot be specified. Finally, Declare does not provide support for perdurant obligations or permissions.

eCRG supports describing and deciding on most obligations. Punctual obligations can be modelled by combining activity or point-in-time nodes with data and resource conditions as well as relations (cf. *R5*). The specification of maintenance obligations is realized through antecedence occurrence nodes (cf. *R6* and *R7*). In turn, consequence sequence-flow connectors support all kinds of achievements (cf. *R1*, *R3*). In particular, the combination of sequence-flow connectors with temporal conditions also enables specifying perdurant obligations. Moreover, eCRG models preemptive and non-preemptive obligations based on antecedence and consequence occurrence nodes and consequence sequence-flow connectors. In cases of a preemptive obligation, the sequence-flow connector

shows from the consequence to the antecedence node (e.g., *R1*), and vice versa in the case of a non-preemptive obligation (e.g., *R4*). Prohibitions are expressed by consequence-absence nodes. However, eCRGs lack explicit support for the reuse of permissions as, for example, provided by PCL. As a consequence, eCRGs have to explicitly remodel the conditions of a certain permission, when earlier specified permissions are used for defining the scope or trigger of another compliance rule (e.g., an agent with permission for activity A is prohibited from performing activity B). The eCRG monitoring framework [26] describes how various copies of eCRGs are annotated with symbols, colours and text, in order to reflect its current state and to detect compliance violations. Finally, eCRG supports the modelling of compensations through the use of multiple consequence patterns (cf. *R9*). This solution is technically correct as it allows the exact recognition of compliant traces. However, the information about the desired consequence is lost as well as the order in which compensations are considered. Note that, as a workaround, the numbers of the different consequence patterns may be used to indicate this information (e.g., consequence pattern 1 is, per convention, the desired consequence, and pattern 2 the first compensation in case 1 is violated).

PCL can represent and reason about all obligation modalities, thanks to its deontic and non-monotonic properties and the formal logic it uses. Apart from intuitively modelling basic modalities, PCL is also able to effectively reason about the violations and, especially, time-varying properties of obligations, e.g., achievement, punctual obligation and the persistent effects over time. For example, for *R2*, we chose to model the violation. The second part, on the other hand, is modelled as achievement obligation. In contrast, *R8* provides an interesting case of exception and is modelled as permission. As discussed above, for a permission to be effective a prohibition must exist. In case of *R8*, it is prohibited to conclude a custody account until a solution to the unresolved money-laundry case is reached. Since prohibitions can be expressed as negative obligations such that, since $O\psi \leftarrow \neg O\neg\psi$, if it is obligatory to perform ψ, then its negation $\neg\psi$ is not (in legal theory, a permission is considered as the absence of the obligation to the contrary [25].). Thus, modelling the prohibition as permission is a natural choice. In the alternative representation, a prohibition is modelled as maintenance obligation because it has been argued that maintenance obligations are suitable to represent prohibitions (see [87] for details). As far as perdurant obligations are concerned, the current variant of PCL is not able to represent perdurant obligations [72]. However, this issue has been addressed in [86], enabling PCL to effectively capture all types of normative requirements.

PENELOPE, on the other hand, is able to represent achievement obligations and permissions while other requirements cannot be modelled. This is due to EC, which is a first-order logic. It has been argued that only higher order logic can properly capture deontic notions [92]. In addition, EC's $Initiates(E, X, T)$ predicate meaning is that the event E at time T initiates the fluent X, and the fluent starts to hold from the next instant of time. This would effectively mean that an obligation does not start to hold from the time when the event is triggered, but from the next instant, which might not be the case for legal norms. Hence, it is not possible to capture when the obligation becomes effective. On the same note, there might be cases where a norm becomes effective at the same time instant as the triggering event occurs. However, EC's Initiates predicate is not able to capture such cases (such as punctual obligations) [76]. This is similar for EC's other predicates, such as the Terminates predicate suffering from the same problems as the *Initiates* predicate because it takes the same number of parameters. Hence, the Terminate predicate may not properly represent when an obligation Terminates. Although, with PENELOPE, it is possible to detect violations, it is currently not possible to correctly reason about violations. For example, in PENELOPE, violations are generally detected at deadline, i.e., δ, which is not correct from the reasoning perspective. In case of an achievement obligation, the violation must be detected at $\delta + 1$ not at δ. Nor are compensations and perdurant obligations supported [25]. In addition, the notion of a permission cannot be expressively represented by PENELOPE's patterns, even though the notion is explicitly supported by the language.

6.2. Complexity

Metrics are traditionally applied in software engineering to manage the complexity of software, and thereby provide means to improve the software development process and assure software quality [93]. Along with object-oriented programming and program modularisation, metrics have become an important measure to analyse and reduce software complexity at virtually all stages of the software development process. The best known examples are Halstead's Complexity [78], McCabe's Cyclomatic Complexity [94], Henry and Kafura's Information Flow [95] and, finally, the many variants of *Lines of Code* (LOC) [96]. Their usefulness for the detection and prediction of errors in process models has been investigated in [97], where metrics are picked up to retrieve patterns that are less likely to result in formal modelling errors. However, measuring the complexity of compliance rules creates a new application area for these metrics and, with it, new problems.

Halstead was one of the first to develop a measure for the effort required to generate a program from simple counts of distinct operators and operands and their total frequencies [98]. A main point of criticism, though, is that the metrics lack a theoretical foundation which clearly states what to count to make the results more comparable [99]. Another restriction is that the metrics merely concentrate on the lexical complexity rather than the program's structure. McCabe's metric, on the other hand, is computed based on the control-flow graph of a program [94,100]. It measures the number of paths that exist to traverse a code sequence, and hereby serve as an indicator of the unstructuredness of a program [99]. In this regard, the nodes of a graph correspond to a set of indivisible code sequences, while a directed edge indicates a link between two nodes if the second code sequence can be executed directly after the first one [100]. Given the restriction to the process structure and the semantic discrepancy between graph and program code [101], the metric (in its current state) seems to be inappropriate to differentiate between the diverse process perspectives with which compliance rules are concerned.

The importance of measuring inter-modular factors is stressed by Henry and Kafura, who calculate the data flow as incoming and outgoing data calls with respect to the LOC of the different modules of a program [95]. LOC counts the number of semicolons in a method, except for those within comments and string literals. Although many variations, including *Source Lines of Codes* (SLOC) or *Lines of Codes Equivalents* (LOCE) [100], have been proposed, the measure can hardly be transferred to the generally much more compact compliance rules, which neither benefit from their segmentation into lines (SLOC) nor from weights for the nesting of code (LOCE), without conceptual groundwork.

Thus, even if, in general, metrics can be considered a powerful measuring instrument to estimate a program's complexity, only Halstead's metrics appear to be directly transferable to compliance rules without further modifications. Being formerly contested for its universality for programming languages, its classification into operators and operands now facilitates a comparison across different languages requiring only an a-priori declaration of operators and operands. Hence, we decided to apply Halstead's metrics to estimate the complexity of the effort to write (or read) a compliance rule. The Halstead metrics are defined as follows:

$$\textbf{Vocabulary} \quad n = n_1 + n_2 \tag{1}$$

$$\textbf{Length} \quad N = N_1 + N_2 \tag{2}$$

$$\textbf{Volume} \quad V = N \cdot log_2(n) \tag{3}$$

$$\textbf{Difficulty} \quad D = \frac{n_1}{2} \cdot \frac{N_2}{n_2} \tag{4}$$

$$\textbf{Effort} \quad E = D \cdot V \tag{5}$$

where n_1 (n_2) is the number of unique operators (operands) and N_1 (N_2) is the total number of operators (operands).

To achieve a uniform evaluation basis, we limited *operators* in compliance rules to commands and structuring elements such as patterns, literals, logical connectors, start/end

events, flows or parentheses; and *operands* to elements that have a (fixed) value such as variables or constants, which are usually represented by activities or data attributes. Note that Halstead's metrics are not calculated per language, but per concrete program in respect to the characteristics of the language. Accordingly, all operators and operands are counted per compliance rule. As this paper's focus lies on the end-user perspective, the metrics are only applied to the high-level language. The following example demonstrates how the measures for the metrics are counted.

Example: Figure 10 shows three different compliance rules. The graph is modelled in BPMN 2.0 while the textual rules are specified in CRL and PCL. Activities are simply represented by the letters A, B and C. According to the definition of the metrics, *R10* can be expressed by five operators: n_1 = {xor-split, xor-join, start event, end event, sequence flow} = 5 and two operands: n_2 = {A, B} = 2. After defining the language's properties, their occurrence can be counted. The operators defined in n_1 result in ten, as there are multiple sequence flows, two gateways and two events to consider: N_1 = 10; whereas the operators defined in n_2 appear only once each, and, hence, result in two: N_2 = 2. Similarly, *R11* can be decomposed into three operators: n_1 = {pattern, and, parentheses} = 3, where the parentheses are counted as a pair; and three operands: n_2 = {A, B, C} = 3 resulting in: N_1 = 3, respectively N_2 = 3. Analogously, *R12* can be determined by three different operators: n_1 = {deontic modality, $\otimes$-reparation} = 2, and three operands: n_2 = {A, B, C} = 3 resulting in: N_1 = 2, respectively N_2 = 3.

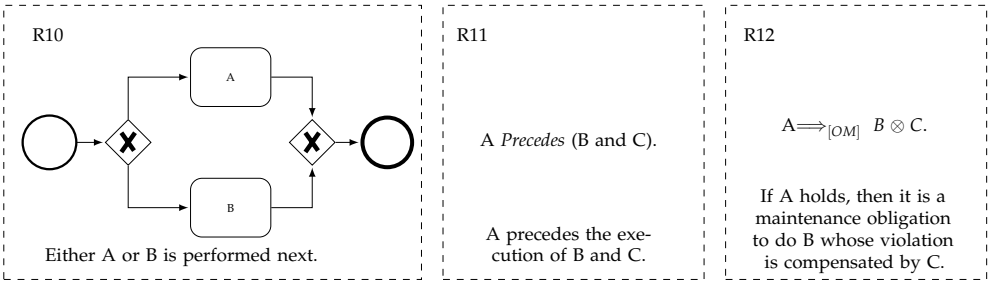


Figure 10. Example of the operator and operand count

Following this example, the metrics calculation results in two lists per language (cf. Appendix A). The first list details the assumptions which quantify the rules, while the second list compiles the individual values for *R1* to *R9*. The metrics compute the effort based on the number of operators and operands that take part in the evaluation of the rule. Simply speaking, the vocabulary (length) of a language increases depending on the elements (size) of the rules.

As can be seen from Table 17, the lowest efforts ($E = 15.76/16.36$; $E = 15.92$) are computed for PCL and Declare. This fact could be linked to the linear complexity of deontic and defeasible logic as the base language of PCL. The complexity of the overall compliance problem is NP-complete, though. Declare on the other hand, cannot depict the data conditions or resource constraints given in *R5* and *R6*, which naturally limits the complexity of expressions. Looking at the measures used to calculate the effort, it becomes clear that both volume ($V = 62.05$; $V = 44.31$) and difficulty ($D = 3.94$; $D = 4.44$) are significantly higher for PENELOPE and eCRG, which ultimately lead to the highest efforts ($E = 267.51$; $E = 440.70$). In contrast to PCL, the complexity of the standard EC used in PENELOPE is exponential, assuming the relative time and partial order nature of the EC. Furthermore, each operator has its own complexity level, which contributes to the overall complexity. In addition, in the case of PENELOPE, we also have nested predicates which not only increase the size of the predicate but also the complexity level. In the context of eCRG, the specification of the time intervals in *R9* is the main driver for this issue. Only considering *R1–R9* significantly reduces the effort ($E = 91.93$), volume ($V = 24.21$) and difficulty ($D = 3.02$) of eCRG. DMQL shows a remarkable high volume

($V = 22.43$) at a comparably low difficulty ($D = 1.54$) leading to the second highest effort ($E = 47.75$). BPMN-Q ($E = 67.03$) and CRL ($E = 31.17$) come to more moderate and even similar results for difficulty, which can be explained by the fact that both approaches share a subset of patterns.

Table 17. Complexity measured by metrics proposed in [78].

Language	n	N	V	D	E	SD	Min.	Max.
BPMN-Q	4.67	8.00	16.89	2.79	67.03	100.76	R7	R9
CRL	3.44	5.11	9.66	2.25	31.17	41.49	R1/R8	R9
Declare	2.57	3.86	5.81	1.43	15.92	34.45	R2	R9
DMQL	5.67	8.56	22.43	1.54	47.75	64.20	R1	R9
eCRG	8.22	12.89	44.31	4.44	440.70	1,051.10	R8	R9
PCL	5.67	5.89	13.03	1.81	15.76/16.36 *	6.86/6.73 *	R1	R5
PENELOPE	12.14	17.29	62.05	3.94	267.51	197.73	R5	R4

* Two alternatives for R6 and R9. n, N, V, D and E are calculated as mean over R1 to R9 of the tables given in Appendix A. SD, Min. and Max. are calculated for E.

Examining the effort of the individual rules shows that it decreases for *R1* while it increases for *R4* and *R9*. *R1* relates to a quite common control-flow rule, for which most languages provide a predefined modelling construct so that the effort does not increase for most languages. The compensation incorporated in *R9*, on the other hand, affects the effort due to its higher semantical complexity. Remarkable, in this context, is that apart from *R9*, a simple sequence flow, such as described in *R4*, seems to raise the effort as well. The reason for this might be that *R4* features two activities and two data conditions at once, though the languages realise this rather differently. Overall, our analysis shows that the chosen modelling approach leads to varied outcomes not only by means of their expressiveness (cf. Section 6.1), but also with regard to their lexical complexity.

7. Related Work

In this section, we compare the presented evaluations to several existing surveys and evaluation studies reported in the literature. The presented work is complementary and different from existing evaluations. It is complementary in the sense that it provides more insights into the state of affairs in the compliance-requirements modelling domain. In contrast to other evaluations (such as the work of [91,102]), ours includes more frameworks and formal languages than any other work, and, thus, has a wider scope.

El Kharbili [103] analyses operational and (non-)functional aspects of regulatory compliance management (RCM) from a business-process management perspective using three categories as evaluation criteria. Using one of these criteria of the compliance dimensions, they extract three distinct types of rules, namely, *structural, temporal* and *contractual* rules, which are supported by the modelling languages. However, their evaluation lacks a systematic evaluation of "*legal requirements*" for compliance checking—in particular, from a reasoning perspective for compliance checking. Furthermore, they do not make any distinction between the classes of norms [72]; and they do not consider the specific class of norms and how they can be properly represented.

In the context of the larger project Reasoning on the Web with Rules and Semantics (REWERSE) Bonatti et al. [104] investigate the state of affairs for logic-based modelling languages for representing policy, trust, actions and business rules in the security and policy domains. Ly et al. [105] evaluate the core functionalities of existing compliance monitoring approaches. Their evaluation is based on ten core compliance-monitoring functionalities, including modelling the compliance requirements related to control flow, data, and human resources. However, the scope of their work is limited only to the functionalities of monitoring approaches, while the functionalities of design-time and post-execution-time

approaches are left out. In addition, their evaluation includes checking the expressiveness of modelling languages to model typical compliance patterns. Fenech et al. [106] evaluate the expressiveness of CTL, LTL and CSP with the deontic-based contract language CL [107] for modelling the full specifications of electronic contracts.

Our survey is different from [106] in the sense that we are evaluating the expressiveness of existing languages using more complex compliance rules but [106] only consider simple rules. In contrast, [107] evaluates the electronic contracts for only functional and behavioural requirements, which are more relevant to conformance checking, while we evaluate compliance requirements from a legal perspective.

Caron et al. [42] provide a comprehensive rule-based compliance checking and risk management with a process-mining framework. The authors also evaluate the expressiveness of rule-based languages for modelling different compliance rule patterns consisting of a rule-restriction focus (e.g., cardinality-based rules, co-existence rules) and rule patterns pertaining to various process aspects (e.g., data, resources). The pragmatic, semantic and syntactic foundations of visual modelling languages are discussed in [108]. Moreover, Otto and Antón [102], study existing compliance approaches to extracting the required information for modelling compliance requirements. Whereas the authors of COMPAS-Project [3] provide an overview of the state of the art of the compliance languages with an emphasis on languages for regulatory and legislative provisions. This work is closely related to the work of Elgammal et al. [91] but does not cover complex patterns and various process perspectives, as we do. In contrast, [109] focuses on how modelling languages are used to align the compliance requirements for business processes. In addition, the authors discuss various graphical, logic-based, mark-up languages and constraint-based languages for representing compliance rules pertaining to data, resources and temporal rules.

Turki and Bjekovic-Obradovic [110] investigates the practices of regulation analysis for extracting key information from legal sources for the information system engineering. Their work also includes the analysis of existing practices in the compliance rules domain for achieving and maintaining the compliance of e-Government services and IS pertaining regulations. However, the downside of their work is that they focus only on goal-oriented modelling approaches, e.g., SecureTropos [111] and goal-oriented requirement language (GRL) [112,113]. The SecureTropos is based on the i^* framework, and involves the extraction and modelling of various types of goals, activities and resources expressing obligations. Whereas GRL, which combines URN (*User Requirements Notation* [114]) and UCM (*Use Case Map* [115]), represents the requirements from legal sources in terms of actors, goals and tasks.

A rather similar work is reported in Otto and Antón [102], where the authors examine various approaches to regulation modelling languages and the extraction of key legal concepts from legal documents. Contrary to [102,110], we go beyond the goal-oriented languages and analyse formal and visual languages with complex requirements patterns from both spheres of the compliance problem. Furthermore, in contrast to [102], we exclude mark-up based languages from our analysis because, although such languages (e.g., REGNET-Project [116], Standard Generalized Markup Language (SGML: https://www.w3.org/TR/WD-html40-970708/intro/sgmltut.html, accessed on 20 September 2020.), OASIS LegalRuleML (http://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/csprd02/legalruleml-core-spec-v1.0-csprd02.html, accessed on 20 September 2020)) can capture the structure of the regulations, and meta-data regarding various sections of the legal documents, they do not provide enough details on the underlying representations of the compliance requirements.

The work by Elgammal et al. [91] reports a comparative analysis of modelling languages from temporal and deontic families of logics and evaluates three languages, LTL, CTL and FCL. Their comparison includes the eleven features that a modelling language should have, including *formality, expressiveness, declarativeness and non-monotonicity*, to name but a few. They also discuss the strengths and weaknesses of the evaluated languages—in particular, the expressiveness and computational complexity, and the complexity of the sources of the compliance requirements. Hashmi and Governatori [25], on the other

hand, reported a rather similar work where the authors investigated seven CMFs and evaluated their conceptual foundations to gain a better understanding of whether they provide reasoning support of legal norms. As CMFs build on weak conceptual foundations, the authors used predefined evaluation criteria and obligation modalities [72] to check the one-to-one mappings between the types of obligations and modelling patterns provided by the language used in a CMF. Although these works are similar to the one presented here, they are different in the sense that Elgammal et al.'s work does not include graph-based languages in their analysis such as DMQL or BPMN-Q. It, thus, has a limited scope. BPMN-Q is a CTL-based (CTL is a superset of LTL; hence, the characteristics of LTL can be generalized to CTL. See Vardi [117] for details on the pros and cons of linear and branching time logics, though such discussions are not related to their ability to represent legal norms.) query language; hence, their comparative results can be generalized to BPMN-Q only—but not to DMQL, which is based on graph theory. In contrast, Hashmi et al.'s [72] evaluation is close to ours but has limited scope, as we also include graph-based, primitive-based and visual modelling languages, e.g., DMQL and eCRG, on top of logic-based modelling languages. In addition, their evaluations do not consider compliance requirements related to data, or functional and structural process information.

Apart from these focused evaluations, other surveys accumulating understanding of compliance management from a variety of perspectives have been mentioned in the literature. For example, Becker et al. [23] present a literature study based on the generalisability and applicability of business-process compliance frameworks, and only cover the aspect of the implementation results of the surveyed frameworks. Fellmann and Zasada [24] survey the dominating trends and issues in business-process compliance through four dimensions. These include variables in general business-process modelling (e.g., information, location, resources), temporal aspects and a distinction between the approaches based on the formality, that is, whether the approach is a verification or a validation approach. Hashmi et al. [118], on the other hand, accumulate a holistic view of the state-of-affairs in the compliance domain. They review the literature across 13 dimensions including the expressiveness of formal language for representing and reasoning about legal norms, rules pertaining to process aspects (i.e., data, time, resource etc.), and highlight intriguing issues faced by the domain. In contrast, Sackmann et al. [119] examine BPC approaches in relation to their suitability to support the compliance management (CM) life-cycle phases (proposed in [120]) for their interoperability with business processes in the context of the digitalisation of the compliance function. Casanovas et al. [121], on the other hand, investigate the existing literature and research projects into *compliance by design* (CbD) and clarify the double process of converging trends in the legal theories, legal technologies including AI, and spreading process-management approaches to other domains. Moreover, the authors discuss the relationships and differences between various domains and proposals. Based on their analysis, the authors classify CbD into business-, legal- and regulatory-driven compliance problems. Their analysis is somewhat similar to the work of [24,118], but their preliminary focus is on differentiating the concept of CbD to achieving *compliance through design* (CtD). The authors also implicitly discuss the expressiveness of various compliance modelling languages. However, no formal expressiveness and complexity analysis is conducted, and, thus, they have a different orientation. Goedertier et al. [122], on the other hand, evaluate different approaches, principles to declarative process modelling ranging from imperative models to representing declarative modelling approaches. Mostly, the evaluated approaches differ from the business concern, state space and the constraints types they are able to model, and the modelling and reasoning framework they use—yet their objective remains the same. In addition, this work is limited in scope as it focuses only on declarative process-modelling paradigms and does not consider the compliance concerns, as is achieved in [56].

8. Discussion

This study was conducted to compare and evaluate compliance rules languages in two ways: semantical expressiveness and textual complexity. Due to the quite heterogeneous pool of language artefacts, we chose to organize our approach into three phases. In *Phase 1*, we selected twelve languages during a systematic literature review and classified them according to their different conceptual foundations. In *Phase 2*, we modelled a sample of compliance requirements with a subset of seven languages. In *Phase 3*, we evaluated the formally specified rules with regard to their expressiveness and complexity. In the remainder of this section, we focussed on the formalisation aspect, i.e., the coverage of process perspectives and deontic effects as well as the languages' textual complexity.

8.1. Summary of Results

As the formalisation showed, all languages feature the decomposition of a requirement into singular constraints, and provide a mapping to formal conditionals addressing one or more compliance concerns in either textual or visual form, or a combination of both. Our investigation reveals that most approaches facilitate the modelling of control-flow and data-flow requirements that are an integral part of every process description. However, compliance rules can also be assessed by their normative meanings, which go beyond the formal representation of process elements and the functional relation between them.

In this regard, the analysis helped diversify the results and to pinpoint languages that are more sophisticated in displaying the different normative concepts, when evaluated on the given framework. Interestingly, not only a language designed to capture such premises such as PCL, but also eCRG as a visual language, is able to differentiate themselves from the other languages. Although both can be seen as fairly expressive compliance languages, eCRG copes with the complexity and redundancy imposed by its many constructs. This aspect becomes even more interesting considering the effect on users in terms of time and effort, which are likely to influence the modelling performance. PENELOPE, however, did not achieve its full potential in a legal-reasoning context due to its narrowed focus on compliance modelling. Note that the user perspective, which clearly plays an important role in developing a holistic view of every modelling language, is linked to this research, but not considered explicitly.

Coming from the generic classification used in Table 1, there is no obvious trend concerning the pattern-based approaches such as CRL, or the hybrid approaches Declare and BPMN-Q, which combine visual patterns with a formal representation in temporal logics. Remarkably, all pattern-based languages convince by offering a manageable size of language elements and the ease of use of predefined mappings, which is also clear from the complexity discussion. As for the deontic modalities, the main concepts can be modelled with some semantic losses concerning permissions, and (non)preemptive as well as perdurant obligations. Similar to the user perspective, the implications of bypassing or substituting the intended normative meaning are beyond the scope of this research. Based on our evaluation, we assume that certain constructs from the control-flow or resource perspective express the intuitive meaning of the compliance requirements correctly, even if they deviate from the legal theory. This has been demonstrated for resource and composite patterns. From the sample of pattern approaches, Declare is the one language that turns out to be less expressive in terms of the other process perspectives, but more capable of adopting the varied normative concepts.

For query-based languages such as DMQL (and to some extent for BPMN-Q), the rule formalisation showed that it is possible to implement compliance rules in a systematic and appealing way. Especially for simple order and occurrence as well as time-related compliance requirements, the construction of a DMQL query for checking BPMN models is rather straight-forward. In contrast, data flow and compensation-related compliance rules are much harder to capture. Most of the issues occur due to the interpretation of abstract compliance rules in terms of constructs of a concrete modelling language such as BPMN. In some cases, translating an abstract compliance rule means creating new process

knowledge and describing it in terms of a query pattern. To do so, many design decisions have to be made (e.g., inserting timer events and intermediate events). Finally, since DMQL is a query language intended for retrieving the specified patterns, enforcing compliance rules on process models would mean searching for violations or the absence of compliant behaviour. In this regard, the patterns have to be negated.

8.2. Implications for Research and Practice

Implications of this work pertain to both research and practice. First, from a research perspective, the survey reveals a plethora of approaches but no common evaluation basis. There is no comparable study to ours that goes beyond the conceptual investigation of a language's constructs evaluating their expressiveness on the one hand and normative implications, on the other. The approach also facilitates a more conscious use of different languages such as graphs, patterns, queries or logics. Moreover, we were able to provide new insights into the complexity of compliance languages. By combining these two essential evaluation tools, we devise a systematic procedure to evaluate compliance languages, and invite researchers to rethink their design decisions to be more user-aware. *In this direction, we encourage consideration of the trade off between expressiveness, complexity and usability*. Our approach to measuring complexity is scalable to a large number of compliance rules and transferable to other compliance-specific, or non-specific process modelling languages.

Second, its relevance for practice is shown by the applicability to a real business process. We argue that apart from the financial domain, the approach is sufficiently flexible to be easily adapted to many other domains. In addition, the deontic notion of obligations adds to the correct understanding of regulations by decomposing a requirement into the most important normative principles, and thereby helps to overcome weaknesses such as under-/over-specification. In addition, in general, less complex languages that are easy to write and read save time and money. A measure for evaluating modelling languages in this regard, and even a language-independent one, is, hence, a desirable feature. Moreover, metrics for the compliance domain have not been exploited so far and have huge potential to estimate the compliance effort and identify problem areas arising from complex process structures. More precisely, accessing and modulating frequently occurring structures in compliance rules might be excellent levers to decrease the overall complexity.

8.3. Limitations

As with every research, there are limitations to be taken into account in order to present the findings in the proper proportions. We seek to reflect on the results in, essentially, four areas, that related to the literature review, the business process and compliance rules, the formalisation itself as well as the two evaluation criteria: expressiveness and complexity.

All in all, seven languages were included in this evaluation, which means that five languages resulting from the literature review had to be dismissed out of practical reasons. Thereby, some languages and pattern systems relevant to the BPC area remain open to further research (cf. Table 1). The approach to the language selection is restricted by the applied filtering criteria which, by all exercised care might occasionally have lead to the exclusion of suitable approaches (cf. Figure 2). Finally, using only one scholarly database, no alternative queries and only the title search, might have limited the pool of languages, although this does not conflict with the general qualitative notion of this research. In addition, future empirical studies could shed light on how practitioners used the languages that we compared in our work or which other instruments they use to solve compliance problems, i.e., include sources from outside academia.

The running example is linked to a process of the financial domain, and, hence, is driven by domain semantics and restricted by the number of compliance rules. The evaluation was carried out using real-life compliance requirements. Each of them represents a different process perspective and carries another deontic modality, which for some requirements yielded more than one possible normative meaning. The remaining deontic effects not covered by the sample were mapped to language constructs suitable to support

the normative concepts. Thus, the practical application of the compliance languages could only be carried out for those concepts included in the sample. As the formalisation shows, not all critical process perspectives were modelled exactly as indicated by the classification in Table 2. In fact, some process perspectives (i.e., data, time) had to be interpreted more intuitively.

Furthermore, the adopted metrics are fundamentally designed to check the complexity of software languages, not formal languages. Each software design language may have different semantics, syntax and pragmatics. Since formal languages have different semantics, as well as patterns, they might not necessarily be fully analysed with software complexity analysis tools such as Halstead's metrics. The main problem, however, arises from the individual classification of operators and operands. In this regard, we tried to unwrap variables, constants and structuring elements as best as possible, and created a glossary (cf. Appendix A) to keep record of all definitions.

9. Conclusions

Requirements modelling is an important part of the overall compliance problem. In particular, the gap between legal formulations and process description requires extensive interpretation, whether essential compliance information has been mapped, what consequences a compliance violation may have, or which remedies exist. Modelling languages should, therefore, guide the modelling process and, above all, be expressive enough to capture the different concepts, but not so complex that modelling becomes overly demanding. Our research was driven by an interest in the extent to which existing languages control this element.

In a nutshell, our evaluation provides a holistic view of the state of affairs in the compliance-requirements modelling domain, accumulates a detailed understanding and highlights the strengths and weaknesses of the existing modelling languages—especially, from the perspective of their expressive power and complexity. In addition, it underpins what the challenges are that need to be investigated in order to provide full reasoning and representation support for the modelling of compliance requirements.

Future research calls for user studies on the usability of compliance languages and the comprehension of formal representations. Experimental investigations in this area are, so far, targeting well-established visual notations such as BPMN, while compliance languages lack user evaluation of any kind. Surprisingly, many approaches make use, for this purpose, of formal patterns without questioning their relevance to the process or how users would be able to handle them. Here, the question of how efficiently users apply these languages in terms of performance and time poses the main challenge. With this additional information, it should be possible to either improve existing tools or invent new solutions to then well-known problems.

Author Contributions: Conceptualization, A.Z. and M.H.; methodology, A.Z. and M.H.; formal analysis, A.Z., M.H., M.F. and D.K.; resources, A.Z., M.H., M.F. and D.K.; writing—original draft preparation, A.Z., M.H., M.F. and D.K.; writing—review and editing, A.Z., M.H., M.F. and D.K.; funding acquisition, All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Complexity Measures for Evaluated Languages

Table A1. Definitions on BPMN-Q.

ID	n_1	n_2
R1	{pattern, sequence flow}	{activity}
R2	{data flow}	{activity, data}
R3	{pattern, sequence flow, data flow}	{activity, data}
R4	{pattern, sequence flow, data flow}	{activity, data}
R5	n.a.	n.a.
R6	n.a.	n.a.
R7	{data flow}	{activity, data}
R8	{pattern, start event, sequence flow}	{activity}
R9	{pattern, sequence flow, data flow}	{activity, data}

Table A2. Metrics calculations for BPMN-Q.

ID	n_1	n_2	n	N_1	N_2	N	V	D	E
R1	2	1	3	2	2	4	6.34	2.00	12.68
R2	1	2	3	2	3	5	7.92	0.75	5.94
R3	3	2	5	3	3	6	13.93	2.25	31.35
R4	3	2	5	4	4	8	18.58	3.00	55.73
R5	.	.	.	.	.	.	.	.	.
R6	.	.	.	.	.	.	.	.	.
R7	1	2	3	1	2	3	4.75	0.50	2.38
R8	3	1	4	3	1	4	8.00	1.50	12.00
R9	3	2	5	9	9	18	41.79	6.75	282.11

Table A3. Definitions on CRL.

ID	n_1	n_2
R1	{pattern}	{activity}
R2	{pattern, and, parenthesis}	{activity}
R3	{pattern}	{activity, data}
R4	{pattern, and, parenthesis}	{activity}
R5	{pattern}	{activity, data, resource}
R6	{pattern, and, parenthesis}	{activity}
R7	{pattern}	{activity, data}
R8	{pattern}	{activity}
R9	{pattern, else, parenthesis}	{activity, data}

Table A4. Metrics calculations for CRL.

ID	n_1	n_2	n	N_1	N_2	N	V	D	E
R1	1	1	2	1	2	3	3.00	1.00	3.00
R2	3	1	4	3	2	5	10.00	3.00	30.00
R3	1	2	3	1	2	3	4.75	0.50	2.38
R4	3	1	4	3	3	6	12.00	4.50	54.00
R5	1	3	4	1	3	4	8.00	0.50	4.00
R6	3	1	4	3	3	6	12.00	4.50	54.00
R7	1	2	3	1	3	4	6.34	0.75	4.75
R8	1	1	2	1	2	3	3.00	1.00	3.00
R9	3	2	5	6	6	12	27.86	4.50	125.38

Table A5. Definitions on DECLARE.

ID	n_1	n_2
R1	{Precendence}	{activity type}
R2	{Existence}	{activity type, multiplicity label}
R3	{Response}	{activity type}
R4	{Response}	{activity type}
R5	n.a.	n.a.
R6	n.a.	n.a.
R7	{Precedence}	{activity type}
R8	{Responded_Existence}	{activity type}
R9	{Branched_Response,Response,Choice}	{activity type, choice label}

Table A6. Metrics calculations for DECLARE.

ID	n_1	n_2	n	N_1	N_2	N	V	D	E
R1	1	1	2	1	2	3	3.00	1.00	3.00
R2	1	2	3	1	2	3	4.75	0.50	2.38
R3	1	1	2	1	2	3	3.00	1.00	3.00
R4	1	1	2	1	2	3	3.00	1.00	3.00
R5	.	.	.	.	.	.	.	.	.
R6	.	.	.	.	.	.	.	.	.
R7	1	1	2	1	2	3	3.00	1.00	3.00
R8	1	1	2	1	2	3	3.00	1.00	3.00
R9	3	2	5	3	6	9	20.90	4.50	94.04

Table A7. Definitions on DMQL.

ID	n_1	n_2
R1	{arc path}	{node event, node task}
R2	{arc non-directed, arc original/opposite direction}	{node lane, node task, node data}
R3	{arc non-directed, arc original/opposite direction, attribute relation}	{node lane, node task, node data, attribute}
R4	{arc path, arc non-directed, arc original/opposite direction}	{node lane, node send, node receive, node task}
R5	{arc non-directed}	{node lane, node task, attribute}
R6	{arc non-directed, arc original/opposite direction}	{node lane, node task, node data}
R7	{arc non-directed, arc original/opposite direction}	{node lane, node task, node event}
R8	{arc non-directed, arc path non-directed, arc original/opposite direction}	{node lane, node various, node task, node data}
R9	{node event, node send, node timer, node, node task}	{node event, node send, node timer, node, node task}

Table A8. Metrics calculations for DMQL.

ID	n_1	n_2	n	N_1	N_2	N	V	D	E
R1	1	2	3	1	2	3	4.75	0.50	2.38
R2	2	3	5	3	4	7	16.25	1.33	21.67
R3	3	4	7	3	4	7	19.65	1.50	29.48
R4	3	4	7	6	6	12	33.69	2.25	75.80
R5	1	3	4	2	3	5	10.00	0.50	5.00
R6	2	3	5	4	6	10	23.22	2.00	46.44
R7	2	3	5	2	3	5	11.61	1.00	11.61
R8	3	4	7	3	4	7	19.65	1.50	29.48
R9	3	5	8	10	11	21	63.00	3.30	207.90

Table A9. Definitions on PCL.

ID	n_1	n_2
R1	{operatorType, deontic modalityType}	{(activity)literal}
R2	{operatorType, deontic modalityType, operatorType}	{literal}
R3	{operatorType, deontic modalityType}	{literal, (activity)literal}
R4	{operatorType, deontic modalityType, operatorType}	{literal, (activity)literal}
R5	{operatorType, deontic modalityType, operatorType}	{literal, literal, literal, OR}
R6(1)	{operatorType, deontic modalityType, operatorType}	{literal, literal}
R6(2)	{operatorType, deontic modalityType, operatorType}	{literal, (resource)literal}
R7	{operatorType, deontic modalityType, operatorType}	{literal, literal}
R8	{operatorType, deontic modalityType, operatorType}	{literal, (resource)literal}
R9(1)	{operatorType, deontic modalityType, operatorType}	{literal}
R9(2)	{operatorType, deontic modalityType, operatorType}	{literal, (activity)literal}

Table A10. Metrics calculations for PCL.

ID	n_1	n_2	n	N_1	N_2	N	V	D	E
R1	2	1	3	3	1	4	6.34	1.00	6.34
R2	3	1	4	3	1	4	8.00	1.50	12.00
R3	2	2	4	2	3	5	10.00	1.50	15.00
R4	3	3	6	3	2	5	12.92	1.00	12.92
R5	3	2	5	3	3	6	13.93	2.25	31.35
R6(1)	3	2	5	3	2	5	11.61	1.50	17.41
R6(2)	3	2	5	3	2	5	11.61	1.50	17.41
R7	3	2	5	3	2	5	11.61	1.50	17.41
R8	3	2	5	3	2	5	11.61	1.50	17.41
R9(1)	3	1	4	3	1	4	8.00	1.50	12.00
R9(2)	3	2	5	3	2	5	11.61	1.50	17.41

Table A11. Definitions on PENELOPE.

ID	n_1	n_2
R1	{event, activity, pattern,logical operator,temporal operator, fluent, agent}	{pattern type, event, fluent, time, AND, LEQ}
R2	{event, pattern, logical operator, temporal operator, fluent, agent}	{pattern type, event, fluent, time, AND, LEQ}
R3	{event, pattern, fluent, agent, temporal operator, logical operator}	{pattern type, resource, fluent, time, AND, LEQ}
R4	{pattern, activity, agent, fluent,temporal operator, logical operator}	{pattern type, AND, LEQ, time, resource}
R5	{pattern, event, agent, temporal operator, logical operator}	{pattern type, fluent, event, advisor, AND, OR, LEQ, time}
R6	n.a.	n.a.
R7	{pattern, event, agent, fluent, temporal operator, logical operator}	{pattern type, activity, advisor, AND, LEQ, time}
R8	n.a.	n.a.
R9	{pattern, agent, fluent, activity, temporal operator,logical operator}	{pattern type, activity, fluent, AND, LEQ, time}

Table A12. Metrics calculations for PENELOPE.

ID	n_1	n_2	n	N_1	N_2	N	V	D	E
R1	7	6	13	10	5	15	55.51	2.92	161.89
R2	6	6	12	9	5	14	50.19	2.50	125.47
R3	6	6	12	10	4	14	50.19	2.00	100.38
R4	6	5	11	79	13	22	76.11	7.80	593.64
R5	5	8	13	8	6	14	51.81	1.88	97.14
R6	.	.	.	.	.	.	.	.	.
R7	6	6	12	9	10	19	68.11	5.00	340.57
R8	.	.	.	.	.	.	.	.	.
R9	6	6	12	12	11	23	82.45	5.50	453.50

Table A13. Definitions on eCRG.

ID	n_1	n_2
R1	{activity AO, message CO, sequence flow C}	{organization, message type, activity type}
R2	{activity CO, data flow CO, performing relation CO}	{data, resource, activity type}
R3	{activity AO, activity CO, data object CO, data flow A, data flow C, performing relation C, sequence flow C}	{data, resource, activity type}
R4	{activity AO, message CO, data object C, data flow C, sequence flow C}	{data, organization, message type, activity type}
R5	{activity AO, resource A, condition C, performing relation A, resource relation C}	{resource, condition label, relation label, activity type}
R6	{activity AO, resource relation C}	{resource, relation label, activity type}
R7	{activity AO, activity CO, sequence flow C, performing relation C}	{resource, activity type}
R8	{activity AO, activity CO}	{activity type}
R9	{activity AO, activity CO, message AO, message CA, data object AO, condition AO, condition CO, sequence flow AO, sequence flow CO, data flow AO, dataflow CO, performing relation, xor}	{data, condition label, organization, unit, message type, activity type}

Table A14. Metrics calculations for eCRG.

ID	n_1	n_2	n	N_1	N_2	N	V	D	E
R1	3	3	6	3	3	6	15.51	1.50	23.26
R2	3	3	6	4	4	8	20.68	2.00	41.36
R3	7	3	10	8	6	14	46.51	7.00	325.55
R4	5	3	8	9	5	14	42.00	4.17	175.00
R5	5	4	9	5	4	9	28.53	2.50	71.32
R6	2	2	4	4	4	8	16.00	2.00	32.00
R7	4	2	6	4	3	7	18.09	3.00	54.28
R8	2	1	3	2	2	4	6.34	2.00	12.68
R9	14	8	22	28	18	46	205.13	15.75	3230.86

References

1. *SOX, Sarbanes-Oxley Act of 30 July 2002, 15 USC 7201 Note, Public Law 107-204, 107th Congress, 116 Statistics Act, Section 404*; Technical Report; 2002. Available online: https://www.govinfo.gov/app/details/PLAW-107publ204 (accessed on 2 August 2022).
2. Leone, A.J. Factors related to internal control disclosure: A discussion of Ashbaugh, Collins, and Kinney (2007) and Doyle, Ge, and McVay (2007). *J. Account. Econ.* **2007**, *44*, 224–237. [CrossRef]
3. COMPAS-Project. *D2.1 State-of-the-Art in the Field of Compliance Languages—Compliance-Driven Models, Languages, and Architectures for Services*; Report D2.1; Tilburg University: Tilburg, The Netherlands, 2008.
4. Bartolini, C.; Giurgiu, A.; Lenzini, G.; Robaldo, L. Towards Legal Compliance by Correlating Standards and Laws with a Semi-automated Methodology. In Proceedings of the BNAIC 2016: Artificial Intelligence—28th Benelux Conference on Artificial Intelligence, Amsterdam, The Netherlands, 10–11 November 2016; Bosse, T., Bredeweg, B., Eds.; Revised Selected Papers; Communications in Computer and Information Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 765, pp. 47–62. [CrossRef]
5. Sadiq, S.; Governatori, G.; Namiri, K. Modeling control objectives for business process compliance. In Proceedings of the International Conference on Business Process Management (BPM'07), Brisbane, Australia, 24–28 September 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 149–164.
6. Reichert, M.; Weber, B. Business Process Compliance. In *Enabling Flexibility in Process-Aware Information Systems*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 297–320.
7. Liu, Y.; Muller, S.; Xu, K. A static compliance-checking framework for business process models. *IBM Syst. J.* **2007**, *46*, 335–361. [CrossRef]
8. Häußler, M.; Esser, S.; Borrmann, A. Code compliance checking of railway designs by integrating BIM, BPMN and DMN. *Autom. Constr.* **2021**, *121*, 103427. [CrossRef]

9. Awad, A.; Decker, G.; Weske, M. Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In Proceedings of the International Conference on Business Process Management (BPM'08), Milan, Italy, 2–4 September 2008; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5240, pp. 326–341.
10. Ghanavati, S.; Amyot, D.; Rifaut, A. Legal Goal-Oriented Requirement Language (Legal GRL) for Modeling Regulations. In Proceedings of the 6th International Workshop on Modeling in Software Engineering (MiSE 2014), Hyderabad, India, 31 May–7 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1–6. [CrossRef]
11. Barnawi, A.; Awad, A.; Elgammal, A.; El Shawi, R.; Almalaise, A.; Sakr, S. Runtime Self-Monitoring Approach of Business Process Compliance in Cloud Environments. *Clust. Comput.* **2015**, *18*, 1503–1526. [CrossRef]
12. Castellanos-Ardila, J.P.; Gallina, B.; Governatori, G. Compliance-aware engineering process plans: The case of space software engineering processes. *Artif. Intell. Law* **2021**, *29*, 587–627. [CrossRef]
13. Ghanavati, S.; Hulstijn, J. Impact of Legal Interpretation on Business Process Compliance. In Proceedings of the 2015 IEEE/ACM 1st International Workshop on TEchnical and LEgal aspects of data pRivacy and SEcurity (TELERISE '15), Florence, Italy, 18 May 2015; pp. 26–31.
14. De Masellis, R.; Maggi, F.M.; Montali, M. Monitoring Data-Aware Business Constraints with Finite State Automata. In Proceedings of the 2014 International Conference on Software and System Process (ICSSP 2014), Nanjing, China, 26–28 May 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 134–143. [CrossRef]
15. Corea, C.; Delfmann, P. Detecting Compliance with Business Rules in Ontology-Based Process Modeling. In Proceedings of the Towards Thought Leadership in Digital Transformation: 13. Internationale Tagung Wirtschaftsinformatik, WI 2017, St.Gallen, Switzerland, 12–15 February 2017; Leimeister, J.M., Brenner, W., Eds.; 2017.
16. Gong, P.; Knuplesch, D.; Feng, Z.; Jiang, J. bpCMon: A Rule-Based Monitoring Framework for Business Processes Compliance. *Int. J. Web Serv. Res. (IJWSR)* **2017**, *14*, 23. [CrossRef]
17. Loreti, D.; Chesani, F.; Ciampolini, A.; Mello, P. Distributed Compliance Monitoring of Business Processes over MapReduce Architectures. In Proceedings of the ICPE '17 Companion; Association for Computing Machinery: New York, NY, USA, 2017; pp. 79–84. [CrossRef]
18. Tosatto, S.C.; Governatori, G. Computational Complexity of Compliance and Conformance: Drawing a Line Between Theory and Practice. *J. Appl. Logics—IfCoLog J. Logics Their Appl.* **2021**, *8*, 1023–1064 .
19. Oyekola, O.; Xu, L. Verification and compliance in collaborative processes. In *Proceedings of the Working Conference on Virtual Enterprises*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 213–223.
20. Hamdani, R.E.; Mustapha, M.; Amariles, D.R.; Troussel, A.; Meeùs, S.; Krasnashchok, K. A combined rule-based and machine learning approach for automated GDPR compliance checking. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law, São Paulo, Brazil, 21–25 June 2021; pp. 40–49.
21. Cunha, V.H.C.; Caiado, R.G.G.; Corseuil, E.T.; Neves, H.F.; Bacoccoli, L. Automated compliance checking in the context of Industry 4.0: From a systematic review to an empirical fuzzy multi-criteria approach. *Soft Comput.* **2021**, *25*, 6055–6074.
22. Krasnashchok, K.; Mustapha, M.; Al Bassit, A.; Skhiri, S. Towards privacy policy conceptual modeling. In *Proceedings of the International Conference on Conceptual Modeling*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 429–438.
23. Becker, J.; Delfmann, P.; Eggert, M.; Schwittay, S. Generalizability and applicability of model-based business process compliance—Checking approaches: A state-of-the-art analysis and research roadmap. *Bus. Res.* **2012**, *5*, 221–247.
24. Fellmann, M.; Zasada, A. State-of-the-Art of Business Process Compliance Approaches: A Survey. In Proceedings of the 22st European Conference on Information Systems, ECIS 2014, Tel Aviv, Israel, 9–11 June 2014.
25. Hashmi, M.; Governatori, G. Norms modeling constructs of business process compliance management frameworks: A conceptual evaluation. *Artif. Intell. Law* **2018**, 26, 251–305. [CrossRef]
26. Knuplesch, D.; Reichert, M.; Kumar, A. A framework for visually monitoring business process compliance. *Inf. Syst.* **2017**, *64*, 381–409. [CrossRef]
27. Ly, L.T.; Maggi, F.M.; Montali, M.; Rinderle-Ma, S.; van der Aalst, W.M. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Inf. Syst.* **2015**, *54*, 209–234. [CrossRef] [PubMed]
28. Van Der Aalst, W.M.; Pesic, M. DecSerFlow: Towards a truly declarative service flow language. In Proceedings of the 4th International Workshop on Web Services and Formal Methods (WS-FM'06), Brisbane, Australia, 28–29 September 2007; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–23.
29. Garey, M.R.; Johnson, D.S. Computers and intractability: A guide to the theory of npcompleteness (series of books in the mathematical sciences), ed. In *Computers and Intractability*; Freeman: San Francisco, CA, USA, 1979; Volume 340.
30. Webster, J.; Watson, R.T. Analyzing the past to prepare for the future: Writing a literature review. *MIS Q.* **2002**, *26*, xiii–xxiii.
31. Kitchenham, B.; Brereton, O.P.; Budgen, D.; Turner, M.; Bailey, J.; Linkman, S. Systematic literature reviews in software engineering–a systematic literature review. *Inf. Softw. Technol.* **2009**, *51*, 7–15.
32. Awad, A.; Weidlich, M.; Weske, M. Visually Specifying Compliance Rules and Explaining their Violations for Business Processes. *J. Vis. Lang. Comput.* **2011**, *22*, 30–55. [CrossRef]
33. Elgammal, A.; Turetken, O.; van den Heuvel, W.J.; Papazoglou, M. Formalizing and appling compliance patterns for business process compliance. *Softw. Syst. Model.* **2016**, *15*, 119–146.
34. Letia, I.A.; Goron, A. Model checking as support for inspecting compliance to rules in flexible processes. *J. Vis. Lang. Comput.* **2015**, *28*, 100–121. [CrossRef]

35. Delfmann, P.; Breuker, D.; Matzner, M.; Becker, J. Supporting Information Systems Analysis Through Conceptual Model Query–The Diagramed Model Query Language (DMQL). *Commun. Assoc. Inf. Syst.* **2015**, *37*, 24. [CrossRef]
36. Knuplesch, D.; Reichert, M. A visual language for modeling multiple perspectives of business process compliance rules. *Softw. Syst. Model.* **2017**, *16*, 715–736.
37. Governatori, G.; Rotolo, A. A Conceptually Rich Model of Business Process Compliance. In Proceedings of the 7th Asia-Pacific Conference on Conceptual Modelling (APCCM'10), Brisbane, Australia, 18–21 January 2010; Volume 110, pp. 3–12.
38. Ramezani, E.; Fahland, D.; Van Dongen, B.; Van Der Aalst, W. *Diagnostic Information in Temporal Compliance Checking*; Technical Report; BPM Center: San Pedro, Costa Rica, 2012.
39. Goedertier, S.; Vanthienen, J. Designing Compliant Business Processes with Obligations and Permissions. In Proceedings of the 4th Business Process Management Workshops: BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, 4–7 September 2006; Eder, J., Dustdar, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 5–14. [CrossRef]
40. Yu, J.; Manh, T.P.; Han, J.; Jin, Y.; Han, Y.; Wang, J. Pattern Based Property Specification and Verification for Service Composition. In Proceedings of 7th International Conference in Web Information Systems Engineering (WISE'06); Aberer, K., Peng, Z., Rundensteiner, E.A., Zhang, Y., Li, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 156–168.
41. Forster, A.; Engels, G.; Schattkowsky, T.; Van Der Straeten, R. Verification of business process quality constraints based on visual process patterns. In Proceedings of the 1st Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE'07), Shanghai, China, 6–8 June 2007; pp. 197–208.
42. Caron, F.; Vanthienen, J.; Baesens, B. Comprehensive rule-based compliance checking and risk management with process mining. *Decis. Support Syst.* **2013**, *54*, 1357–1369.
43. Becker, J.; Delfmann, P.; Dietrich, H.A.; Steinhorst, M.; Eggert, M. Business process compliance checking–applying and evaluating a generic pattern matching approach for conceptual models in the financial sector. *Inf. Syst. Front.* **2016**, *18*, 359–405.
44. Ramezani, E.; Fahland, D.; van der Aalst, W.M. Where did I misbehave? diagnostic information in compliance checking. In Proceedings of the 10th International Conference on Business Process Management (BPM'12), Tallinn, Estonia, 3–6 September 2012; Springer: Berlin/Heidelberg, Germany; pp. 262–278.
45. Timm, F.; Zasada, A.; Thiede, F. Building a Reference Model for Anti-Money Laundering in the Financial Sector. In Proceedings of the 18th Conference on Learning, Knowledge, Data, Analytics (LWDA'16), Potsdam, Germany, 12 September 2016.
46. Zasada, A.; Bui, T. More than meets the eye: A Case Study on the Role of IT Affordances in Supporting Compliance. In Proceedings of the 24th Americas Conference on Information Systems (AMCIS'18), New Orleans, LA, USA, 16–18 August 2018.
47. Bank for International Settlements. *A Global Regulatory Framework for More Resilient Banks and Banking Systems*; Technical Report; Bank for International Settlements: Basel, Switzerland, 2011.
48. Awad, A.; Barnawi, A.; Elgammal, A.; Elshawi, R.; Almalaise, A.; Sakr, S. Runtime detection of business process compliance violations: An approach based on anti patterns. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, (SAC '15), Salamanca, Spain, 13–17 April 2015; pp. 1203–1210.
49. Dwyer, M.B.; Avrunin, G.S.; Corbett, J.C. Patterns in property specifications for finite-state verification. In Proceedings of the 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, USA, 16–22 May 1999; pp. 411–420.
50. Elgammal, A.; Turetken, O.; van den Heuvel, W.J.; Papazoglou, M. Root-cause analysis of design-time compliance violations on the basis of property patterns. In Proceedings of the International Conference on Service-Oriented Computing (ICSOC'10), Perth, Australia, 13–15 December 2010; pp. 17–31.
51. Turetken, O.; Elgammal, A.; van den Heuvel, W.J.; Papazoglou, M.P. Enforcing compliance on business processes through the use of patterns. In Proceedings of the 19th European Conference on Information Systems, (ECIS'11), Helsinki, Finland, 9–11 June 2011; p. 5.
52. Turetken, O.; Elgammal, A.; van den Heuvel, W.J.; Papazoglou, M.P. Capturing compliance requirements: A pattern-based approach. *IEEE Softw.* **2012**, *29*, 28–36.
53. Elgammal, A.; Turetken, O.; Van Den Heuvel, W.J. Using patterns for the analysis and resolution of compliance violations. *Int. J. Coop. Inf. Syst.* **2012**, *21*, 31–54. [CrossRef]
54. Pnueli, A. The temporal logic of programs. In Proceedings of the 18th Annual Symposium on Foundations of Computer Science, Washington, DC, USA, 30 September–31 October 1977; pp. 46–57.
55. Knuplesch, D.; Reichert, M.; Ly, L.T.; Kumar, A.; Rinderle-Ma, S. Visual modeling of business process compliance rules with the support of multiple perspectives. In Proceedings of the 32nd International Conference on Conceptual Modeling (ER'13), Hong-Kong, China, 11–13 November 2013; Lecure Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8217, pp. 106–120.
56. Pesic, M.; Schonenberg, H.; van der Aalst, W. DECLARE: Full Support for Loosely-Structured Processes. In Proceedings of the 11th IEEE International Conference on Enterprise Distributed Object Computing (EDOC'07), Annapolis, MD, USA, 15–19 October 2007; p. 287.
57. Pesic, M. Constraint-Based Workflow Management Systems: Shifting Control to Users. Ph.D. Thesis, Eindhoven University of Technology: Eindhoven, The Netherlands, 2008.
58. Montali, M.; Maggi, F.M.; Chesani, F.; Mello, P.; Aalst, W.M.P.v.d. Monitoring Business Constraints with the Event Calculus. *ACM Trans. Intell. Syst. Technol.* **2014**, *5*, 1–30. [CrossRef]

59. Ly, L.T.; Rinderle-Ma, S.; Dadam, P. Design and Verification of Instantiable Compliance Rule Graphs in process-aware Information Systems. In Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10), Hammamet, Tunisia, 7–11 June 2010; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6051, pp. 9–23.
60. Ly, L.T. SeaFlows—A Compliance Checking Framework for Supporting the Process Lifecycle. Ph.D. Thesis, University of Ulm, Ulm, Germany, 2013.
61. Delfmann, P.; Steinhorst, M.; Dietrich, H.A.; Becker, J. The generic model query language GMQL – Conceptual specification, implementation, and runtime evaluation. *Inf. Syst.* **2015**, *47*, 129–177. [CrossRef]
62. Antoniou, G.; Billington, D.; Governatori, G.; Maher, M.J. Representation results for defeasible logic. *ACM Trans. Comput. Log. (TOCL'01)* **2001**, *2*, 255–287. [CrossRef]
63. Antoniou, G.; Billington, D.; Governatori, G.; Maher, M.J. Embedding Defeasible Logic into Logic Programming. *Theory Pract. Log. Program.* **2006**, *6*, 703–735. [CrossRef]
64. Governatori, G.; Rotolo, A. Logic of Violations: A Gentzen System for Reasoning with Contrary-To-Duty Obligation. *Australas. J. Log.* **2006**, *4*, 193–215. [CrossRef]
65. Lam, H.P.; Hashmi, M.; Scofield, B. Enabling Reasoning with LegalRuleML. In Proceedings of the 10th International Symposium on the Web: Research and Applications (RuleML'16), Stony Brook, NY, USA, 6–9 July 2016; Alferes, J.J., Bertossi, L., Governatori, G., Fodor, P., Roman, D., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 241–257. [CrossRef]
66. Governatori, G. Representing business contracts in *RuleML*. *Int. J. Coop. Inf. Syst.* **2005**, *14*, 181–216. [CrossRef]
67. Hansen, J.; Pigozzi, G.; van der Torre, L.W.N. Ten Philosophical Problems in Deontic Logic. In Proceedings of the Normative Multi-Agent Systems, 19th European Summer School in Logic, Language and Information (ESSLLI 2007) Trinity College, Dublin, Ireland, 6–17 August 2007.
68. Alchourrón, C.E. Philosophical Foundations of Deontic Logic and the Logic of Defeasible Conditionals. In *Deontic Logic in Computer Science*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1994; pp. 43–84.
69. Governatori, G.; Olivieri, F.; Rotolo, A.; Scannapieco, S. Computing Strong and Weak Permissions in Defeasible Logic. *J. Philos. Log.* **2013**, *42*, 799–829. [CrossRef]
70. Stolpe, A. Relevance, Derogation and Permission. In *Proceedings of the Deontic Logic in Computer Science*; Governatori, G., Sartor, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 98–115.
71. Hashmi, M.; Governatori, G.; Wynn, M.T. Business Process Data Compliance. In Proceedings of the 6th International Symposium on Rules on the Web: Research and Applications (RuleML'12), Montpellier, France, 27–29 August 2012; pp. 32–46. [CrossRef]
72. Hashmi, M.; Governatori, G.; Wynn, M.T. Normative requirements for regulatory compliance: An abstract formal framework. *Inf. Syst. Front.* **2016**, *18*, 429–455. [CrossRef]
73. Kowalski, R.; Sergot, M. A Logic-Based Calculus of Events. In *Foundations of Knowledge Base Management*; Schmidt, J., Thanos, C., Eds.; Topics in Information Systems; Springer: Berlin/Heidelberg, Germany, 1989; pp. 23–55.
74. Miller, R.; Shanahan, M. The Event Calculus in Classical Logic—Alternative Axiomatisations. *Electron. Trans. Artif. Intell.* **1999**, *3*, 77–105.
75. Sadri, F.; Kowalski, R. Variants of the Event Calculus. In *Proceedings of the 12th International Conference on Logic Programming*; Sterling, L., Ed.; MIT: Cambrdige, MA, USA, 1995.
76. Hashmi, M.; Governatori, G.; Wynn, M.T. Modeling Obligations with Event-Calculus. In Proceedings of the 8th International Symposium on Rules on the Web: Research and Applications (RuleML'14), Prague, Czech Republic, 18–20 August 2014; pp. 296–310.
77. Kunen, K. Negation in logic programming. *J. Log. Program.* **1987**, *4*, 289–308.
78. Halstead, M.H. Elements of Software Science. *Oper. Program. Syst. Ser.* **1977**, *7*, 26 .
79. Carmo, J.; Jones, A.J. Deontic logic and contrary-to-duties. In *Handbook of Philosophical Logic*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 265–343.
80. Jones, A.J.; Sergot, M. On the characterisation of law and computer systems: The normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*; Association of Computing Machine (ACM): New York, NY, USA, 1993; pp. 275–307.
81. Piqué-Angordans, J.; Posteguillo, S.; Andreu-Besó, J.V. Epistemic and deontic modality: A linguistic indicator of disciplinary variation in academic English. *LSP Prof. Commun. (2001–2008)* **2002**, *2*, 49–65.
82. van der Torre, L.; Tan, Y.H. An update semantics for deontic reasoning. *Norms Logics Inf. Syst.* **1998**, 73–90.
83. Hilpinen, R. Deontic logic. *Blackwell Guide Philos. Log.* **2001**, *4*, 159–182.
84. Strasser, C.; Antonelli, G.A. Non-Monotonic Logic. In *The Stanford Encyclopedia of Philosophy*; Edward, N.Z., Ed.; Metaphysics Research Lab, Stanford University: Stanford, CA, USA, 2018 .
85. McCarty, L.T. Defeasible deontic reasoning. *Fundam. Informaticae* **1994**, *21*, 125–148. [CrossRef]
86. Allaire, M.; Governatori, G. On the Equivalence of Defeasible Deontic Logic and Temporal Defeasible Logic. In Proceedings of the 17th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'14), Gold Coast, QLD, Australia, 1–5 December 2015; Dam, H.K., Pitt, J., Xu, Y., Governatori, G., Ito, T., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2014; pp. 74–90.

87. Governatori, G.; Hashmi, M. No Time for Compliance. In Proceedings of the 19th IEEE International Enterprise Distributed Object Computing Conference (EDOC'15), Adelaide, Australia, 24–25 September 2015; pp. 9–18. [CrossRef]
88. Governatori, G.; Rotolo, A. A Gentzen system for reasoning with contrary-to-duty obligations: A preliminary study. In Proceedings of the 6th International Workshop on Deontic Logic in Computer Science (Deon'02), London, UK, 22–24 May 2002; pp. 97–116.
89. Governatori, G. An Abstract Normative Framework for Business Process Compliance. *Inf. Technol.* **2013**, *55*, 231–238. .
90. Governatori, G. The Regorous approach to process compliance. In Proceedings of the 19th IEEE International Enterprise Distributed Object Computing Workshop (EDOCW'15), Adelaide, SA, Australia, 22–25 September 2015; pp. 33–40.
91. Elgammal, A.; Turetken, O.; van den Heuvel, W.J.; Papazoglou, M. On the Formal Specification of Regulatory Compliance: A Comparative Analysis. In Proceedings of the 8th International Conference on Service-Oriented Computing (ICSOC'10), San Francisco, CA, USA, 7–10 December 2010; pp. 27–38.
92. Herrestad, H. Norms and Formalisation. In Proceedings of the 3rd International Conference on Artificial Intelligence and Law (ICAIL'91), Oxford, UK, 25–28 June 1991.
93. Yu, S.; Zhou, S. A survey on metric of software complexity. In Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME'10), Chengdu, China, 16–18 April 2010; pp. 352–356.
94. McCabe, T.J. A complexity measure. *IEEE Trans. Softw. Eng.* **1976**, *4*, 308–320. [CrossRef]
95. Henry, S.; Kafura, D. Software structure metrics based on information flow. *IEEE Trans. Softw. Eng.* **1981**, *5*, 510–518. [CrossRef]
96. Albrecht, A.J.; Gaffney, J.E. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Trans. Softw. Eng.* **1983**, *6*, 639–648. [CrossRef]
97. Mendling, J. Detection and Prediction of Errors in EPC Business Process Models. Ph.D. Thesis, Vienna University of Economics and Business Administrationa, Vienna, Austria, 2007.
98. Curtis, B.; Sheppard, S.B.; Milliman, P.; Borst, M.; Love, T. Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. *IEEE Trans. Softw. Eng.* **1979**, *2*, 96–104. [CrossRef]
99. Shepperd, M.; Ince, D.C. A critique of three metrics. *J. Syst. Softw.* **1994**, *26*, 197–210. [CrossRef]
100. Ferrer, J.; Chicano, F.; Alba, E. Estimating software testing complexity. *Inf. Softw. Technol.* **2013**, *55*, 2125–2139. [CrossRef]
101. Cardoso, J.; Mendling, J.; Neumann, G.; Reijers, H.A. A discourse on complexity of process models. In *Proceedings of the 4th International Conference on Business Process Management (BPM'06), Vienna, Austria*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 117–128.
102. Otto, P.N.; Antón, A.I. Addressing legal requirements in requirement engineering. In Proceedings of the 15th IEEE International Requirements Engineering Conference (RE'07), New Delhi, India, 15–19 October 2007; pp. 5–14.
103. El Kharbili, M. Business Process Regulatory Compliance Management Solution Frameworks: A Comparative Evaluation. In *Proceedings of the 8th Asia-Pacific Conference on Conceptual Modelling (APCCM'12)*; Australian Computer Society, Inc.: Wollongong, NSW, Australia, 2012; Volume 130, pp. 23–32.
104. Bonatti, P.A.; Shahmehri, N.; Duma, C.; Olmedilla, D.; Nejdl, W.; Baldoni, M.; Baroglio, C.; Martelli, A.; Coraggio, P.; Antoniou, G.; et al. *Rule-Based Policy Specification: State of the Art and Future Work, REWERSE Project Report-i2-D1*; Report; Universitá di Napoli Fedrecio II: Naples, Italy, August 2004.
105. Ly, L.T.; Maggi, F.M.; Montali, M.; Rinderle, S.; van der Aalst, W. A Framework for the Systematic Comparison and Evaluation of Compliance Monitoring Approaches. In Proceedings of the 17th IEEE International Enterprise Computing Conference (EDOC'13), Vancouver, BC, Canada, 9–13 September 2013.
106. Fenech, S.; Pace, G.J.; Okika, J.C.; Ravn, A.P.; Schneider, G. On the Specification of Full Contracts. In Proceedings of the 6th International Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA'09), York, UK, 28 March 2009.
107. Prisacariu, C.; Schneider, G. A Formal Language for Electronic Contracts. In Proceedings of the 9th International Conference on Formal Methods for Open Object-Based Distributed Systems (IFIP'07), Paphos, Cyprus, 6–8 June 2007; Springer: Berlin/Heidelberg, Germany; pp. 174–189.
108. John, T.; Kundisch, D.; Szopinski, D. Visual languages for modeling business models: A critical review and future research directions. In Proceedings of the Thirty Eighth International Conference on Information Systems (ICIS), Seoul, Repulic of Korea, 10–13 December 2017 .
109. Cabanillas, C.; Resinas, M.; Ruiz-Cortés, A. On the identification of data-related compliance problems in business processes. In Proceedings of the Jornadas Científico-Técnicas En Servicios Web Y SOA (JSWEB'10), Sevilla, Spain, 20–21 September 2010; Volume 1, pp. 89–102.
110. Turki, S.; Bjekovic-Obradovic, M. Compliance in e-Government Service Engineering: State-of-the-Art. In Proceedings of the 1st International Conference on Exploring Services Science (IESS'10), Geneva, Switzerland, 17–19 February 2010; Lecture Notes in Business Information Processing; Springer: Berlin/Heidelberg, Germany; pp. 270–275.
111. Giorgini, P.; Massacci, F.; Zannone, N. Security and Trust Requirements Engineering. In *Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures*; Aldini, A., Gorrieri, R., Martinelli, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 237–272. [CrossRef]

112. Ghanavati, S.; Amyot, D.; Peyton, L. Towards a Framework for Tracking Legal Compliance in Healthcare. In Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE'07), Trondheim, Norway, 11–15 June 2007; Springer:Berlin/Heidelberg, Germany; pp. 218–232.
113. Ghanavati, S.; Amyot, D.; Peyton, L. A Requirements Management Framework for Privacy Compliance. In Proceedings of the Anais do—Workshop em Engenharia de Requisitos (WER'07), Toronto, ON, Canada, 17–18 May 2007; pp. 149–159.
114. Amyot, D. Introduction to the User Requirements Notation: Learning by example. *Comput. Netw.* **2003**, *42*, 285–301. [CrossRef]
115. Amyot, D.; He, X.; He, Y.; Cho, D.Y. Generating scenarios from use case map specifications. In Proceedings of the 3rd International Conference on Quality Software (QSIC'03), Dallas, TX, USA, 6–7 November 2003; pp. 108–115. [CrossRef]
116. Kerrigan, S.; Law, K.H. Logic-based Regulation Compliance-assistance. In Proceedings of the 9th International Conference on Artificial Intelligence and Law (ICAIL'03), Scotland, UK, 24–28 June 2003; pp. 126–135. [CrossRef]
117. Vardi, M.Y. Branching vs. Linear Time: Final Showdown. In *Tools and Algorithms for the Construction and Analysis of Systems*; Margaria, T., Yi, W., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2031, pp. 1–22. [CrossRef]
118. Hashmi, M.; Governatori, G.; Lam, H.P.; Wynn, M.T. Are We Done With Business Process Compliance: State-of-the-Art and Challenges Ahead. *Knowl. Inf. Syst.* **2018**, *57*, 79–133. [CrossRef]
119. Sackmann, S.; Kühnel, S.; Seyffarth, T. Using Business Process Compliance Approaches for Compliance Management with regard to Digitization: Evidence from a Systematic Literature Review. In Proceedings of the 16th International Conference Business Process Management (BPM'18), Sydney, NSW, Australia, 9–14 September 2018.
120. Ramezani, E.; Fahland, D.; van der Werf, J.M.E.M.; Mattheis, P. Separating Compliance Management and Business Process Management. In Proceedings of the 9th International Workshops (BPM'11), Clermont-Ferrand, France, 29 August 2011; Revised Selected Papers; Part II, pp. 459–464. [CrossRef]
121. Casanovas, P.; González-Conejero, J.; de Koker, L. Legal Compliance by Design (LCbD) and through Design (LCtD): Preliminary Survey. In Proceedings of the 1st Workshop on Technologies for Regulatory Compliance Co-Located with the 30th International Conference on Legal Knowledge and Information Systems (JURIX'17), Luxembourg, 17 September 2017; pp. 33–49.
122. Goedertier, S.; Vanthienen, J.; Caron, F. Declarative business process modelling: Principles and modelling languages. *Enterp. Inf. Syst.* **2015**, *9*, 161–185. [CrossRef]