Evidence-based Compliance Engineering

Joris Hulstijn[1]

[1]University of Luxembourg, Esch sur Alzette, Luxembourg

Abstract

Most research on compliance checking is about business processes, focusing on the order and presence of activities, roles, and temporal properties. However, legislation demands evidence of legal conditions. An organization is considered to be compliant to a piece of legislation, when it can demonstrate that the corresponding legal requirements continue to hold for the entire set of cases to which the legislation applies. In this paper we propose a new perspective: *evidence-based compliance engineering*. We sketch how to utilize existing tools for automated theorem proving and natural language understanding, to allow automated verification of legal objectives based on evidence documents. To demonstrate compliance, the system maintains an invariant for the set of cases, that involves legislation, legal requirements, cases and evidence documents. Whenever a change would disrupt this invariant, processes are started to select and analyse documents, update the cases, update the evidence and run the necessary proofs. The compliance status is monitored continuously and can be made visible on a dashboard. The applicability of the approach is illustrated by two examples.

Keywords

compliance checking, evidence, invariant

1. Introduction

Compliance checking involves a lot of administrative work. For example, legislation against anti-money laundering [11, 37, 17] demands that financial institutions must 'know their customer' (KYC). For every transaction, they must verify customer identity, trace their funding, and find the ultimate beneficiary. Solving these issues is complex and many banks fail. For example, Rabobank was fined for failing to uphold customer due diligence [40]. These difficulties in meeting compliance demands are common [33]. Software tools exist for know-your-customer and for anti-money laundering tasks. For example, Computer Assisted Subject Examination and Investigation Tool (CASE*it*), Customer Due Diligence Tool (CDD), tools for name-entity matching, data analysis tools for fine-tuning the suspicious activity detection, and a quick reference guide to track the relevant legislation in various countries [39]. However, each tool only covers part of the investigation task.

Compare this varied landscape of compliance tools to the academic literature on compliance checking. Most approaches focus on business processes [13, 15]. Compliance is interpreted as a set of formal properties, expressed in a form of temporal or deontic logic, which is then verified for all traces generated by the business process. See Governatori and colleagues [30, 19], but also [15, 31] and later [22]. Here is a recent overview [7].

In these works, the legal problem of compliance checking – to verify whether the outcome of

joris.hulstijn@uni.lu (J. Hulstijn)

a process conforms to objectives demanded by law – is reduced to the computer science problem of conformance testing – to verify whether the traces generated by a process specification satisfy formal properties, e.g. [44]. From a computer science point of view, such a reduction is understandable, but from a legal or business point of view, there are several problems:

1. *conformance, not compliance.* Compliance with the law is reduced to conformance to formal properties. Properties defined over execution traces of a process specification, so they cover the order of activities, constraints on roles or resources, and temporal conditions. Other aspects of being compliant, such as organisational practices, governance, risk and controls and legal evidence, are not immediately covered.
2. *no interpretation.* In conformance testing, the properties to test are supposed to be specified upfront. The decision which interpretation of the law must be selected, how that translates into formal properties, and how these properties must be measured or monitored, are not taken into account [5, 18, 32]. See also recent NLP approaches [34, 3].
3. *process, not cases.* In the process-based view, the object of compliance is a specific business process. Instead, as the know-your-customer example shows, the object of the law is a set of cases, representing customers or transactions. These must be shown to comply to requirements, that correspond to the law [16]. The process is only a means to that end.
4. *no evidence* Many administrative processes collect evidence in a dossier, to demonstrate that some legal conditions are met. Evidence usually takes the form of documents, often digitised. In the process based view, no records are stored after verification.
5. *no data analysis.* Compliance monitoring has benefited from process mining [31], but makes insufficient use of advances in databases, ERP systems and analytics [2, 23]. Processes can verify properties of some new or updated cases, but only a data-analysis can verify properties of all cases.
6. *no risk management.* Compliance is never enough. In interpreting the law and implementing controls, managers must make a trade-off between the costs of compliance and the risk of violation [8]. Assisted by compliance officers, management is ultimately responsible for risk management [24], but, the process-based view of compliance reduces it to an operational level.

In this exploratory paper, we propose a new perspective: *evidence-based compliance engineering*, to address some of these problems. The approach centres on legal interpretation and risk management, account for evidence, and combines process-based and data-based approaches to compliance management, in a unified way. The approach is called 'compliance engineering' by analogy with requirements engineering [45]. The central idea is *requirements traceability* [36, 45]. All (legal) requirements lead to specific properties of a system (forward). Conversely, all system properties are motivated by (legal) requirements (backward).

How to develop an evidence-based approach to compliance engineering?

Consider a conceptual model that involves legislation (text), legal requirements (formal representation), and evidence (documents, traces of verified queries), all linked to a database of cases. Such a model is sketched in Figure 1. The model maintains a set of *invariant* properties: (i) all cases conform to the legal requirements, (ii) compliance of cases is supported by evidence,

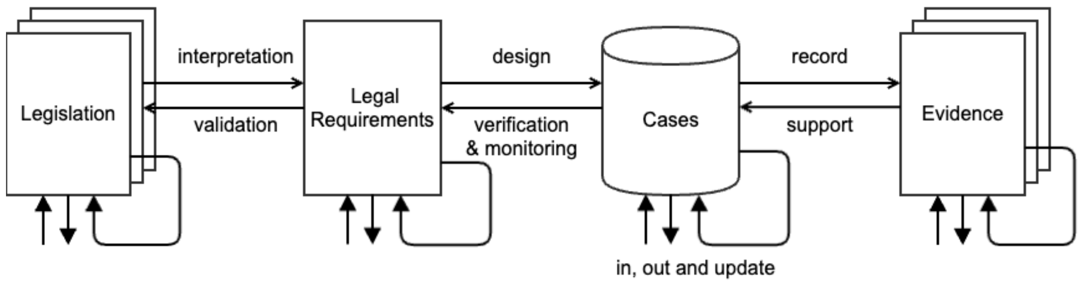

Figure 1: Conceptual model for Evidence-based Compliance Engineering. The model maintains an *invariant*: all cases comply to the legal requirements, which correspond to relevant legislation, and where compliance of cases is supported by evidence.

and (iii) legal requirements correspond to relevant legislation. Automated reasoning, as well as data base and data analytics queries, allow verification of the legal requirements on the data set that represents all cases. Whenever a change would temporarily disrupt the invariant, for example when laws are changed or when cases are added, semi-automatic processes will start, to restore the invariant. These processes are shown in the diagram as *in, out* or *update* arrows. For example, when a new client is entered, due diligence checks are performed and evidence is collected and analyzed, to verify the client conforms to the legal requirements, and therefore to the law. Similarly, when a new law is adopted, the interpretation process is supported by tools for natural language processing and automated reasoning, to derive legal requirements that correspond to the legal interpretation chosen. At any time, the compliance status can be automatically verified, or monitored regularly and made visible on a dashboard.

In this exploratory paper we aim to introduce this vision of compliance engineering. Later, the vision must be further worked out, in the form of an architecture, design principles, and a formal semantics, that makes it possible to specify the invariant property.

The research method is a form of design science [46]. We present an artefact, a conceptual model for evidence-based compliance. The usefulness of the approach is illustrated by two scenarios: (1) know-you-customer, and (2) a building permit.

The remainder of the paper is structured as follows. First, section 2 contains the scenarios. Section 3 provides a theoretical background. Section 4 develops the vision. The paper ends with discussion of the research limitations and suggestions for further research.

2. Scenarios

We analyse two simple scenarios: (1) know-your-customer, and (2) building permit. Summaries are given in two tables: Table 1 and Table 2.

2.1. Know your customer

Legislation against anti-money laundering in various countries [11, 37, 38] demands that financial institutions must 'know their customer' (KYC). For every transaction, these organisations

law:	WVWFT [11]
domain:	financial
object of compliance:	transactions
legal objectives:	(1) relevant details of all transactions are recorded (2) all and only transactions that are deemed unusual according to WVWFT administrative decrees are identified and reported
business objectives:	(3) uninterrupted flow of transactions (4) confidentiality of client details is respected (5) acceptable compliance risk
(some) requirements:	(1) Ensure records of transaction; identity, bank account, authentication and authorisation of sender; identity, bank account, and authentication of receiver; source of funding, identity of ultimate beneficiary, description (2) Maintain lists of indicators of 'unusual'. Develop corresponding criteria. Develop automated tests to match criteria against transaction data. Run tests before committing transactions. Hits are reviewed by compliance officer. Regularly review tests and criteria for effectiveness. (3) % investigated transactions < threshold; duration investigation < threshold; other monitoring carried out off-line (4) Access to system and sensitive fields on need-to-know basis only; most data handled by automated processes; search is blocked; cases reviewed by compliance officers are made anonymous (if possible). (5) number and severity of unusual cases < threshold. number and severity of confidentiality breaches < threshold. all known unusual cases found.

Table 1
Summary of Scenario 1. Know-your-Customer

must verify customer identity, trace their funding, and find the ultimate beneficiary. Subsequently, they must report any suspicious activities to the regulator.

Essentially, there are two main obligations: (1) to investigate all financial transactions, identify the clients, trace the source of funding, and identify the ultimate beneficiaries of the transaction. Here the main purpose is to establish a proper record of each transaction. (2) on the basis of these records, monitor and report any unusual or suspicious transactions, to the regulator. Note that the unusual nature of a transaction may only be found by comparing series of transactions, by advanced data analytics. For example, a statistical outlier, may need further investigation.

For example, in the Netherlands, under the WVWFT [11, §16], businesses must immediately report any unusual transaction to the Financial Intelligence Unit. What is considered to be unusual is defined by indicators, in an administrative decree.

We can make some observations. First, many *different business processes* are involved. Second, these processes need many *different sources of data*, *information* and *evidence documents*. These data sources can be structured in databases, but also be textual, filled-out forms, or even oral testimony. Third, these processes run at *different time-scales*. For example, in executing a transaction, there is an immediate time-scale, but in monitoring, there is a long-term time scale.

Summarising, in the know-your-customer case, a business process is the wrong unit of analysis for compliance purposes.

law:	Municipal Environment plan, Zoning plan, Omgevingswet [12]
domain:	construction
object of compliance:	buildings
legal objectives:	(1) relevant details of all buildings are recorded
	(2) all buildings meet quality and safety criteria
business objectives:	(3) unobstructed flow of housing and renewal projects
	(4) owners demands are respected
(some) requirements:	(1) Ensure records of identity and residence of owner, architect, and construction company, a detailed blue print and construction plan
	(2) Make sure that all and only buildings that meet quality and safety criteria are granted a permit
	(3) % stalled procedures < threshold; average duration < threshold

Table 2
Summary of Scenario 2. Building Permit

2.2. Scenario 2. Building Permits

In most municipalities, a building permit is required before one is allowed to construct or adjust a building. The procedure terminates when a permission is granted, or rejected. A permit is only granted, when it is motivated by documents (blue print; safety assessment), and when these documents show that quality criteria for the building will be met, in the plans. The purpose is to make sure that (1) the municipality has records of all buildings and their construction specifications, and (2) to maintain quality and safety criteria for all buildings. For example, buildings must not collapse under normal use or weather influence, be resistant to fire, and, increasingly, be energy efficient, etc. It is easier to regulate permits, then actual construction of buildings. In case of a mistake in a building permit, the building may have to be demolished.

In the Netherlands, the legal framework for permits has changed. As of 1st of January 2024, after several years of delays due to required IT systems. The Omgevingswet (Environment Act)[12] tries to bundle many types of permits: buildings, waterworks, safety, national heritage, etc, and harmonise the permission procedures of municipalities.

Summarising, for one construction project, often many different processes are involved, and many types of data and information are needed (blue prints; construction plans; assessments). These documents must be archived. The object of compliance is the body of housing; the process is only a means to achieve quality and safety criteria.

3. Theoretical Background

In this section, we provide some theoretical background for the vision shown in Figure 1. This overview must necessarily remain sketchy.

3.1. Information integrity

The core is a database of cases, that is designed to uphold *integrity constraints* [20], see also [9]. Integrity constraints are conditions on the data or information in a database or information system, that must remain true (invariant), based on their type or meaning.

Some integrity constraints are about data types or syntactic constraints. For example, February 30, is not a valid date. Other integrity constraints are semantic. For example, an interest rate is a percentage, so in principle negative interest rates should not exist. Some integrity constraints are relationships. For example, the total income in a year calculated as the sum of all monthly incomes, should equal the income calculated as the sum of incomes generated per division. In accounting, these relational constraints are called reconciliation relations.

Transactions (TPs) are designed in such a way that (1) they ensure the integrity constraints (IVP) for new input data, and (2) they preserve the integrity constraints (IVP) for existing cases [9]. All transactions are logged. Only qualified users, are authorised to execute a transaction. Similar ideas are built into ERP systems and database management systems [20].

3.2. Traceability in Requirements Engineering

Another source of inspiration is requirements engineering, and specifically the idea of *traceability* [14], see also [45]. There are two kinds of traceability: forward and backward. *Forward traceability* is the property that each part of a specification can be traced to specifications of lower-level components that implement it. Forward traceability requires maintenance of links in the *how*-direction, which is down the aggregation hierarchy. *Backward traceability* is the property that for each part of a specification, it is clear why it was included. This requires links in the *why*-direction, which is up the aggregation hierarchy. [45, p. 26]

In our case, each case condition to be tested, is motivated by a legal requirement, which must be in turn be motivated by an interpretation of a legal text or clause (backward). Conversely, each interpretation of a legal text or clause, must lead to some legal requirements, which can be translated into formal conditions, that can be tested for cases in the database (forward).

3.3. Compliance by Design

In compliance, we can say that there are two roles for information systems: (i) *source*: information systems are used to measure, record and store information about compliance behaviour, that can be used as a source of evidence, and (ii) *analysis*: information systems are used to analyse various sources of information, and determine whether the company is compliant.

In both cases, the reliability of the information system itself must be part of the evaluation. For this reason, the information system must be designed with compliance in mind: compliance by design. In particular, *internal controls* must be implemented, to ensure reliability of evidence [10, 41]. Here reliability means correctness (all recorded data corresponds to reality) and completeness (all relevant aspects of reality are recorded as data). The idea is to get the data from the source, as close as possible to the measurement.

Lu et al [30] present *compliance by design* as essentially a preventative approach. Compliance is built into the business processes that support it. We use the term in a broader sense, including trade-offs between compliance risk and other business objectives. To achieve compliance-by-design, several steps have to be followed. (1) *Maintain a controls directory:* representation of relevant control objectives, (2) *Analyse:* compare actual business processes to the control objectives, (3) *Respond:* accept deficiencies found, or redesign the process, and (4) *Monitor:*. This is very similar to the famous plan-do-check-act framework, of Deming.

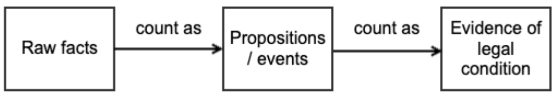

Figure 2: From raw data (source), to information (propositions and events), to evidence of a legal condition (decision)

Actually, there are two cycles, operating at different time scales: (1) urgent, when a risk is identified, and controls must be designed and implemented, and (2) monitoring, to regularly evaluate the risks and effectiveness of controls.

We also mention an approach called *lawfulness by design* [16]. It suggests design patterns to ensure lawfulness of IT artefacts, specifically data sets. By following design patters that work, the cognitive load on the developers of systems can be reduced, and can be used for other tasks.

3.4. Legal Interpretation

Much work on legal informatics (e.g. [6]), wrongly assumes that the law is captured in a single document, that can be translated into a formal representation, to be checked automatically. In fact, there are many legal documents (e.g. national law, administrative degrees, policies), and per document, many different interpretations (e.g strict versus lenient), that will lead to different implementations [5][18]. To decide about these implementations, it is crucial to maintain an intermediate (formal) representation for each separate interpretation, so different interpretations can be analysed and compared. For example, which alternative is cheaper?

The law is often ambiguous [32]. This is deliberate. The law must be future-proof and not written for a specific technology or practice. This is called the *open texture of the law* [21]. Black [4] compares the debate about contested terms, to a from of *regulatory conversations* among stakeholders, that determine what is considered acceptable.

Recently, also NLP approaches to compliance monitoring have addressed the legal interpretation process. They do not require an intermediate formal representation, but instead address changes and differences immediately in the text itself [3, 34].

3.5. Legal rules and evidence

Data has to be aggregated, cleaned and interpreted, before it has meaning as information. Furthermore, information provides evidence of some legal condition that is relevant to some decision (Figure 2). For example, a footprint in the mud means that the suspect was present at the scene on the relevant date, which is enough evidence to hold the suspect in custody.

In this diagram, the meaning relationship is know as 'counts-as', as made popular by Searle [43]; see also [25]. Such counts-as rules are commonly known in AI and law as constitutional rules, which include legal terms and definitions. In a sense, these rules specify what conditions constitute (generate) the institutional facts. Constitutive rules are contrasted with regulative rules, which specify obligations and permissions, and possibly even sanctions, in case of a violation. Regulative rules almost always make use of terms and conditions, specified in constitutional rules. Similar ideas are found in assertion-based auditing [29].

3.6. Continuous Monitoring

Prevention is not enough. No organisation is ever fully compliant, and therefore needs to run regular tests, to find any remaining violations and solve issues. This is the task of compliance monitoring. Our compliance engineering philosophy is inspired by ideas from continuous control monitoring and continuous auditing [1, 27, 23].

Kocken and Hulstijn [26] show that it is possible in principle to provide a continuous assurance service, on the basis of a continuous auditing system (monitoring specific properties of the data stream) combined with a continuous control monitoring system (monitoring operational effectiveness of internal controls), which both feed into a dashboard for showing the current status, and finally, a workflow to request a written assurance document, that summarises the last months of data in the platform, for those clients we need written proof.

3.7. Data Analytics and Data Mining

Data mining has been used extensively in fraud detection and compliance monitoring [35, 2].

A classic problem in fraud detection is to distinguish a set of odd cases from violations of a rule. For example, it may be suspicious if a client takes out a large amount of money in a foreign city, until a telephone call reveals the person is going to get married. Nevertheless, recent advances in anomaly detection use this idea: to identify outliers in streams of data, as indicators of a violation. An advantage is that unsupervised learning removes the need for human experts to annotate data. These techniques have first been tested in an audit context [42], and later applied in many application scenarios.

A general problem is that one depends on the *data quality*. Data that would be crucial for taking decisions is often missing or wrong. For that reason, all data-driven decisions must be cross-verified, for example with the subject. This went terribly wrong in the RoboDebt case, in Australia, where in some cases, a computer system determined the tax debt based on estimates, not real data. This resulted in financial difficulties for the people involved. This situation was made worse by the reluctance of the tax office and responsible politicians, to accept complaints and respond. In general, this illustrates the importance of regular checks, and of allowing feedback and control.

Note in this respect that many compliance applications assume *negation-as-failure*: absence of data or evidence means that the relevant property is false. For example, in the building permit case, if no evidence of a safety check is recorded, we may assume, no check was done. This assumption may only be made, if all cases went through a strict entry process, that ensures data completeness (compare integrity constraints). In practice however, decision making must be made robust to uncertain or incomplete data.

4. Towards an Architecture

We will start with a general vision. After that, we will outline the various components, and finally, detail how these components can be connected.

Take the two views that we have discussed: the *data view* and the *process view*. Here the data view looks at evidence of states of affairs being compliant. The process view looks at

	Components	Purpose
(1)	database of cases	store and retrieve properties of cases, integrity constraints, control mechanisms to maintain integrity constraints
(2)	evidence repository	store and retrieve evidence, verify validity of evidence, summarise evidence documents
(3)	requirements repository	store and retrieve requirement specifications, trace dependencies
(4)	legislation repository	store and retrieve legal sources, trace dependencies

Table 3
List of components

	Links	Tools
(1)-(2)	interpretation	NLP, search, indexing, RE tools
(2)-(1)	validation	RE tools
(2)-(3)	design	compliance-by-design, process mining
(3)-(2)	verification	DBMS, DB query, data mining
	monitoring	continuous audit, anomaly detection
(3)-(4)	record	archiving, indexing, search
(4)-(3)	support	NLP, automated verification

Table 4
List of links between components and useful compliance tools

newly created states of affairs, or at transactions, which change or update those states of affairs. We believe that these two views can be combined into a unified whole, if we make use of an *invariant property* [28]. Compliance, like safety or security, is a property that must always hold. So compliance is specified as an invariant property. Changes or incidents threaten to disrupt compliance. So that triggers a research question: can we ensure by a suitable architecture (system and procedures) that a company remains demonstrably compliant?

Consider the diagram in Figure 1 again. We start from the set of cases. Now suppose that the organisation is actually compliant: they have evidence to demonstrate that all cases in the set satisfy the legal requirements. So the invariant property, for all cases legal compliance is demonstrated by evidence, is true. This can be broken down into three sub-properties (page 3):

(i) all cases conform to the legal requirements,
(ii) compliance of cases is supported by evidence, and
(iii) legal requirements correspond to relevant legislation.

What components are needed? A summary is given in Table 3.

How can such invariant properties be verified and demonstrated? We look at the six arrows in the diagram. (i) To verify compliance, run a set of queries on the cases, that corresponds to the legal requirements (verify and monitor, right to left). The database must be designed in such a way, that it has attributes than can answer such queries (design, left to right). (ii) Moreover, for all claims or assertions stored in the case database, there must be a valid reference to a piece of evidence to support it: a document, a credential, or the outcome of query (support, right-to-left). Conversely, whenever a claim or assertion is tested, either as part of the regular monitoring or as part of processes that handle transactions or changes, outcomes of such tests are recorded (record, left-to-right). (iii) Analogous to forward traceability, for all relevant interpretations of legislation, there must be one or more requirements, that represent them (interpretation, left-to-right). Conversely, for all requirements in the repository, there most be a preferred interpretation of a legal source, that motivates it (validation, right to left).

Now consider all the possible ways in which this invariant can be (temporarily) breached. For example, cases may be added, updated or deleted; evidence may be lost; laws may change, or be re-interpreted, and the IT infrastructure in which these processes is maintained, may change too. For all these potential threats to the invariant property of compliance, adequate business processes must be designed, that will restore the invariant.

In the diagram, these potential changes are shown by three arrows labelled *in*, *out* and *update*, inspired by system input, output and change. In business process management, one often refers to the four basic operations of persistent storage: *create*, *read*, *update* and *delete* (CRUD). They have a similar role. Ideally, read would not seem to alter the invariant, but that is wrong, if we allow for uncertain or incomplete data. Suppose we do a random check, and it appears data is added or removed. In that case, a read will change the outcome.

5. Conclusions.

In this paper we have presented a more ambitious and legally more plausible conceptual model for compliance: evidence-based compliance engineering. By contrast to the dominant view in information systems research, compliance is not predominantly about processes, but rather about data that counts as evidence of continued compliance: an invariant property.

Such a conceptual model of compliance, is more natural for lawyers. It takes trade-offs between compliance and business goals, into account. It would also allow for differences in legal interpretation, using NLP tools. It allows for the advances in data analytics and data mining to be deployed. If done well, it should be robust for incomplete and uncertain data.

So far, this is only a vision. A lot of work remains to be done. First, the vision must be developed into an architecture, of components, and interfaces. Given a proper semantics of the functionality of the components in terms of assertions (Boolean statements about compliance), it must in principle be possible to demonstrate, that the invariant can be maintained, if all components continue to function, and if all in, out and update procedures, perform as assumed.

Second, the architecture must be populated with real tools and techniques, that can be used to perform the functionality on the arrows. That can be then be demonstrated for one or more real-life cases, such as the scenarios in Section 2

References

[1] M. Alles, G. Brennan, A. Kogan, and M. A. Vasarhelyi. Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at Siemens. *International Journal of Accounting Information Systems*, 7:137–161, 2006.

[2] B. Baesens, S. Höppner, and T. Verdonck. Data engineering for fraud detection. *Decision Support Systems*, 150: 113492, 2021.

[3] M. Barrientos, K. Winter, J. Mangler, and S. Rinderle-Ma. Verification of quantitative temporal compliance requirements in process descriptions over event logs. In *Advanced Information Systems Engineering*, volume 13901 LNCS, page 417 – 433, 2023.

[4] J. Black. Regulatory conversations. *Journal of Law and Society*, 29(1):163–196, 2002.

[5] G. Boella, M. Janssen, J. Hulstijn, L. Humphreys, and L. van der Torre. Managing Legal Interpretation in Regulatory Compliance. In B. Verheij, editor, *Proceedings of the XIV-th International Conference on Artificial Intelligence and Law (ICAIL 2013)*, pages 23–32. ACM, Rome, 2013.

[6] T. Breaux and A. Anton. Analyzing regulatory rules for privacy and security requirements. *IEEE Transactions on Software Engineering*, 34(1):5–20, 2008.

[7] J. Carmona, B. van Dongen, and M. Weidlich. Conformance checking: Foundations, milestones and challenges. In W. M. P. van der Aalst and J. Carmona, editors, *Process Mining Handbook*, LNBIP 448, page 155–190. Springer, 2022.

[8] R. Christiaanse and J. Hulstijn. Control automation to reduce costs of control. *International Journal of Information System Modeling and Design*, 4(4):27 – 47, 2013.

[9] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *IEEE Symposium on Security and Privacy*, pages 184–194. IEEE, 1987.

[10] COSO. Internal Control - Integrated Framework. Technical report, Committee of Sponsoring Organizations of the Treadway Commission, 1992.

[11] T. K. der Staten Generaal. Wet ter voorkoming van witwassen en het financieren van terrorisme (wwft). Technical report, Koninkrijk der Nederlanden, 2008.

[12] T. K. der Staten Generaal. Omgevingswet 2016, 2016.

[13] M. Dumas, W. van der Aalst, and H. M. ter Hofstede Arthur. *Process-Aware Information Systems.* John Wiley and Sons., 2005.

[14] A. Egyed and P. Grbacher. Supporting software understanding with automated requirements traceability. *International Journal of Software Engineering and Knowledge Engineering*, 15(5):783–810, 2005. ISSN 02181940.

[15] A. F. S. A. Elgammal, O. Türetken, W. J. A. M. van den Heuvel, and M. Papazoglou. Formalizing and applying compliance patterns for business process compliance. *Software and Systems Modeling*, 15(1):119–146., 2014.

[16] M. S. Ernestine Dickhaut, Andreas Janson and J. M. Leimeister. Lawfulness by design – development and evaluation of lawful design patterns to consider legal requirements. *European Journal of Information Systems*, 0 (0):1–28, 2023.

[17] FinCEN. Beneficial ownership information reporting requirements. Technical report, Financial Crimes Enforcement Network, US Treasury, 2022.

[18] S. Ghanavati and J. Hulstijn. Impact of legal interpretation in business process compliance, 2015.

[19] G. Governatori and S. Sadiq. The journey to business process compliance. In *Handbook of Research on Business Process Management*, pages 426–445. IGI Global, 2009.

[20] P. W. P. J. Grefen and P. M. G. Apers. Integrity control in relational database systems- an overview. *Data and Knowledge Engineering*, 10:187–223, 1993.

[21] H. L. A. Hart. *The Concept of Law.* Clarendon Press, Oxford, 1961.

[22] M. Hashmi, G. Governatori, H.-P. Lam, and M. T. Wynn. Are we done with business process compliance: state of the art and challenges ahead. *Knowledge and Information Systems*, 57(1):79–133, 2018. doi: 10.1007/s10115-017-1142-1.

[23] F. Huang, W. G. No, M. A. Vasarhelyi, and Z. Yan. Audit data analytics, machine learning, and full population testing. *The Journal of Finance and Data Science*, 8:138–144, 2022. ISSN 2405-9188.

[24] IIA. The three lines of defense in effective risk management and control, 2013.

[25] A. J. I. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the Interest Group in Pure and Applied Logic*, 3:427–443, 1996.

[26] J. Kocken and J. Hulstijn. Providing Continuous Assurance. In H. Weigand, editor, *Proceedings of the 11th International Workshop on Value Modeling and Business Ontologies (VMBO 2017).* Luxembourg Institute of Science and Technology (LIST), Luxembourg, 2017.

[27] A. Kogan, M. G. Alles, and M. A. Vasarhelyi. Design and evaluation of a continuous data level auditing system. *Auditing: A Journal of Practice and Theory*, 33(4):221–245, 2014.

[28] L. Lamport. A new approach to proving the correctness of multiprocess programs. *ACM Transactions on Programming Languages and Systems*, 1:84–97, 1977.

[29] D. A. Leslie, S. J. Aldersley, D. J. Cockburn, and C. J. Reiter. *An Assertion Based Approach to Auditing*, pages 31–64. University of Kansas, Kansas, 1986.

[30] R. Lu, S. Sadiq, and G. Governatori. Measurement of Compliance Distance in Business Work Practice. *Information Systems Management*, 25(4):344–355, 2009.

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

[32] A. K. Massey, E. Holtgrefe, and S. Ghanavati. Modeling regulatory ambiguities for requirements analysis. In H. C. Mayr et al., editors, *Proceedings of the 36th International Conference on Conceptual Modeling (ER 2017)*, LNCS 10650, pages 231–238. Springer, 2017.

[33] A. Merz. 'it could have been us'. peer responses to money-laundering violations in the dutch banking industry. *Crime, Law and Social Change*, 2023. doi: https://doi.org/10.1007/s10611-023-10120-y.

[34] H. Mustroph, M. Barrientos, K. Winter, and S. Rinderle-Ma. Verifying resource compliance requirements from natural language text over event logs. In *Business Process Management (BPM 2023), Utrecht, The Netherlands*, LNCS14159, pages 249–265. Springer, 2023.

[35] E. Ngai, Y. Hu, Y. Wong, Y. Chen, and X. Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, page 559–569, 2011.

[36] J. D. Palmer. Traceability. In R. H. Thayer and M. Dorfman, editors, *Software Requirements Engineering*, page 364–374. IEEE Computer Society Press, 1997.

[37] Parliament. The money laundering and terrorist financing (amendment) (no. 2). Technical Report No. 860, United Kingdom, 2022.

[38] U. K. Parliament. Proceeds of crime act, 2002.

[39] Pwc. Using the right tools for anti-money laundering compliance, 2023. URL https://www.pwc.com/us/en/industries/financial-services/financial-crimes/anti-money-laundering/compliance-tools.html.

[40] Reuters. Rabobank faces punishment over customer anti-money-laundering checks, 2021.

[41] M. B. Romney and P. J. Steinbart. *Accounting Information Systems*. Pearson Education, 14th edition, 2018.

[42] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer. Detection of anomalies in large scale accounting data using deep autoencoder networks. *CoRR*, (abs/1709.05254), 2017.

[43] J. R. Searle. *The Construction of Social Reality*. The Free Press, 1995.

[44] S. C. Tosatto, G. Governatori, and P. Kelsen. Business process regulatory compliance is hard. *IEEE Transactions on Service Computing*, 8(6):958 – 970, 2014.

[45] R. J. Wieringa. *Requirements Engineering: Frameworks for Understanding*. Wiley, 1996.

[46] R. J. Wieringa. *Design science methodology for information systems and software engineering*. Springer Verlag, London, 2014.