A Model-Based Framework for Simplified Collaboration of Legal and Software Experts in Data Protection Assessments

Nicolas Boltz [1], Leonie Sterz [2], Christopher Gerking [3], Oliver Raabe [4]

Abstract: The protection of personal data has become an increasingly important issue. Legal norms focused on data protection, such as the EU General Data Protection Regulation (GDPR), provide legally binding requirements for systems that process personal data. Article 25 of the GDPR refers to the obligation to Data Protection by Design and Default. This can be achieved by conducting legal assessments of the system in the early stages of development and implementing data protection concepts where necessary. This ties in with Article 35, which refers to an obligation to conduct legal assessments before the actual processing of data.

To aid in conducting continuous legal assessments during the design time of software systems, we propose a model-based collaboration framework. This framework not only aids in providing consistent views of the software system for legal experts and software architects but also simplifies communication between both parties. We discuss the overall goals and benefits of such a framework and go into detail about the processes that interact as part of the framework. We also try to align legal concepts with the processes and describe the continuous iterative development using the collaboration framework.

Keywords: data protection by design; legal assessment; GDPR; software architecture; metamodel; design time

1 Motivation

Due to the increasing digital transformation of everyday life, the protection of personal data has become a more and more important issue for society. Legal norms such as the EU General Data Protection Regulation (GDPR) are being increasingly enforced and penalties for non-compliance are high. Also, Article 25 of the GDPR refers to the obligation to Data Protection by Design (DPbD) and Default, i.e. the implementation of data protection concepts during the design time of the system.

[1] Karlsruhe Institute of Technology, Institute of Information Security and Dependability (KASTEL), Dependability of Software-intensive Systems Group (DSiS), nicolas.boltz@kit.edu

[2] Karlsruhe Institute of Technology, Institute of Information Security and Dependability (KASTEL), Information Law for Technical Systems and Legal Informatics Group (ITR), leonie.sterz@kit.edu

[3] Karlsruhe Institute of Technology, Institute of Information Security and Dependability (KASTEL), Dependability of Software-intensive Systems Group (DSiS), christopher.gerking@kit.edu

[4] Karlsruhe Institute of Technology, Institut für Informations- und Wirtschaftsrecht (IIWR), Information Law for Technical Systems and Legal Informatics Group (ITR), raabe@kit.edu

doi:10.18420/inf2022_44

One way to achieve DPbD is to offer concepts during the design time of a system, that aid in the process of designing a system focused on preserving data privacy. During design time, software architecture models are created, which represent the structure and context of the system under development (SUD). Using only software architecture models, it is already possible to derive quality attributes of the future system such as confidentiality [Se22] (besides other attributes like performance or reliability [Re16]). Through an iterative development process, quality issues can be found by analyzing the architecture models, and later resolved by fixing or optimizing the system's architecture. As the cost of fixing an issue increases drastically the later it is fixed [BMU75], in the best case issues should be fixed during design time. Consequently, data protection issues should also be addressed during the design of the system. Furthermore, besides the obvious privacy problems for the individual, which may arise from the improper processing of personal data, issues with data protection potentially entail high penalties and loss of reputation for the organizations handling the data.

While there already exist approaches for privacy and data protection engineering during design time [Ah18; Bi16; Si19] and the definition of context-specific legal requirements [Gr20; MCP21; To21], the interdisciplinary communication and collaboration between legal and technical domains is often overlooked. However, this subject, in particular, has the potential to be critical, as both professions are highly specialized in their field, and in-depth knowledge about the other profession can not be assumed. Even more critical is the fact that similar-sounding terms regarding privacy and data protection might refer to disparate underlying understandings or concepts. During the collaboration, e.g. while conducting a legal assessment of a software system, these similarities, which constitute the basis of the conversation, create the need to first define a uniform terminology to avoid potential misunderstandings. Even if a uniform terminology can once be successfully established, new communication problems are likely to emerge in case of legal change. If the legal situation or its interpretation change over time, these changes must be communicated to the technical personnel, so that they can be reflected in the design of the SUD. A failure to communicate such changes quickly and correctly can not only delay the rollout of systems. If unnoticed, it may even lead to the rollout of non-compliant systems involving the aforementioned financial penalties or cost-intensive product recalls.

We propose a collaboration framework, where direct communication between legal experts and technical personnel is simplified. The framework aims to aid in DPbD and the development of legally compliant systems. Software architects plan a software system, creating a software architecture model in the process. This software architecture model represents a technical view of the system. Information about the software system, required by the legal experts, is provided by another specialized view that is created by transferring, translating, and summarizing information from a software architecture model. This view omits technical information not relevant for a legal assessment and visualizes the system in a way that is more catered toward the legal domain. Furthermore, legal experts use this view to perform a subsumption, identifying a system-specific set of rules, which cover more complex legal norms. This set of rules is translated to technical analysis queries, that are used to analyze the actual software architecture model. If violations of data protection law are found using the translated rules, the software architecture model can be adjusted as consequence. Each view of the system is model-based and relationships between models are based on transformations or manual processes. As the iteration of the framework ends in the potential adjustment of the planned software system, the resulting software architecture model can feed into a new iteration. This allows for the *continuous* development of the software architecture and conducting of legal assessments at the same time.

The remainder of this paper is structured as follows: Section 2 gives an overview of the foundations regarding the topic of software architecture modeling and data protection legal assessments. Section 3 describes the problems we aim to address and the goals we aim to achieve with the proposed framework. In Section 4 we describe the proposed collaboration framework in detail. In Section 5 concepts from legal methodology are mapped to the corresponding framework element. Section 6 describes the continuous development and assessment process. Section 7 presents related work and Section 8 concludes the paper.

2 Foundations

In this section, we give a brief overview of the main concepts that make up our proposed collaboration framework. In Subsection 2.1 we describe two established approaches to modelling and analyzing software architecture. Subsection 2.2 provides an overview of what a legal assessment is and the different guidelines and methodologies that exist.

2.1 Modelling and Analyzing Software Architecture

There are multiple approaches to describe and analyze software architecture. We focus on Data Flow Diagrams (DFDs) as a lightweight data-centric representation of a software system and the Palladio Component Model (PCM) [Re16] as a detailed representation.

DFDs based on the notion of DeMarco [De79] are unidirectional graphs representing the flow and processing of data in a system. Graph nodes are either Actors, Processes, or Stores. Edges that connect nodes are called data flows and describe a data transmission between the connected nodes. Actors are the source and sink nodes, which start or terminate a sequence of data flows. Process nodes transform incoming data and pass it on as outgoing data. Store nodes hold or emit data. The approach of Seifermann et al. [Se22] extends the DFD notion and provides a way to analyze DFDs for illegal or unwanted data flows. Queries that describe what kind of data flows are unwanted are defined in a domain-specific language (DSL) [Ha21].

Palladio is a tool-supported software architecture simulation approach, that is used to predict an architecture's quality of software properties [BKR09]. The PCM is a model of component-based software architectures. The basic PCM is made up of several metamodels, each representing a different architectural view of a system [Re16]. The approach of Seifermann et al. [Se22] also provides a way to analyze an extended variant of the PCM using the same DSL as described above.

2.2 Legal Assessments

During a legal assessment, the *facts relevant for the legal assessment*[5] are examined for their legal conformity. Usually, such an assessment refers to an aspect of the law or a specific area of the facts. Our framework has a focus on data protection law. Therefore, our legal assessment concentrates on substantive lawfulness according to Art. 6 GDPR as well as the principles relating to the processing of personal data such as data minimization and purpose limitation (Art. 5 GDPR) and security of processing (Art. 32 GDPR). Article 35 requires that data protection impact assessments should be conducted where the type of processing of personal data results in a high risk to, or if there is a possible change in the risk of the privacy of the natural person. Instead of only focusing on the impact on high-risk areas, our assessment is a more general kind and considers the whole system.

3 Goals

In this section, we describe the problems we aim to address and the goals we aim to achieve with our framework.

The most prominent problem, which we have already touched on in Section 1, is the knowledge gap between legal experts and software architects or engineers. In the legal domain, for example, anonymization is measured using objective factors, such as cost and time for re-identification [GD16, Rec. 26]. However, in the technical domain, anonymization is often measured using concepts from theoretical computer science and mathematical definitions, like k-Anonymity [Sw02]. By providing our framework, we aim to reduce and simplify direct communication that is necessary while conducting a legal assessment, which in turn reduces the potential for misunderstandings. Further, the predefined models and automatic transformations reduce the need to define a uniform terminology, making collaboration more effective and simple.

Another problem that arises due to our view-based approach is maintaining consistency between the views. The software architecture model represents the SUD and is subject to constant changes and development. A view of the SUD catered towards the legal domain, could simply be derived from the software architecture model once at a certain point during development. However, since changes of the software architecture can occur frequently, the legal view might be outdated shortly after. A legal assessment that is based on an outdated

[5] Translated from German legal term ”Sachverhalt”

view might come to false conclusions, which do not correspond to the actual SUD. Even in later phases, the legal view of the system must always be consistent with the software architecture, so that the change to the system can be assessed. Our proposed framework achieves this through the automatic transformation from software architecture model to legal view.

The other way is more complex, as the legal view is used to conduct a legal assessment and identify legal consequences. These consequences can entail requirements and potential changes, that legal experts can not sensibly incorporate into the system. However, our framework provides the legal experts with a way to formulate the legal requirements, that allows for automated translation into technical analysis queries. These queries in turn can be used to automatically analyse the software architecture model for compliance with the legal requirements. This way, our proposed framework allows the views of the system to evolve at the same pace and enables the development of software architecture and conducting legal assessments in parallel.

Wright et al. [WFR13] describe, that the assessments of the privacy/data protection should be a process that begins at the earliest possible stages of system development and continue even after deployment. Our proposed framework aims at providing such a process, by enabling continuous legal assessment of the system. If the software architecture model is kept consistent with the actual runtime system, our proposed framework can also be applied in subsequent phases of system development, keeping an already running system compliant with data protection law. Through continuous assessments, changes in the system can be addressed more quickly. This is supported by the fact that an already existing assessment conclusion, in the form of formulated legal requirements, might only need minor adjustments and the technical analysis process of the software architecture can be performed automatically. Changes in the interpretation of data protection law, due to new legal norms or court rulings, can be addressed similarly.

4 Framework Description

As briefly described in Section 1 and Section 3, the proposed framework aims to simplify communication between legal experts and software architects and in turn aid in DPbD and the development of legally compliant systems. As shown in Figure 1, each view of the system is model-based and consists of a structural model representing the SUD and a rule model representing legal rules that are either derived from or applied to the corresponding structural model.

On the technical side, the structural model is a software architecture model, as described in Subsection 2.1. As the software architecture model informs how the SUD is eventually implemented, it undergoes constant changes during planning and development. Information about the software system, required by the legal experts, is provided by the structural model of the legal view. This structural model called *Legal Assessment Facts Model* is part of our proposed approach. It contains the information necessary for legal assessments of the system. A model is created by transforming relevant information from the software architecture model and shifting the focus in a way that is more intuitive for legal experts. Legal experts enrich the resulting *Legal Assessment Facts Model* with law-specific elements that cannot be derived from the software architecture model or modelled in the software architecture model. Based on the enriched model, properties of the system that are for example defined in the GDPR, such as *applicability* according to Article 2, *purpose limitation* according to Article 5(1)c or *lawfulness of processing* according to Article 6(1), can already be checked automatically. With these checks, elements that do not fulfill these can be identified. Based on the identified elements, legal experts can either implement legal measures, such as adapting consent forms, contracts, or purpose definitions, to establish the properties, or inform the technical side to initiate changes in the software architecture model. Using the *Legal Assessment Facts Model*, the legal experts conduct further legal assessment and identify additional legal consequences, which apply to the system. These legal consequences can require the system to have specific properties or, make active changes to the system. The changes are often not trivial to implement correctly, which is why they cannot simply be implemented by legal experts in the *Legal Assessment Facts Model* and transformed back to the software architecture model. Instead, the legal experts manually derive a *Legal Norm Rules Model*. This model represents the rule model of the legal view. Based on the identified legal consequences, the legal experts derive legal requirement rules, which are translated into technical analysis queries. The queries can subsequently be used to check the software architecture model for compliance with the legal requirements.

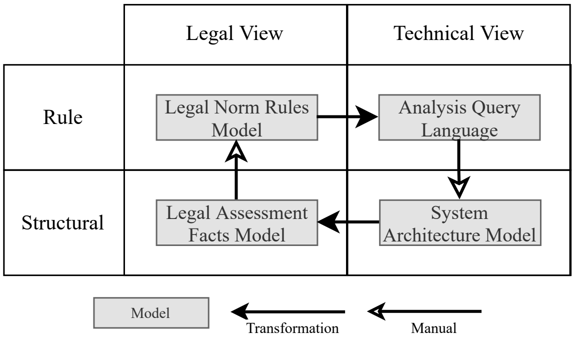

Fig. 1: Models and relationships, categorized by type of information and domain.

5 Legal Methodology

To better align the overall technical process described in Section 4 with the legal domain, we describe the legal concepts that are part of the process. Figure 2 shows the framework with annotated legal concepts. As the software architecture model describes how the SUD is eventually implemented, it represents the *matter of fact*[6]. Based on the matter of fact,

[6] Translated from German legal term ”Tatsache”

the *facts relevant for the legal assessment*[7], need to be captured. As we aim to aid in conducting legal assessments of software systems, the transformation step between software architecture model and *Legal Assessment Facts Model* captures the facts which are relevant for a data protection legal assessment. The structure of the *Legal Assessment Facts Model* is influenced by the information necessary for legal assessments, but also by data protection legal norms in general.

The manual step in which the *Legal Norm Rules Model* is created starts with the identification of the relevant legal norms by the legal expert. The laws contain various *legal consequences*[8] and the respective prerequisites, which are called *definitional elements of a rule*[9]. If all definitional elements of a rule are met, the corresponding legal consequence applies. The legal expert uses the *Legal Assessment Facts Model* to check whether the definitional elements are met. This step is called *subsumption in the narrower sense*[10] in continental European law [Ra12]. The *interpretation of a law*, for example on the basis of court decisions, can also play a role here. Depending on whether or not all relevant facts are fulfilled, it can be determined whether the legal consequence under consideration applies. This is also called *subsumption in the broader sense*[11]. The legal expert transfers the results of his examination into logical and technically readable rules. These result in the *Legal Norm Rules Model*.

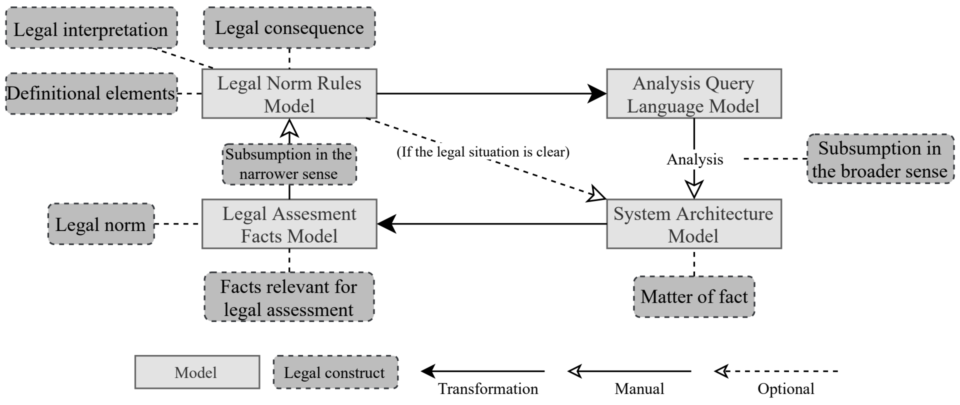
Legal interpretation
Legal consequence
Definitional elements
Legal Norm Rules Model
Analysis Query Language Model
Subsumption in the narrower sense
(If the legal situation is clear)
Analysis
Subsumption in the broader sense
Legal norm
Legal Assesment Facts Model
System Architecture Model
Facts relevant for legal assessment
Matter of fact
Model
Legal construct
Transformation
Manual
Optional

Fig. 2: Concepts of legal methodology, assigned to the corresponding framework elements.

6 Continuous Assessment Process

The process starts as a technical software architecture modeling activity. A software architect plans a software system, creating a software architecture model which represents the SUD.

[7] Translated from German legal term ”Sachverhalt”
[8] Translated from German legal term ”Rechtsfolge”
[9] Translated from German legal term ”Tatbestandsmerkmal”
[10] Translated from German legal term ”Subsumtion im engeren Sinne”
[11] Translated from German legal term ”Subsumtion im weiteren Sinne”

Using a transformation algorithm, information that is relevant for legal assessments is extracted from the software architecture model and a *Legal Assessment Facts Model* is created. A legal expert uses the *Legal Assessment Facts Model* to perform a subsumption in the narrower sense (see Subsection 2.2). As described in Section 4, a *Legal Norm Rules Model* is derived which is custom to the SUD, as represented in the *Legal Assessment Facts Model*. In the following activity, the *Legal Norm Rules Model* is translated to a technical *Analysis Query Language Model*. During this process, trance links from the first transformation to the analysis *Legal Assessment Facts Model* are used to link rules to actual elements of the software architecture model. Using the analysis process of Seifermann et al. [Se22] described in Subsection 2.1, the software architecture model is analyzed with the queries from the resulting *Analysis Query Language Model* to determine if the SUD meets all data protection requirements. Any found data protection requirements that are not met, represent an architectural element or design decision in the software architecture model, that is not compliant with data protection law and requires adjustment. How the software architecture needs to be adjusted is either evident from the requirement or needs to be determined by the software architect.

After the initial iteration, the proposed process is made up of two concurrent loops: The first loop, shown on the right side of Figure 3, only contains technical activities. Once a *Legal Norm Rules Model* has been created by a legal expert, this loop can be iterated independently from any legal cooperation. The software architect can continue to adjust and further plan the software architecture model, analyzing the system's legal conformity using the most current *Legal Norm Rules Model* available. With each iteration, it can be assured that at least a subset of data protection requirements that are not met by the software architecture can be identified. Rules that reference a software architecture element, that has been removed or substantially modified, can be ignored by the analysis. Data protection requirements that are not met, can then be mitigated by making adjustments to the software architecture and starting a new iteration of this loop.

The second loop, on the left side of Figure 3, primarily contains legal activities. Similar to the initial iteration of the process, the now newly adjusted software architecture model is transformed to an updated *Legal Assessment Facts Model*. Based on the updated model, a legal assessment is conducted and the *Legal Norm Rules Model* is updated as a result. The updated *Legal Norm Rules Model* is then translated to a technical *Analysis Query Language Model*, which is used to automatically analyze the software architecture model to find data protection requirements that are not met. If the software architecture is changed, either as a result of a previously not met data protection requirement or as part of the development of the system, a new iteration of this loop is started.

The main benefit of our proposed approach is that both loops can be iterated independently from each other. This allows for the continuous development of the software architecture and conducting of legal assessments at the same time. As each loop ends in the adjustment of the planned software system, the results of a loop feed into the next iteration.

As can be seen in Figure 3, the communication activities that connect technical and legal activities are all based on automated transformations, removing the need for direct interaction. Only if there are very relevant legal consequences, which are obvious and that need immediate attention, direct interaction is necessary.

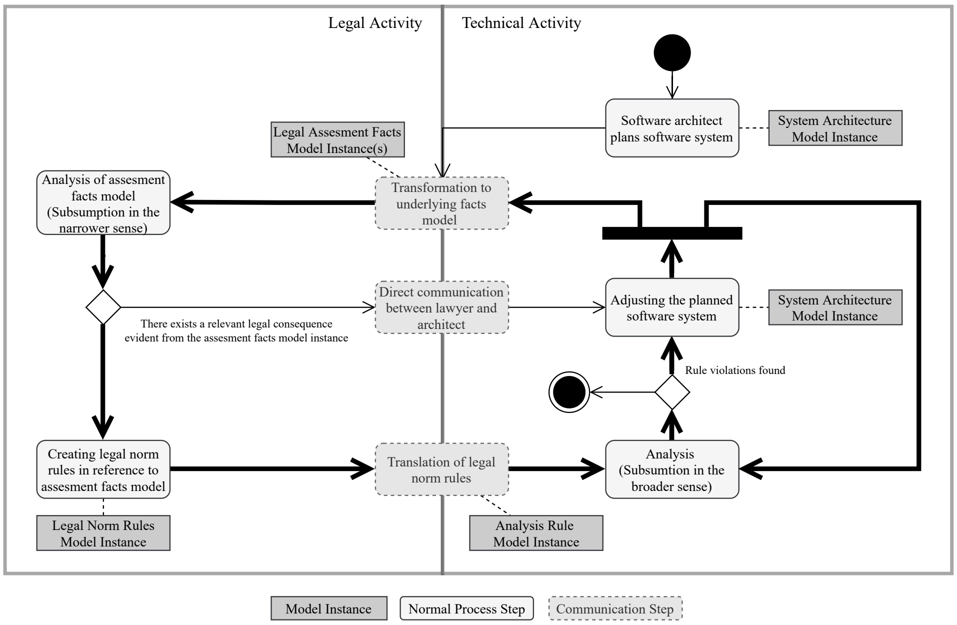
Legal Activity
Technical Activity
Software architect plans software system
System Architecture Model Instance
Legal Assesment Facts Model Instance(s)
Analysis of assesment facts model (Subsumption in the narrower sense)
Transformation to underlying facts model
Direct communication between lawyer and architect
There exists a relevant legal consequence evident from the assesment facts model instance
Adjusting the planned software system
System Architecture Model Instance
Rule violations found
Creating legal norm rules in reference to assesment facts model
Translation of legal norm rules
Analysis (Subsumtion in the broader sense)
Legal Norm Rules Model Instance
Analysis Rule Model Instance
Model Instance
Normal Process Step
Communication Step

Fig. 3: Activity diagram of the continuous assessment and development process.

7 Related Work

In this section, we discuss other approaches that are related to our proposed framework. Furthermore, as some elements of our described approach still need to be implemented, parts of related work also constitute a base for future work.

In the years since the GDPR was published, there have been a variety of model-based approaches to legal compliance and interdisciplinary cooperation. Torre et al. [To21] define a visual representation of the GDPR, as a way to provide a communication bridge between IT and legal experts. They define a generic GDPR model and invariants expressed in the Object Constraint Language (OCL) that can be used for automated GDPR compliance checking. Further, they describe how the generic model can be specialized depending on the context of the application. However, the specialized model is not derived from a system representation using a formal transformation but rather created by manually changing variation points of the GDPR model. Consequently, changes in the system require manual adjustment of the specialized model. Additionally, changes in the interpretation of law require direct changes of OCL constraints or the addition of variation points, which might be challenging for legal experts. Sion et al. [Si19] propose an architectural view for DPbD. The view is based on a model, that is derived from definitions of the GDPR. They describe how a transformation could keep the view consistent with a design time model so that there is no discrepancy between view and the actual system. While their approach covers similar aspects as the structural part of our proposed collaboration framework, the process which involves legal experts utilizing the DPbD view and feeding information back into the design time model is not explored. Ahmadian et al. [Ah18] propose a methodology to support data protection impact assessments by performing model-based privacy analyses during design time. They analyze the design of a system and where necessary suggest privacy controls to improve the design. The privacy requirements are defined by annotating a UML diagram with elements of a privacy-focused UML profile. However, the privacy UML profile extension does not directly reference requirements defined by the GDPR. Further, as only a technical UML view of the system is defined, there exists no tailored view for legal experts to work with.

The previously described approaches focus on representing and extending a model in a way that it can be either manually or automatically assessed. In correspondence to our described *Legal Norm Rules Model*, there also exist approaches that focus on the definition of specific legal requirements. Grifo et al. [Gr20] define a domain-specific visual modeling language, which provides for a visual symbolic representation for legal statements. It aims to support legal experts in identifying legal requirements and to assist in the visual representation of case scenarios. Merigoux et al. [MCP21] define the programming language CATALA, which is designed to allow legal experts to translate law into a formal form, that can be further translated into an executable implementation. This approach could be used to represent the *Legal Norm Rules Model*, as it allows for the definition of legal requirements and already provides a translation, which could be leveraged to analyse the SUD.

8 Conclusion and Future Work

In this paper, we present a collaboration framework for continuous legal assessments of software systems during design time. Our proposed framework aims to aid in DPbD and the development of legally compliant systems.

The framework is made up of two views: A technical view, aimed at the technical domain and a legal view aimed at the legal domain. Each view is made up of two models. One represents the structure of the SUD the other a set of rules for compliance with data protection law. The relationships between models of different views are based on automated transformations. Software architects plan a software system, creating a software architecture model. Information about the software system, required by the legal experts, is provided by transferring information from the software architecture model to a *Legal Assessment Facts Model*. The *Legal Assessment Facts Model* omits technical information not relevant for a legal assessment. Legal experts enrich the resulting *Legal Assessment Facts Model* with law-specific elements that cannot be derived from the software architecture model. They use the *Legal Assessment Facts Model* to apply the law and transform their results into a formal form, creating a *Legal Norm Rules Model* in the process. The *Legal Norm Rules Model* is translated to a technical *Analysis Query Language Model*, which is used to analyze the actual software architecture model.

We define the problems we aim to solve and describe the continuous process, that allows for simultaneous software development and legal assessments. Our framework combines technical and legal methodology and workflows, while at the same time reducing and simplifying the required direct communication between both domains. Through the iterative nature and automated transformations that are part of the proposed process, both views of the SUD are kept consistent with each other. The continuous assessment process not only aligns with the requirements of law but also allows for quicker reactions to changes in the system or law respectively.

In future work, we plan to further implement parts of our proposed framework. Using related work and interdisciplinary collaboration we aim to define a concrete *Legal Assessment Facts Model* and *Legal Norm Rules Model* suited for our described purpose. Further, we plan on evaluating the technical aspects of our proposed framework with a case-study-based evaluation and the applicability and effectiveness with a user study. As this work is part of mobility-focused projects, in a first effort we aim to create a case study for evaluation which is thematically located in the area of connected cars and vehicle to X communication.

Acknowledgments

This publication is partially based on the research project SofDCar (19S21002), which is funded by the German Federal Ministry for Economic Affairs and Climate Action. This work was also supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs (46.23.03).

References

[Ah18] Ahmadian, A. S.; Strüber, D.; Riediger, V.; Jürjens, J.: Supporting privacy impact assessment by model-based privacy analysis. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. Pp. 1467–1474, 2018.

[Bi16] Bieker, F.; Friedewald, M.; Hansen, M.; Obersteller, H.; Rost, M.: A process for data protection impact assessment under the european general data protection regulation. In: Annual Privacy Forum. Springer, pp. 21–37, 2016.

[BKR09] Becker, S.; Koziolek, H.; Reussner, R.: The Palladio component model for model-driven performance prediction. Journal of Systems and Software 82/1, pp. 3–22, 2009.

[BMU75] Boehm, B. W.; Mcclean, R. K.; Urfrig, D.: Some experience with automated aids to the design of large-scale reliable software. Transactions on Software Engineering/1, pp. 125–133, 1975.

[De79] DeMarco, T.: Structure analysis and system specification. In: Pioneers and Their Contributions to Software Engineering. Springer, pp. 255–288, 1979.

[GD16] GDPR - The European Parliament and the Council of the European Union: EU General Data Protection Regulation (GDPR), Apr. 27, 2016, URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

[Gr20] Grifo, C.; Teixeira, M. D. G. D. S.; Almeida, J. P. A.; Gailly, F.; Guizzardi, G.: LawV: Towards an ontology-based visual modeling language in the legal domain. In: ONTOBRAS 2020. Vol. 2728, pp. 75–88, 2020.

[Ha21] Hahner, S.; Seifermann, S.; Heinrich, R.; Walter, M.; Bureš, T.; Hnětynka, P.: Modeling data flow constraints for design-time confidentiality analyses. In: International Conference on Software Architecture Companion (ICSA-C). IEEE, pp. 15–21, 2021.

[MCP21] Merigoux, D.; Chataing, N.; Protzenko, J.: Catala: a programming language for the law. Proceedings of the ACM on Programming Languages 5/ICFP, pp. 1–29, 2021.

[Ra12] Raabe, O.; Wacker, R.; Oberle, D.; Baumann, C.; Funk, C.: Subsumtion im engeren Sinne. In: Recht ex machina: Formalisierung des Rechts im Internet der Dienste. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 263–274, 2012.

[Re16] Reussner, R. H.; Becker, S.; Happe, J.; Heinrich, R.; Koziolek, A.; Koziolek, H.; Kramer, M.; Krogmann, K.: Modeling and Simulating Software Architectures – The Palladio Approach. MIT Press, 2016.

[Se22] Seifermann, S.; Heinrich, R.; Werle, D.; Reussner, R.: Detecting violations of access control and information flow policies in data flow diagrams. Journal of Systems and Software 184/, p. 111138, 2022.

[Si19] Sion, L.; Dewitte, P.; Van Landuyt, D.; Wuyts, K.; Emanuilov, I.; Valcke, P.; Joosen, W.: An architectural view for data protection by design. In: International Conference on Software Architecture (ICSA). IEEE, pp. 11–20, 2019.

[Sw02] Sweeney, L.: k-Anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10/5, pp. 557–570, 2002.

[To21] Torre, D.; Alferez, M.; Soltana, G.; Sabetzadeh, M.; Briand, L.: Modeling data protection and privacy: application and experience with GDPR. Software and Systems Modeling 20/6, pp. 2071–2087, 2021.

[WFR13] Wright, D.; Finn, R.; Rodrigues, R.: A comparative analysis of privacy impact assessment in six countries. Journal of Contemporary European Research 9/1, 2013.