Handling Undefined Legal Terms in Architecture-based Data Protection Analysis

Bachelor's Thesis of

Felix Schwickerath

At the KIT Department of Informatics
KASTEL – Institute of Information Security and Dependability

First examiner:	Prof. Dr. Ralf Reussner
Second examiner:	Prof. Dr.-Ing. Anne Koziolek
First advisor:	M.Sc. Nicolas Boltz
Second advisor:	M.Sc. Sebastian Hahner

10. June 2024 – 10. October 2024

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Abstract

With an ever-increasing amount of software-driven solutions in various parts of modern life, like smart homes, public health or transportation services, more data than ever is collected and analyzed. This surge in data processing raises concerns and led to legal regulations on the processing of personal data, like the General Data Protection Regulation (GDPR). With both legal experts and software architects assessing whether their system is GDPR-compliant, efficient interdisciplinary communication between the legal and technical domain are needed. Matters are complicated by Undefined legal terms (ULTs) that force legal experts during their analysis to consider the context and environment of a system, as this information is needed to assess the meaning of the legal term. Additionally, the context and environment of a system consists of the technical environment, so the software architect needs to communicate the context of each step in the system as well. This communication is tedious and error-prone, so more efficient approaches are needed. Furthermore, when software models grow larger, the context and environment of a system does as well, complicating the legal assessment.

In this thesis, I introduce my extension of a data flow analysis and legal assessment framework that allows legal experts and software architects to work collaboratively on a software model. It also addresses the issues that are caused during the legal analysis of a system by ULTs and allows legal experts to analyze larger software models effectively. The methodology and results of the analysis are presented in a way that legal experts and software architects may understand what caused an issue in the model and how it may be fixed. Information about context is either extracted from the existing software model or annotated by a legal expert or software architect and is available in the results for further analysis and investigation.

The findings in this thesis are evaluated though a case study-based evaluation that assesses the feasibility of the presented approach. The evaluation includes a goal to determine whether the analysis results are accurate and another goal that assesses the scalability of the presented approach in different scenarios. The evaluation comes to the conclusion that the results of the analysis are accurate, and the analysis can be used to evaluate larger software models in reasonable time.

Zusammenfassung

Mit einer wachsenden Menge an softwaregetriebenen Lösungen in verschiedenen Teilen modernen Lebens, wie Smart Homes oder öffentliche Gesundheit- oder Transportlösungen, werden mehr und mehr Daten gesammelt und analysiert. Dieses Wachstum in der Datenverarbeitung hat Bedenken verursacht und letztlich zu rechtlichen Regulierungen zur Verarbeitung von persönlichen Daten, wie der Datenschutz-Grundverordnung (DSGVO), geführt. Mit Rechtsexperten und Softwarearchitekten, die bewerten, ob ein System der Datenschutz-Grundverordnung entspricht, wird effektivere interdisziplinäre Kommunikation zwischen der technischen und rechtlichen Domäne benötigt. Diese Kommunikation wird weiter erschwert durch undefinierte Rechtsbegriffe, welche Rechtsexperten dazu zwingt den Kontext und die Umgebung eines Systems mit in die rechtliche Bewertung einfließen zu lassen, da diese Informationen in der rechtlichen Analyse benötigt werden. Zusätzlich bestehen Teile dieses Kontextes aus der technischen Umgebung, sodass der Softwarearchitekt die technische Umgebung für jeden Verarbeitungsschritt kommunizieren muss. Im Allgemeinen ist diese Kommunikation aufwendig und fehleranfällig, sodass effektivere Ansätze notwendig sind. Weiter wächst mit einem größeren Softwaremodell auch die Größe des Kontextes und die Umgebung eines Systems, sodass die rechtliche Bewertung weiter erschwert wird.

Diese Abschlussarbeit präsentiert meine Erweiterung einer Datenflussanalyse und eines rechtlichen Bewertungsprozesses, sodass Rechtsexperten und Softwarearchitekten kollaborativ an Softwaremodellen arbeiten können. Die Arbeit behandelt auch die Schwierigkeiten, welche durch undefinierte Rechtsbegriffe während der rechtlichen Bewertung durch einen Rechtsexperten auftreten können, und erlaubt somit auch die Analyse größerer Softwaremodelle. Die Methodik und Ergebnisse der Analyse werden den Softwarearchitekten und Rechtsexperten so präsentiert, dass die Quellen möglicher Probleme erkannt werden können und wie diese möglicherweise behoben werden können. Informationen zum Kontext eines Softwaremodells wird entweder direkt aus dem Softwaremodell gewonnen, oder wird durch den Rechtsexperten oder Softwarearchitekt in das Modell eingefügt und kann in den Ergebnissen für weitere Analyse und Untersuchungen verwendet werden.

Der Inhalt der Abschlussarbeit wird abschließend durch eine fallstudienbasierte Evaluation der Machbarkeit evaluiert. Die Evaluation beinhaltet ein Ziel zur Beurteilung der Genauigkeit der Ergebnisse und ein Ziel zur Beurteilung der Skalierbarkeit der vorgestellten Analyse in verschiedenen Szenarien. Ergebnis der Evaluation ist, dass die Ergebnisse der Analyse genau sind und die Analyse verwendet werden kann, um größere Softwaremodelle zu analysieren.

Contents

List of Figures

1. Introduction

The transformation to digital processes reaches more and more parts of people's daily life. Public health services, like COVID-19 tracing, or smart home applications run data-driven processes to analyze or use user data in other processes.

The growth in collected and processed personal data increased concern about data protection and led to legislative action, like the GDPR. Comprehensively addressing these guidelines in software, however, can be demanding and require a joint effort from both legal experts and software architects [3]. Furthermore, the legal impact of the design decisions need to be communicated to the software architect. However, the interdisciplinary communication between legal experts and software architects is prone to errors: For example, the software engineer might describe flows of data though the application, but lacks the knowledge about the legal context of those flows. A legal expert might know the different legal context a data flow might occur in, but does not have knowledge about the functionality of the software system.

The thesis will contain a method to analyze a software model regarding a predefined set of legal norms. The impact of Undefined legal terms (ULTs) on the software model will be communicated to both the software engineer working with the software model and the legal expert interpreting the results of the analysis. By this, the thesis also aims to improve the interdisciplinary communication between legal experts and software engineers in data protection assessments.

Existing approaches to improve the legal analysis of software models include the following: Agarwal et al. [1] present a model that contains conceptual information about the GDPR and compile questions about the whole system based on them. ULTs contained in the legislation are most likely also present in the questions about the system. Matulevicius et al. [12, 18] presents an approach to compare GDPR-compliant systems with the system under consideration. The approach does not specify how it handles ULTs and most likely does not output context information for the legal expert in the results. This thesis aims to show the violations of the constraints outlined in the GDPR on specific model elements and provides support for the legal analysis of ULTs, for example, due to their representation as context dependent attributes (CDAs).

This thesis is structured as follows: In Chapter 2, I will outline both legal and technical foundations required to understand and build this extension of the analysis. The next Chapter 3 the running example of the train model is presented. This model is used throughout the thesis as a demonstration model for the different algorithms and steps presented. Afterward, Chapter 4 outlines how GDPR models can be analyzed as data flow diagrams (DFDs). It outlines how CDAs, a technical representation of ULTs, impact the model and how they are analyzed efficiently. Next, Chapter 5 explains how the expanded legal assessment framework from Boltz et al. [5] and the extended data flow analysis from Hahner and Boltz et al.[4] can be extended to implement the analysis framework outlined in the previous chapter. Chapter 6 outlines the feasibility evaluation of the analysis extension, which includes an accuracy and scalability goal. Finally, Chapter 7 sums up the findings of this thesis and proposes future work that might originate from this thesis.

2. Foundations

This chapter first introduces the GDPR in Section 2.1.1, then outlining the importance of Data Protection Analysis and their legal significance in Section 2.1.2. After that, it presents undefined legal terms and their connection to the GDPR in Section 2.1.3. Section 2.2.1 introduces the GDPR-Metamodel from Boltz et al.[5] and provides an overview over the metamodel used to model a system. Furthermore, Section 2.2.2 introduces CDAs, a representation used to describe undefined legal terms. Lastly, Section 2.2.3 describes how the data flow analysis [15, 4] works and what software models are used as input.

2.1. Legal Foundations

This section outlines the legal foundations that are needed to understand the extension of the data flow analysis.

2.1.1. General Data Protection Regulation (GDPR)

The GDPR establishes a legal framework that safeguards privacy rights of individuals and governs how entities, like companies, may handle personal data. The regulation and its legal guidelines apply to software systems, demanding a joint effort from the legal and technical domain. Therefore, the GDPR defines data protection impact assessments (DPIAs) that can be conducted to investigate, whether a system adheres to the legal guidelines of the regulation.

Article 4 of the GDPR defines multiple terms that are used throughout the legislation. For example, it defines personal data as information relating to an identifiable natural person. Later articles describe rights and obligations, like article 6 that describes when processing is lawful.

2.1.2. Data Protection Analysis

A data protection analysis assesses the legal compliance of a system in the context of the GDPR. In this analysis, the system with its functionality and architecture is analyzed by a legal expert to determine whether the system is in accordance with the law.

Multiple expressions have been translated from German to English, for details on the translation see Appendix A.1. Definitions and examples in this section originate from Zippelius [20]. Usually, laws contain one or multiple legal consequences that arise when the described legal requirement is fulfilled. A legal requirement may contain multiple requisites, also called definitional elements of a rule. Interpretation from a legal expert is already required in determining whether a requisite is fulfilled, as they need to look at different articles and determine whether some laws may apply simultaneously, called cummulative conflicting norms, as specializations or not at all due to higher order legislation.

In a legal analysis, the legal expert captures the legal requirement that is described in the law and determines whether the matter of fact support the legal requirement. If so, the legal consequences apply. This process of applying the legal norms to the real world is also called subsumption. This process requires further interpretation by a legal expert: First, not all information of the matter of fact is used to determine whether a legal requirement is fulfilled or not, but may be used during the interpretation of a law or determining the severity of one case. Second, the subsumption from matter of fact to legal requirement may require further interpretation and may even be unambiguous. This may be due to an undefined term contained in the legal norm, that can be interpreted differently by different people, or a term that cannot be defined at all. An example of this problem, is the definition of personal data in Art. 4(1) GDPR that defines personal data as data from which the user is identifiable. However, the legislator has not defined what identifiable may mean in this context, so the legal expert needs to interpret the matter of fact, like a software system, to determine whether the requisite of the legal requirement is fulfilled.

To assess, whether a legal requirement is fulfilled by the matter of fact often requires specific domain knowledge, like software architecture or computer science in the context of the GDPR, is needed. Due to this, the interdisciplinary communication and collaboration between the legal and technical domains is important in assessing the compliance of a software system. Some publications [19, 17] also address the necessity and difficulties of legal assessments in the context of the GDPR.

2.1.3. Undefined Legal Terms

ULTs are entities in a legal norm, which the legislator has not clearly defined. The undefined terms encountered during the subsumption, as outlined in the previous Section 2.1.2, correspond with this definition of ULTs. Therefore, they require interpretation, for example with subsumption, from a legal expert and possibly other domain experts as well. During this process, the legal expert may try to determine what the legislator meant with the term, for example derived from the purpose of the law or from context.

The context in which the ULT is interpreted must be determined by a legal expert and other domain experts before determining whether a legal requirement is fulfilled. The context in which the ULT is interpreted is determined by the relationship and connection between different norms, as well as the general context of the situation in which the legislation applies.

In the context of the GDPR, ULTs often need to be interpreted in the context of the system by a domain expert. Furthermore, with a growing software system, the context in which a ULT may be interpreted grows, slowing the legal analysis and increasing complexity of it. To assess the compliance of a system, legal experts and software architects need to work together and communicate their domain specific knowledge to build the context in which ULTs are interpreted. As an ULT might affect multiple different data flows in different legal contexts, the number of occurrences of an ULT in the legal description of the system may be high. This increases the need for efficient and accurate communication between legal experts and software architects.

2.2. Technical Foundations

This section outlines the technical foundations required to understand how the baseline data flow analysis works and how the GDPR metamodel is structured.

2.2.1. GDPR-Metamodel

One approach to represent a software model focused on storing and presenting information relevant for a data protection analysis is the GDPR-Metamodel presented by Boltz et al. [5]. An overview over the GDPR-Metamodel [5] can be seen in Figure 2.1. In general, *Processing* elements describe the flow of data though the application: *Collecting* elements are sources of data, while *Storing* elements describe a sink of data. Data is transferred between these sources and sinks via *Usage* or *Transferring* elements that describe where data in the system flows. Each *Processing* element serves a *Purpose* and is governed by a given *Legal Basis*. *Legal Basis* elements include the *Consent* or a *Contract*. The elements in the model are connected via *Data* elements that can be *Personal Data* and reference a *Natural Person*. The parties in the GDPR-Metamodel are represented by *Role* elements, which can be either *Controller*, *Third Party* or a *Natural Person*.

The GDPR-Metamodel can be transformed into a data flow diagram (DFD), as the work by Boltz et al. [5] shows. *Processing*-Elements will be converted to DFD *Node*s. Each node will represent one processing step with incoming and outgoing data.

2.2.2. Context Dependent Attribute Model

In Addition to the description of the system, the GDPR model as outlined in Section 2.2.1 also contains a model to describe CDAs. The CDA model is built to model ULTs and relate them to GDPR model elements. CDA are attributes of GDPR elements that are defined for a given context. They apply a value of a given type to an element, if the context of the element is matched by the specified context.

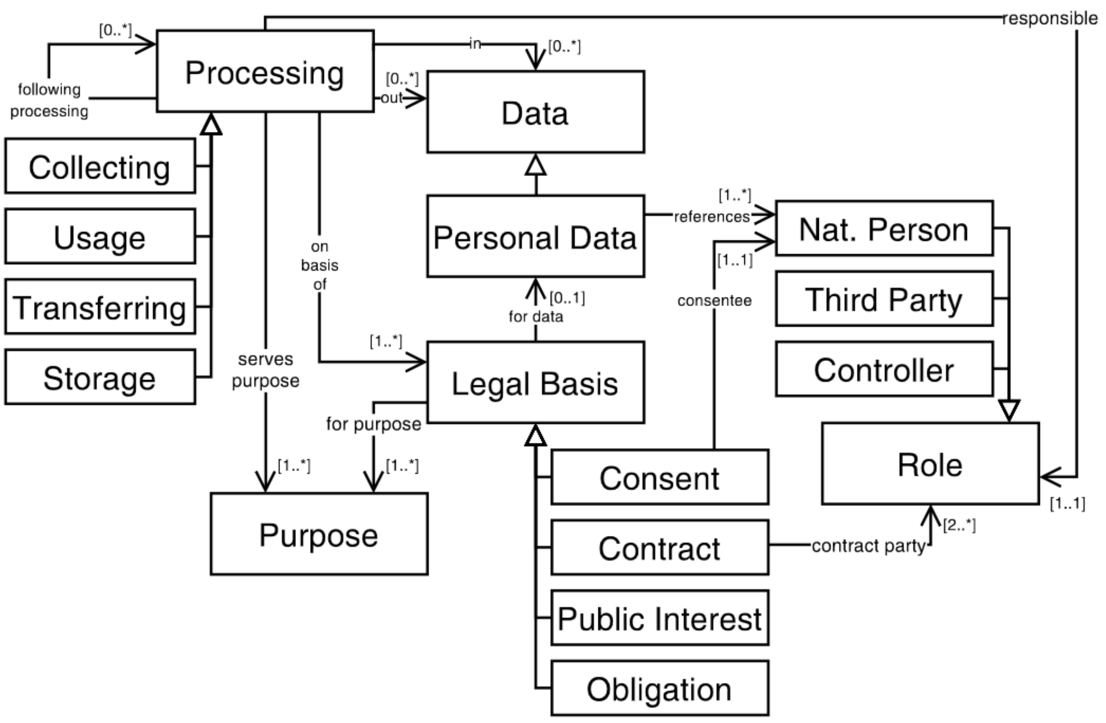


Figure 2.1.: Except from the GDPR-Metamodel (taken from [5])

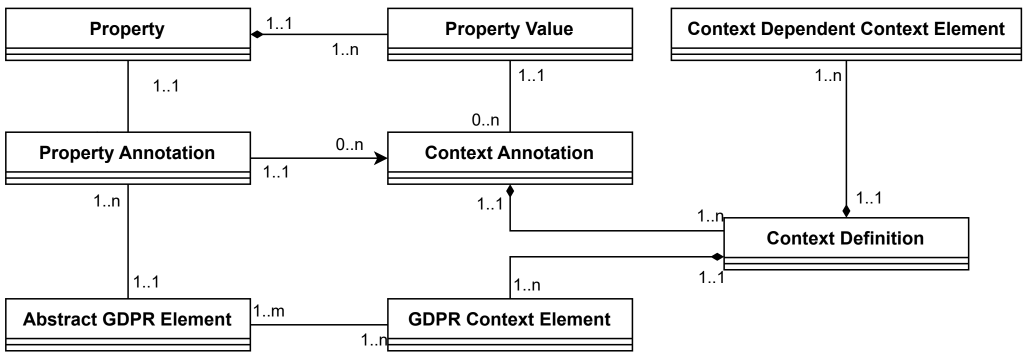


Figure 2.2.: Metamodel of context dependent attributes

The CDA model is structured as follows: A *Property* represents an attribute type that may be applied to an element. An example might be the identifiability of a natural person. It contains a list of *Property Value*s that describe the different attribute values the *Property* can have. In the example of the *Property* identifiability, the *Property Value*s might be true or false.

A *Property Annotation* describes the application of a context dependent attribute to an element. It contains a reference to a property and the annotated element. Furthermore, it might contain a list of *Context Annotations*. A *Context Annotation* references a *Property Value* that is applied if the referenced *Context Definition* is present.

A *Context Definition* describes one possible context in which a context dependent attribute may have a value. It contains a list of GDPR elements and other property annotations. An example of a *Context Definition* might be the identifiability of a user in the context of registering an account.

2.2.3. Data Flow Analysis

The architecture-based confidentiality analysis outlined by Schwickerath et al. [15] and Boltz et al. [4] uses model instances of a software system to analyze security-related attributes of the system.

The confidentiality analysis takes a DFD as input, that describes the exchange and processing of data in a software system. This model is presented by Boltz et al. [4] and is based on research by Seifermann et al. [16] and DeMarco et al. [6]. The representation of a system as a DFD is used to assess security-related properties of a software system, like confidentiality. Nodes are representing a processing, storing or exchange of data in a software system, whereas flows connect the different nodes and denote a flow from one node to another. Attributes of data are represented by label types and their values. Node labels describe the labels applied to a node, like the geolocation of the deployment or role in an access control scheme. Data labels describe attributes of the data that is passed between nodes, for example, the encryption status of a piece of data.

After the models are loaded, the analysis extracts the data flows from the model in form of transpose flow graphs (TFGs). A TFG represents a single flow of data though the system and is uniquely identified by its sink and contains a set of nodes and their connecting flows. Each node in a TFG contains a list of incoming and outgoing data labels, node labels and a list of previous nodes.

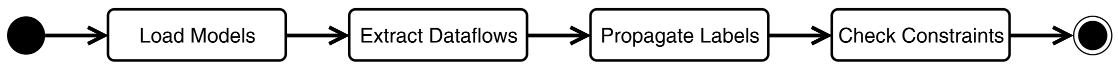


Figure 2.3.: Workflow of the data flow analysis as presented in [15]

After determining the TFGs in the model, the annotated information in the behavior of the nodes is propagated. This is accomplished by recursively determining the incoming flow to a node by evaluating its previous vertices. After the propagation is finished, information about the data labels present at the node are available. Following the propagation, this information can be used to query the data and node labels of nodes in the TFG and determine whether arbitrary conditions are met. For example, this step can be used to determine whether design requirements, like the storing of data only inside an EU datacenter, or check an implemented access control scheme. This information is then presented to a user and used to change the software system.

3. Running Example

This chapter introduces a running example that is referenced and used throughout the thesis to explain the concepts contained in this thesis. The running example models a *train company* that provides functionality for users to book train tickets after they have registered with the *train company*. Additionally, the *train company* uses a *statistics provider* that creates analytics for the *train company*. The data transfered in this process is pseudonymized beforehand. During the process of creating analytics, the *statistics provider* also forwards data to a *marketing service* that creates advertisements to grow the customer base. The resulting model from this description, consists of three different responsibilities: The *train company*, that provides account and booking services, a *statistics provider* that provides statistics for the train company on bookings and a *marketing service*. During a booking, pseudonymized data is passed from the train company to the statistic provider, which in turn uses the marketing provider to generate marketing information.

As presented in Section 2.1.1, Article 4(1) defines personal data as data from which the corresponding natural person is identifiable. Furthermore, Article 2 of the GDPR specifies, that the regulation applies to matters with personal data. Therefore, if the user is not identifiable, the GDPR may not apply in the given context. As **identifiability** is an undefined legal term, as discussed in Section 2.1.3, the running example models a CDA for **identifiability** on the user.

Determining the **identifiability** of the user in the context of the *train company* is easy, as the *train company* must collect and persist identifiable information about the user for the purpose of fulfilling their contractual obligations and accounting, so the CDA models the **identifiability** as `true` in the context of the train company.

In the other contexts of the *statistics provider* and the *marketing service*, however, it remains unclear, whether the user is identifiable or not.

Another CDA is modelled as the **transparency** of the marketing purpose on the data provided in the booking process. The *Property Value* of the CDA also remains uncertain, as a legal expert needs to investigate further whether it is transparent to the user.

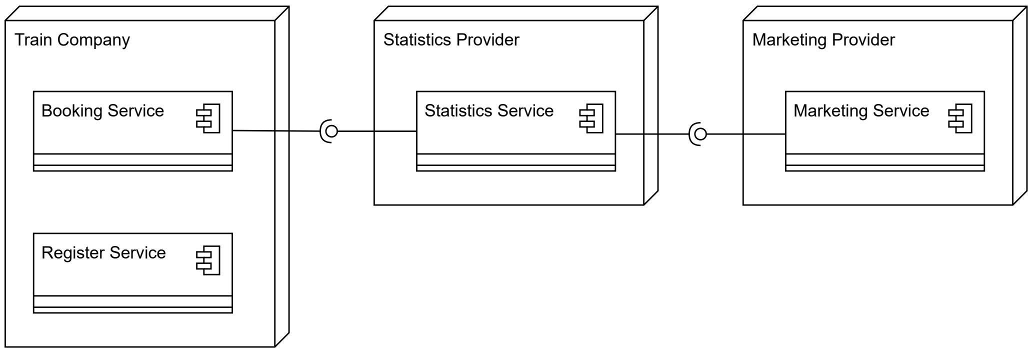


Figure 3.1.: Simplified component and deployment diagram of the train model running example

The *train company* handles the account creation between the user and the *train company*. For this reason, an account contract between the *train company* and customer is made that is used to collect customer data and store it. This allows the *train company* to collect customer data and store it for the purpose of creating an account. In Addition, the customer gives consent to process the stored data for the purpose of creating statistics. For that reason, the data is pseudonymized and transferred to the *statistics provider*.

In regard to the CDA of **identifiability**, the user is identifiable, when the data without pseudonymization is available. Otherwise, the value of the CDA is not defined. The CDA of **transparency** is not relevant in the context of the *train company*.

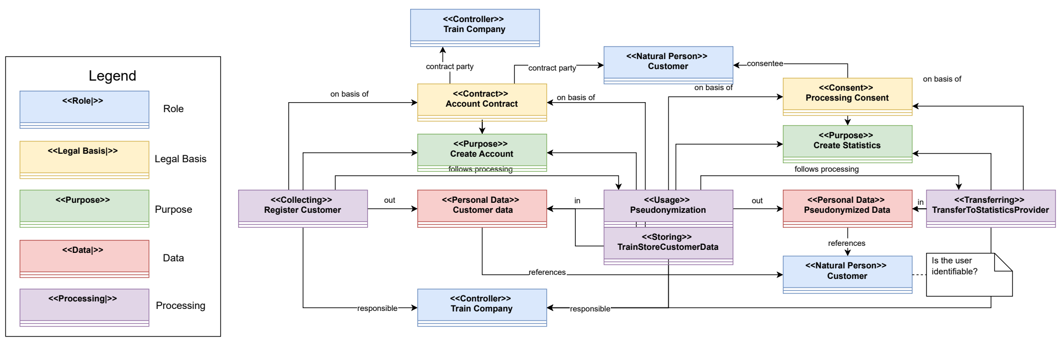

Legend
<<Role|>> Role
<<Legal Basis|>> Legal Basis
<<Purpose>> Purpose
<<Data|>> Data
<<Processing|>> Processing
<<Controller>> Train Company
contract party
contract party
<<Natural Person>> Customer
consentee
on basis of
<<Contract>> Account Contract
on basis of
on basis of
<<Consent>> Processing Consent
on basis of
<<Purpose>> Create Account
follows processing
<<Purpose>> Create Statistics
follows processing
<<Collecting>> Register Customer
out
<<Personal Data>> Customer data
in
<<Usage>> Pseudonymization
out
<<Personal Data>> Pseudonymized Data
in
<<Transferring>> TransferToStatisticsProvider
<<Storing>> TrainStoreCustomerData
references
references
<<Natural Person>> Customer
Is the user identifiable?
responsible
<<Controller>> Train Company
responsible

Figure 3.2.: GDPR Model of the complete Train Example Model; Train Company Responsibility

The *statistics provider* receives pseudonymized data from users originating from the *train company* for the purpose of creating statistics. This purpose is covered by the consent of the user to use their data for creating statistics. The *statistics provider* uses this data to calculate statistics and store the data. Furthermore, the data is further transferred to a *marketing provider* for advice to grow their customer base and increase profits.

The CDA of **identifiability** of the customer regarding the pseudonymized data is not defined. In Addition, the **transparency** of the processing purpose is not relevant for this context

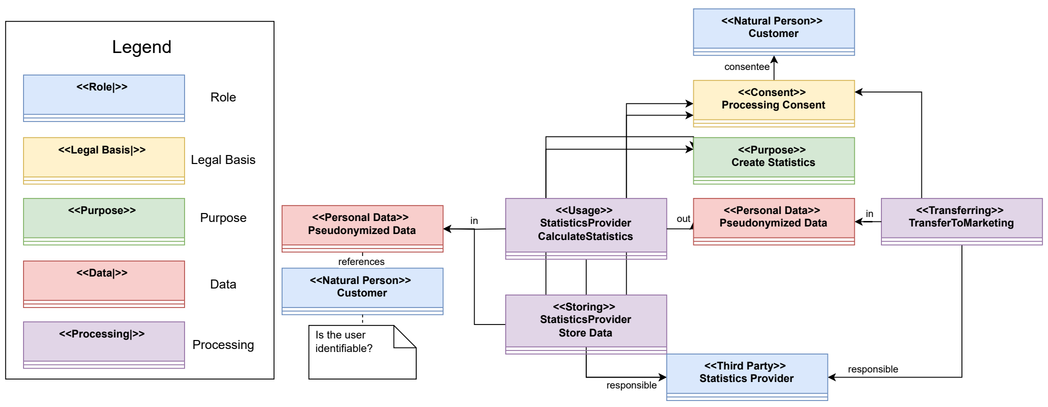

Legend
<<Role|>>
Role
<<Legal Basis|>>
Legal Basis
<<Purpose>>
Purpose
<<Data|>>
Data
<<Processing|>>
Processing
<<Natural Person>> Customer
consentee
<<Consent>> Processing Consent
<<Purpose>> Create Statistics
<<Personal Data>> Pseudonymized Data
in
<<Usage>> StatisticsProvider CalculateStatistics
out
<<Personal Data>> Pseudonymized Data
in
<<Transferring>> TransferToMarketing
references
<<Natural Person>> Customer
Is the user identifiable?
<<Storing>> StatisticsProvider Store Data
responsible
<<Third Party>> Statistics Provider
responsible

Figure 3.3.: GDPR Model of the complete Train Example Model; Statistics Provider Responsibility

The *marketing provider* receives data from the *statistics provider* for the *Purpose* of creating advertisements. The *Legal Basis* of this *Purpose* is set as the processing consent given by the user. On this basis the pseudonymized data, which references the customer, is used in creating marketing and storing relevant information.

Both CDAs are relevant here: The **identifiability** of the user in this context is not defined, similar to the statistics provider. However, the **transparency** is relevant for this context: As the purpose of creating advertisements may not be transparent to the user, the consent may not cover this purpose. For that reason, the purpose is annotated by the **transparency** CDA. However, whether the purpose is transparent to the user has not been covered by the legal analysis so far.

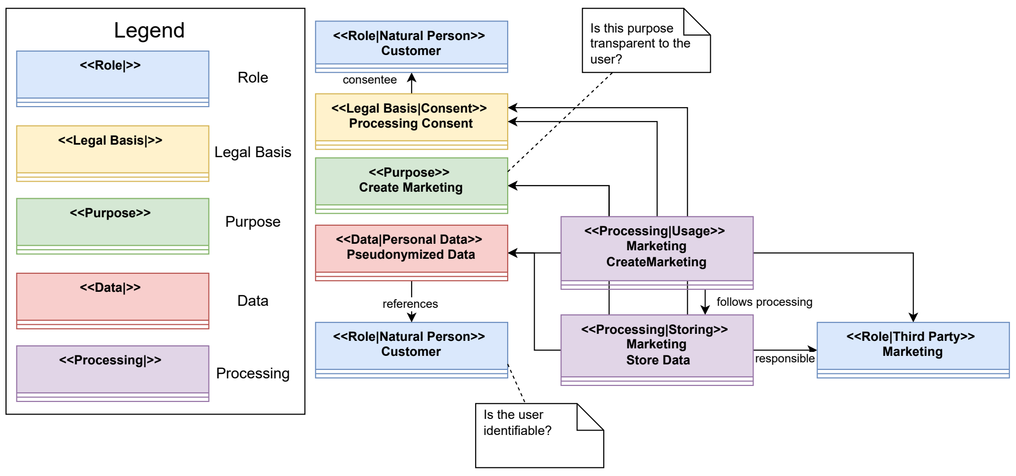

Legend
<<Role|>>
Role
<<Legal Basis|>>
Legal Basis
<<Purpose>>
Purpose
<<Data|>>
Data
<<Processing|>>
Processing
<<Role|Natural Person>> Customer
consentee
<<Legal Basis|Consent>> Processing Consent
<<Purpose>> Create Marketing
<<Data|Personal Data>> Pseudonymized Data
references
<<Role|Natural Person>> Customer
Is this purpose transparent to the user?
<<Processing|Usage>> Marketing CreateMarketing
follows processing
<<Processing|Storing>> Marketing Store Data
responsible
<<Role|Third Party>> Marketing
Is the user identifiable?

Figure 3.4.: GDPR Model of the complete Train Example Model; Marketing Responsibility

Lastly, one might define two constraints on the model, following some preliminary investigation by a legal expert: First, the *marketing provider* may not receive data in which the user is identifiable. Secondly, the processing of data with untransparent purposes is forbidden.

4. Analyzing GDPR models with DFDs

This chapter outlines how the representation of a system in the GDPR model can be analyzed by transforming it into a DFD model. The first Section 4.1 will outline how GDPR models can be used as input for the data flow analysis. In Section 4.2 I will outline the importance of different responsibilities in the GDPR-Metamodel and how they may be implemented with TFGs. Lastly, Section 4.3 describes how context dependent attributes can be evaluated in the context of TFGs.

4.1. Handling GDPR Models as Input

As outlined in Section 2.2.1, instances of the GDPR model can be converted into a DFD model. For elements that affect the data that is passed though the system, the context information only needs to be set once: When a data element that is not referenced in the input data of a node is encountered, the relevant context information is set on the data. This is because the conversion creates the following elements so that it will forward this information, if incoming and outgoing pins exist at the node with the same name.

For the purpose of resolving CDAs, the context of an GDPR element can be defined as:

- The element itself
- All referenced purposes of the element
- All referenced legal contexts of the element
- All incoming data of the element
- All outgoing data of the element
- All referenced natural persons referenced by data in the context

This information is added to each node as additional information and represents the technical context as required for subsumption, introduced in Section 2.1.2. The context of an element is required to determine whether a *Context Definition* can be fulfilled for a node or not. The context for an GDPR or corresponding DFD element can be formalized via the $context$ function.

To allow the legal expert more flexibility in expressing CDAs, not all information of a CDA needs to be set. A **Fully Resolvable CDA** is a CDA which has *Context Annotation*s that are applicable for each context. Therefore, the value of the CDA is clearly defined for every context in the model. On the other hand, an **Uncertain CDA** is a CDA which value is not defined in at least one context, resulting in more effort and special cases during the analysis.

Lastly, the nodes should also store whether a CDA has been applied to it and the corresponding *Property Value* of the CDA has been set. The resulting TFGs will also store the CDA that may apply to nodes within it.

The CDA model can be used to annotate ULTs in a GDPR model. This information is then used in the analysis of the model to produce results from the multiple different model variations described though CDAs.

The CDA model assumes that all possible values of an ULT are known and can be modelled in a reasonable amount of *Property Value*s. Then, a CDA can be used to model an ULT as the CDA is able to model all different variations or possibilities in the value of the ULT. Each *Property Value* of a CDA then reflects a possible value the ULT can assume. If some values of the ULT can be excluded by the legal expert, they are able to express this using *Context Definition*s in the CDA model.

*Property*s in the GDPR model and label types in DFD are equivalent, because the structure of the *Property* and *Label Type* as well as the *Property Value* and *Label* are equivalent. Therefore, *Property*s and *Label Type*s, and *Label*s and *Property Value*s can be used interchangeably.

Finally, a CDA can be expressed in a three-tuple $(element, type, value)$: The tuple-element $element$ describes the annotated GDPR element to which the CDA is applied. The element $type$ describes the GDPR *Property* or its DFD *Label Type* equivalent. Finally, the element $value$ describes the GDPR *Property Value* or DFD *Label* that is applied to the element if the CDA is applied.

In Addition to saving the applied CDAs for a node and a TFG, I also save the relevant CDAs that may apply to a TFG and node. This allows TFGs to later evaluate their state independently on a subset of all present CDAs. A CDA is relevant for a node, when the node has the targeted element of the CDA in its context. Therefore, a CDA is relevant for a TFG, when the TFG contains an element, for which the CDA is relevant.

Running Example

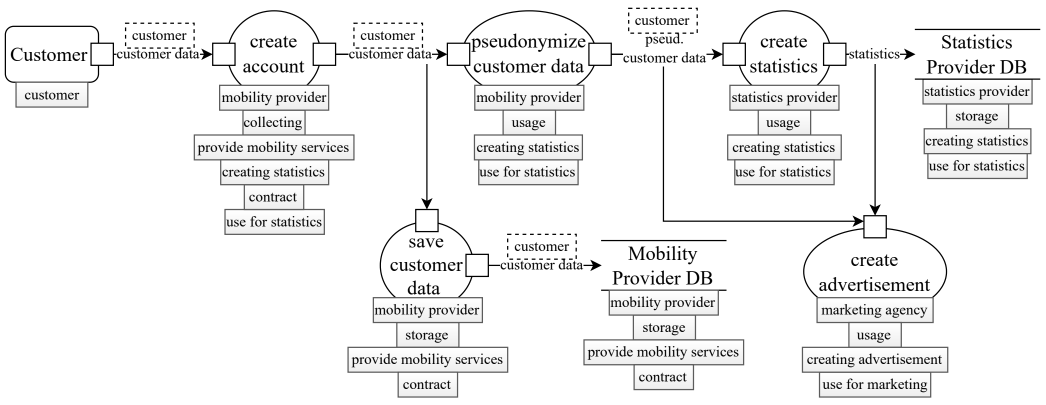


Figure 4.1.: Converted Example Model

The Figure 4.1 shows the converted example train model from Chapter 3.

The GDPR context of each DFD element is stored in the node and data labels of the corresponding node, as shown above. For example, the `Pseudonymization` DFD node has the following node labels as GDPR context: The node labels `LegalBasis.AccountContract` and `LegalBasis.ProcessingConsent` are set, as the processing element uses them as legal basis. Additionally, the node labels `Purpose.CreateAccount` and `Purpose.CreateStatistics` are set, as the model sets them as the purpose of the processing element. Furthermore, the data label `PseudonymizedData.Role.Customer` is set at this node, as the GDPR data element is not set as input of the element as well.

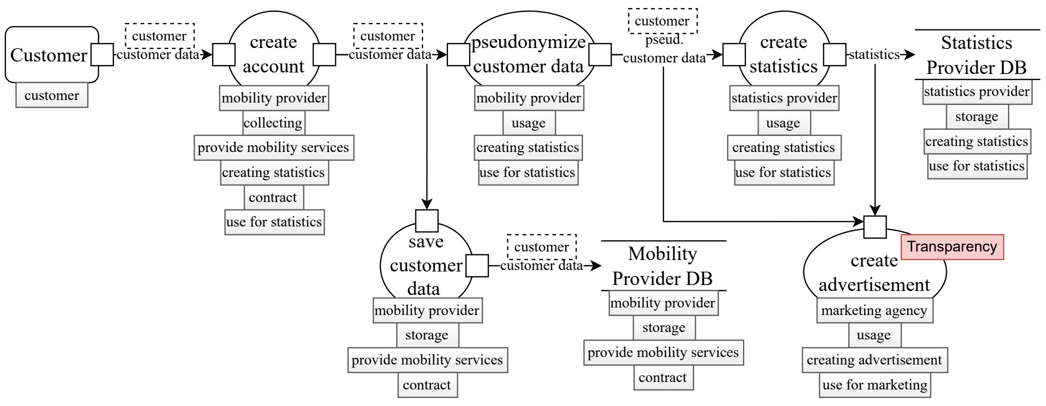


Figure 4.2.: Example Model with Annotated CDA

For illustrating the process of determining the relevancy of a CDA, we determine the relevant elements for the CDA of **transparency** from the running example from Chapter 3. The

Figure 4.2 shows the previous DFD, with the relevant elements for the CDA of **transparency**. Only the element *create advertisement* is marked, as it is the only element in the DFD that uses the purpose *use for marketing*, which the CDA of **transparency** targets.

4.2. Separating Context and Responsibility

As responsibilities of processing within a software system might change, it is important that legal experts might look at a partial view of the system, as it is required for the subsumption as outlined in Section 2.1.2. To accommodate these requirements, a TFG can be split at each transition between different responsible roles. These smaller TFGs will be called **responsibility context-aware TFGs**.

Separating the TFGs into smaller **responsibility context-aware TFGs** is helpful in two ways: First, the legal analysis of these smaller TFGs is easier, as it contains fewer elements with a shared responsible role. This allows the automated analysis to find problematic contexts and present them to the legal expert for further analysis. Furthermore, as the TFG is smaller, the context, in which the regulations of the GDPR may apply, is reduced. This helps legal experts to determine and interpret the ULTs present in regulation using CDAs, as outlined in Section 2.1.2 and Section 2.2.1. Observations, made during the analysis and interpretation of the TFG in the given context, can be used to refine the modelled CDAs and analyze the model further. Additionally, the overview over the system is preserved by analyzing the complete TFG as well.

For the formalization of the responsibility contexts, I define the following:

For a *Processing* GDPR element, I define the function $responsible$ that determines the roles that are responsible for the processing element.

Using this function, I define an equivalence relation $\Leftrightarrow_{\text{ResonsibilityContext}}$ that determines whether elements are in the same responsibility context. Therefore, the equivalence relation is defined for elements $element$ and $element'$:

$$element \Leftrightarrow_{\text{ResonsibilityContext}} \Leftrightarrow element' \text{ exactly when } responsible(e) \cap responsible(e') \neq \emptyset$$

Therefore, two GDPR *Processing* elements are in the same responsibility context, when they share a responsible role for their processing. The equivalence classes of this equivalence relation for each TFG represent the responsibility-context aware TFGs as outlined above.

Running example

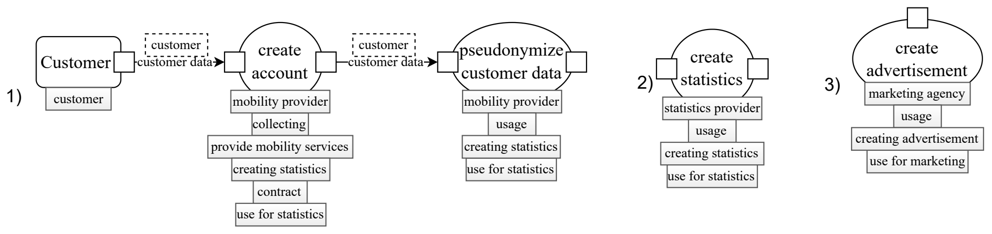


Figure 4.3.: Responsibility TFGs in the example model

The explanation in Section 3 follows the different responsibility boundaries of the system. The different sections of the model can be seen in figures 3.2, 3.3 and 3.4. Looking at the TFG starting from the train company and ending at storing the data at marketing, the resulting TFGs will be the following, represented in Figure 4.3: The first TFG in Figure 4.3 includes the register, pseudonymization and transfer to statistic provider nodes. The second TFG in Figure 4.3 includes the *create statistics* node. The third TFG in Figure 4.3 includes the *create advertisement* node. Finally, the last TFG will include all elements contained in the TFG.

4.3. Evaluating Context Dependent Attributes

The evaluation of **Fully Resolvable CDA** at a node in a TFG ,referencing a GDPR element, follows the following principle: For a CDA find the *Context Annotation* from which at least one *Context Definition* is applicable. A *Context Definition* is applicable, when the context of the GDPR element fulfills the required context of the *Context Definition*. Using the applicable *Context Annotation*, the node in the TFG is modified using the referenced *Property Value* and its *Property* parent. As DFD models are analyzable by the data flow analysis presented in Section 2.2.3, the annotated values from CDA can be used in the constraints as well.

Depending on the element to which the CDA is applied, it will create different model elements:

- Processing GDPR element: DFD Node label
- Legal Basis GDPR element: DFD Node label
- Purpose GDPR element: DFD Node label
- Data GDPR element: DFD Data label
- Natural Person GDPR element: DFD Data label

The presented approach for resolving **Uncertain CDAs** follows the principles of design space exploration that has already been used to explore different variations of software models [7, 11]. Therefore, **Uncertain CDAs**, that cannot be resolved in the context of one node, can be resolved by creating TFGs with the possible *Property Value*s of the CDA. This approach creates multiple copies of the same TFG with different *Property Values* the CDA might have. These *Property Value*s are given as the *Property Value*s from the *Property* the CDA has. If some *Property Value*s have an *Context Annotation* attached, but it failed to match the context of the node the CDA is currently applied to, these *Property Value*s will not appear as values in the created TFGs.

A **Context Attribute State** can be defined as a set of applied CDAs. As multiple CDAs can affect elements in a TFG, each TFG has an assigned **Context Attribute State** it reflects. The results from the design space exploration caused by **Uncertain CDAs** are included in the **Context Attribute State**s as well. The different TFGs created by evaluating CDAs can be differentiated using the defined **Context Attribute State**. The state of each TFG can be used during the review of the results by a legal expert to see the consequences of the design space exploration caused by **Uncertain CDAs**.

Furthermore, the resulting information from the analysis can be used by a legal expert to determine whether the *Property Value* is, in fact, applied to the element. Then the legal expert might insert a *Context Definition* that sets the value of the CDA. Additionally, this approach reduces the amount of states that the different CDAs create and therefore the legal expert need to evaluate. This is because some **Context Attribute State**s do not cause any violations and therefore are not relevant for the legal evaluation of the software model.

As CDAs can reference other CDAs as *Context Definition*s, CDAs must be evaluated in the correct order to avoid conflicts or high evaluation cost. To properly evaluate multiple CDAs in the correct order, a half-order must be defined: For two CDA x and y with corresponding referenced other CDAs C_x and C_y a half-order can be defined as:

$$x \leq y \Leftrightarrow y \in C_x \text{ and } y \leq x \Leftrightarrow x \in C_y$$

If the application of CDAs follows the half-order, each CDA can be evaluated sequentially and no conflict in the evaluation of the CDAs will occur. If there are CDAs that depend on each another, the evaluation of the different CDAs cannot follow the half-order and thus the CDAs cannot be evaluated properly.

Running example

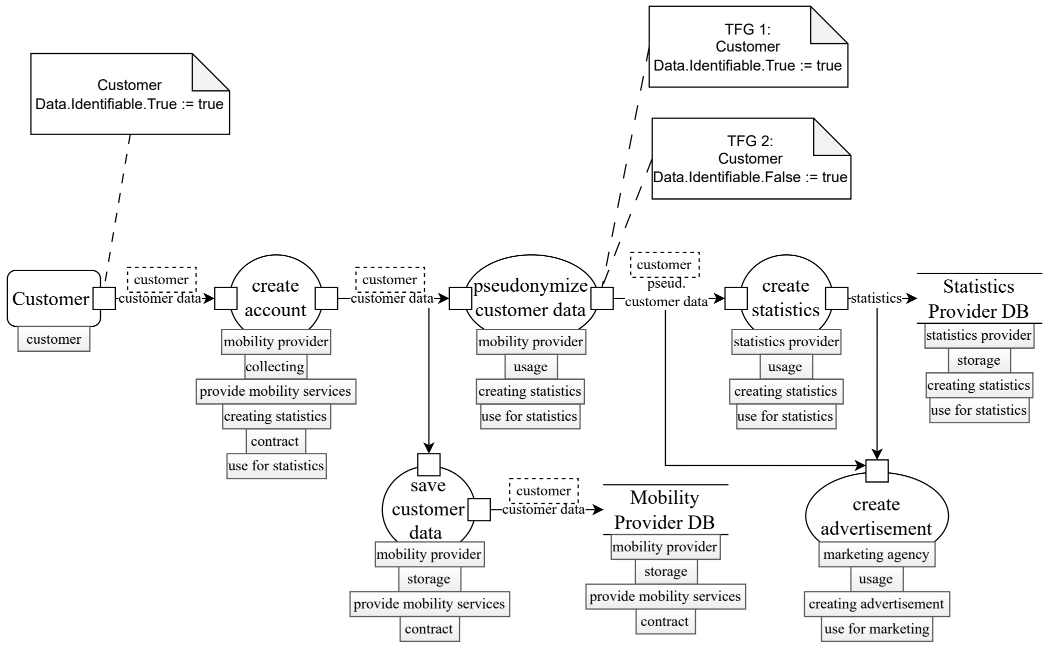


Figure 4.4.: Applied CDA to a TFG of the Example Model

Using the CDA of **identifiablity** as an example, we apply the CDA to the TFG: As outlined in Chapter 3, the user is identifiable from the unpseudonymized user data that they provide for the creation of an account. Therefore, the CDA is modelled to set **identifiability** to true, when the unpseudonymized user data is in the context of an element. Thus, we set the **identifiablity** on the data that originates from the user.

However, the legal expert has not determined whether the user is **identifiable** in the context in which the pseudonymized data is available. Therefore, the CDA is modelled to set **identifiablity**, but does not specify to which value. This behavior is archived by creating a *Property Annotation* with the *Property* **Identifiablity**, but not specifying any *Context Annotation*s. The CDA will be categorized as an **Uncertain CDA** due to this. During the evaluation of the CDA, each TFG that has the pseudonymized customer data in its context will result in two newly created differing TFG, created by the design space exploration: One TFG that sets the **identifiablity** of the user to `True` at the *pseudonymize customer data* node, the other will set the **identifiablity** to `False`.

5. Analysis

This chapter describes the main contribution of this thesis: The adaptation of the legal assessment framework presented by Boltz et al. [3] is outlined in Section 5.1. The Section 5.2 explains how the data flow analysis presented by Schwickerath et al. [15] and Boltz et al. [4] can be extended to allow the analysis in the context of the GDPR. Section 5.3 specifically outlines how CDAs are resolved by the analysis extension. Lastly, the Section 5.4 describes how the results of the analysis are presented to legal experts to support them in their data protection assessment of the software system.

5.1. Legal Assessment Framework

The legal assessment framework presented in Figure 5.1 is adapted and continued from Boltz et al. [3] and aims to support the proposed workflow with the described automated analysis. It aims to support the legal expert in the subsumption and interpretation of legislature in regard to the GDPR. As the GDPR also mandates data protection impact assessments, the proposed assessment framework can be used during them as a supporting tool as well. It also helps the legal expert to gain an understanding about the software system in question, as the proposed assessment framework aims to improve the interdisciplinary communication between legal experts and software architects.

The legal assessment workflow begins with a software architect modelling the software application using architecture description languages, like the Palladio Component Model [13], or DFDs [6]. This model is afterward transformed as outlined by Boltz et al. [3]. The resulting model is then passed to the legal expert, who adds legal information, like purpose or legal bases, to the processing elements in the GDPR model.

The GDPR model can be converted to a DFD after adding legal information and analyzed by the data flow analysis presented in [4]. Resulting violations are presented to the software architect and the system is adjusted accordingly.

The contribution extends the presented approach by Boltz et al. [3] and adds another cycle: By resolving CDAs, the model is further enriched by information about the legal context of the software system. Possible uncertainties about details in the design of the software system can be expressed with CDAs and evaluated how they will impact the legal conformity of the system. The results of the analysis outlined further in this section can be communicated to the software architect, if changes in the software model are required. They may also originate from assumptions made about the legal context of the system and allow the legal expert to do further in-depth research about the CDA.

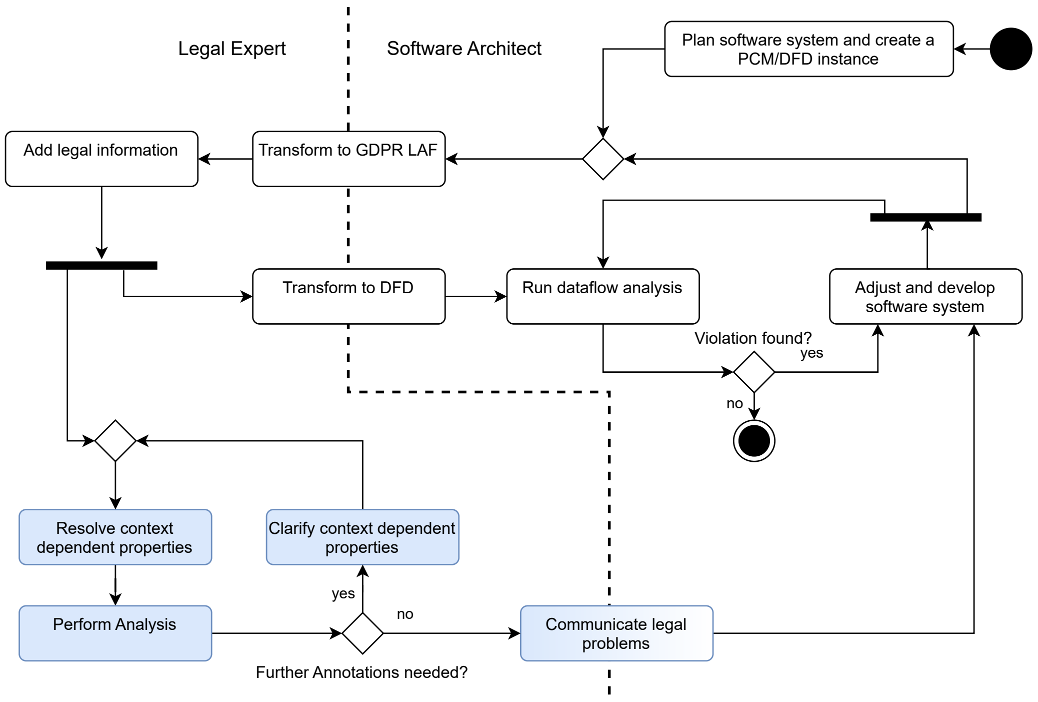

Legal Expert
Software Architect
Plan software system and create a PCM/DFD instance
Add legal information
Transform to GDPR LAF
Transform to DFD
Run dataflow analysis
Adjust and develop software system
Violation found?
yes
no
Resolve context dependent properties
Clarify context dependent properties
Perform Analysis
yes
no
Communicate legal problems
Further Annotations needed?

Figure 5.1.: Legal Assessment Model Workflow expanded from [3]. Modified elements to facilitate the extension are marked in blue

5.2. Extending the Data Flow Analysis

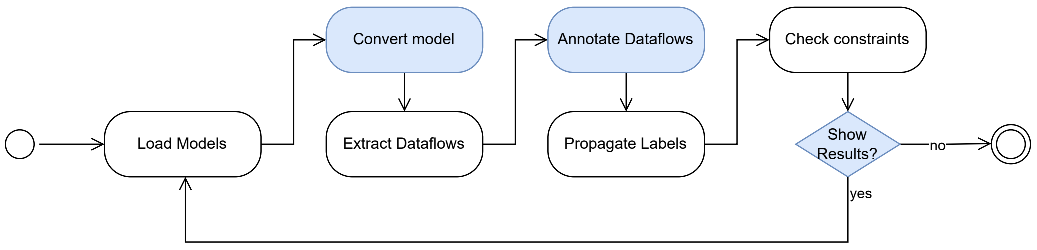


Figure 5.2.: Updated analysis process (adopted from [4, 15]). Modified elements to facilitate the extension are marked in blue

The updated analysis process outlined in Figure 5.2 begins with loading the *Eclipse Modelling Framework (EMF)* GDPR model. The model includes the description of the software system, contained in a `.gdpr` file, and CDAs contained in a `.contextproperties` file. After the models are loaded, they are converted to a DFD model using the converter outlined by Boltz et al. [3]. During this process, the context of the resulting DFD elements are also determined via the definition in Section 4.1 and based on the design outlined in Section 2.2.1. The extended DFD vertex in Figure 5.3 also stores this context.

Furthermore, during this stage, the CDAs contained in the model are parsed into `Context Dependent Attribute Scenario` and `Context Dependent Attribute Source` objects. These classes contain resolved information about the present CDA and describe their values, and their structure is shown in Figure 5.3.

A `Context Dependent Attribute Source` represents an annotation on an GDPR element that a CDA is applied. Important information of the `Context Dependent Attribute Source` is the annotated GDPR element and the GDPR *Property* it affects. The information can be derived from the *Property Annotation* element of the GDPR model.

Additionally, they contain a list of possible `ContextDependentAttributeScenarios` which contain the different *Property Value*s the CDA might have in a given context. For that, it needs to save the *Context Definition*s that need to be fulfilled to apply the referenced *Property Value*.

For **Fully Resolvable CDAs**, all `ContextDependentAttributeScenarios` are created from *Context Definition*s. Furthermore, for **Uncertain CDAs**, additional scenarios for resolving uncertain *Property Value*s of CDA are inserted. This is achieved by comparing the covered *Property Value*s of each *Context Definition* and inserting a `ContextDependentAttributeScenario` for each remaining*Property Value*. This follows the principle to evaluate CDAs as outlined in Section 4.3.

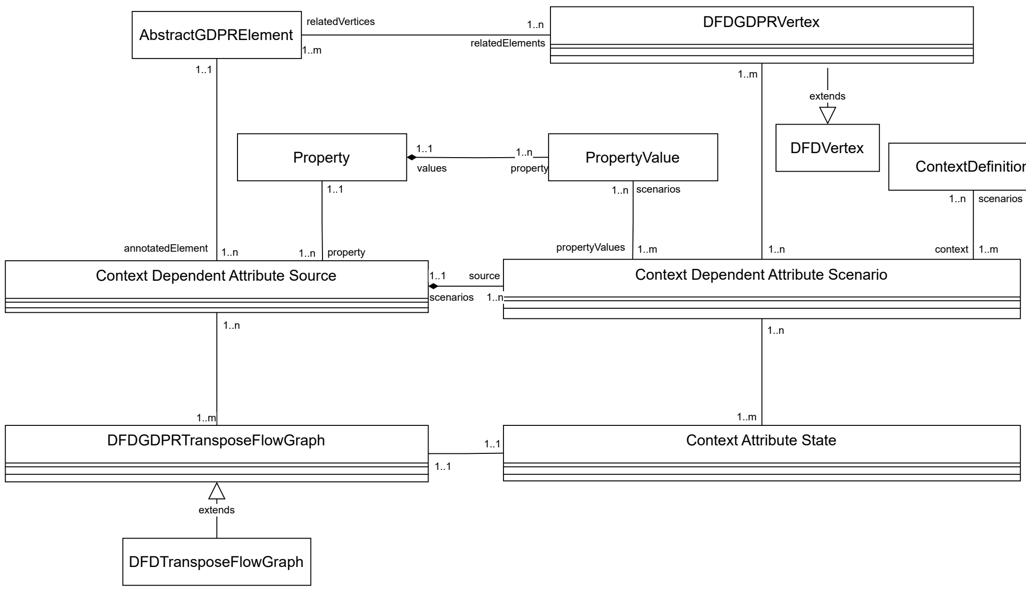


Figure 5.3.: UML Class Diagram of Context Dependent Attribute Source and Scenario

After the DFD model is created, the analysis extension extracts data flows in the form of TFGs as usual [4, 15]. Before the label propagation is started, another step first annotated the TFGs with information from the GDPR model.

A concrete application of a CDA is represented by one or multiple `Context Dependent Attribute Scenarios` and is saved in the TFG and the node as well. In this stage, for each TFG the CDAs are evaluated as well, according to the algorithm described later in this chapter. This will create multiple TFGs in accordance to the description in Section 4.3.

After the data flows in form of TFGs are annotated, they are evaluated with the label propagation algorithm as outlined in the base analysis [4, 15]. The label propagation algorithm does not need to be modified, as all behavior is encoded into the behavior of the DFD model elements the vertices of the TFGs refer to. Additionally, the step of checking constraints and finding violations is also not impacted by the extension of the analysis.

The presentation of the results however was extended in comparison to the base analysis and is outlined in Section 5.4 and better allows both legal experts and software architects to interpret the results from the analysis.

5.3. Resolving Context Dependent Attributes

Implementing the resolving of CDAs uses the `Context Dependent Attribute Source` and `Context Dependent Attribute Scenarios` that were created like outlined in Section 5.2.

The Algorithm 1 describes in detail how a list of CDAs in the form of `Context Dependent Attribute Sources` are evaluated for a TFG. The algorithm takes a list of `Context Dependent Attribute Sources` as input that have been determined to be applicable for the TFG, as outlined in Section 5.2 and Section 4.3. The algorithm will return a list of TFGs that have been determined via design space exploration and reflect the different possibilities of the CDAs affecting the TFG.

First, in line 2, the algorithm creates a list of possible states the given TFG might have. The possible states of the TFG is given as the cartesian product of all different `Context Dependent Attribute Scenarios` that are children of the given list of relevant `Context Dependent Attribute Sources`. The differing handling of **Fully Resolvable CDA** and **Uncertain CDA** is done during the creation of the `Context Dependent Attribute Sources` and `Scenarios`, therefore no additional effort is needed during the resolving.

For each possible state, the given TFG is copied in line 4. After that, each `Context Dependent Attribute Scenario` that is contained in the state is checked and applied to the TFG. While the states only contain `Context Dependent Attribute Sources` that apply to a node present in the TFG, it may contain `Context Dependent Attribute Scenarios` that may not apply to the node at all. This, for example, may be caused by CDAs that require a context that is not present at the element in the TFG. Therefore, these `Context Dependent Attribute Scenarios` will not be applied, and will instead add another states that apply *Property Value*s that are not applied by the `Context Dependent Attribute Scenario`. The check is performed in lines 8 and following. If the `Context Dependent Attribute Source` and `Context Dependent Attribute Scenario` is applicable, the algorithm will apply it to the current TFG, as outlined in Section 4.3. If the `Context Dependent Attribute Source` annotates a *Role* or

Natural Person element, the vertex that is targeted by the `Context Dependent Attribute Source` is replaced with a copy that sets the *Property Value*s in an DFD *Assignment* from the scenario and saved in the current TFG. The modification is described by the Algorithm 1 in lines 12 and following. Else, the vertex that is targeted by the `Context Dependent Attribute Source` is replaced with a copy that sets the *Property Value*s as an DFD *(Node) Label* from the scenario and saved in the current TFG. The modification is described by the Algorithm 1 in lines 15 and following. After all `Context Dependent Attribute Scenarios` are applied, the modified TFG is saved with its corresponding state in the list of possible resulting TFGs.

Algorithm 1 Algorithm for resolving CDA

```
Require: List of context dependent attribute sources $CDA_{\textbf{source}}$, Data flow TFG $tfg$
Ensure: List of transpose flow graphs $TFG$
 1: $TFG$ → empty list of TFGs
 2: $States$ → create possible states from $CDA_{\text{source}}$
 3: for $state \in States$ do
 4:     $currentTFG \leftarrow tfg$
 5:     for $scenario \in state$ do
 6:         $source \leftarrow$ get source of $scenario$
 7:         $vertex \leftarrow$ get vertex to which $source$ applies
 8:         if $scenario$ is not applicable to $vertex$ then
 9:             Create $additionalStates$ from $state$ that applies property values not applied by $scenario$
10:             $State \leftarrow additionalStates$
11:         end if
12:         if $source$ annotates Role or Data then
13:             Create vertex $replacement$ with an added assignment that sets property values from $scenario$
14:             $currentTFG \leftarrow$ Copy $currentTFG$ with $state$ and replace $vertex$ with $replacement$
15:         else
16:             Create vertex $replacement$ with added node label of property values from $scenario$
17:             $currentTFG \leftarrow$ Copy $currentTFG$ with $state$ and replace $vertex$ with $replacement$
18:         end if
19:     end for
20:     $TFG \leftarrow currentTFG$
21: end for
22: return $TFG$
```

5.4. Providing Results

The extension of the legal assessment workflow as outlined in Section 5.1 and the extension of the data flow analysis to allow the evaluation of CDAs as outlined in Section 5.2 and Section 5.3 aim to improve the interdisciplinary communication between legal experts and software architects. Furthermore, it allows legal experts to retain a high level overview over the system under consideration and allows techniques, like subsumption and others outlined in Section 2.1.3, to work more efficiently. Lastly, the automated analysis of the software models allows legal experts to focus their legal analysis on problematic areas of the system and determine if and what clarifications or modifications are necessary.

As the users of the extended data flow analysis are outside the technical domain, the results of the analysis need to reflect what the analysis has analyzed and present the needed information for the assessment from the legal expert. Therefore, the information is presented on the basis of TFGs: For each TFG that contains a violation, a constraint violation object is constructed that includes the violating TFG, the violating vertices with their context, the state of all CDAs and whether they were resolved using design space exploration.

This allows the legal expert to evaluate for each violation what future steps might be: For violations with a state that does not include any CDAs that were resolved using design space exploration, the legal expert, can look at the specific flow of data though the system and communicate any occurring legal obligations to the software architects. Software architects can also work with this information as it is closely linked to the input model of the legal assessment workflow as shown in Figure 5.1: A violating flow with its violating nodes points to the area in the software model in which modifications may be required.

For violations with a state that includes **Uncertain CDAs**, the legal expert might need to further research the impact of the CDAs on the software: If the issue lies within the system under consideration, the legal expert might speak with software architect and reference the violating data flow though the system and ask about technical details that might be important. If the issue lies inside another system that is not under the control of the software architects, like an external system, the legal expert might investigate and communicate with the external partner.

Running Example

Using the running example introduced in Chapter 3 as a demonstration for the provided results by the data flow analysis extension, I analyze the first described constraint: The TFG ending at *create advertisements* is affected by the CDA **identifiablity** as outlined in Section 4.3. Furthermore, as outlined in the running example chapter, **identifiablity** is an **Uncertain CDA** and the *Property Value* is unknown in the context of the *marketing agency*. This results in design space exploration in which both variants, so the system with and without an identifiable user using their pseudonymized customer data, are evaluated. This leads to a violation in the first constraint, if the user is identifiable.

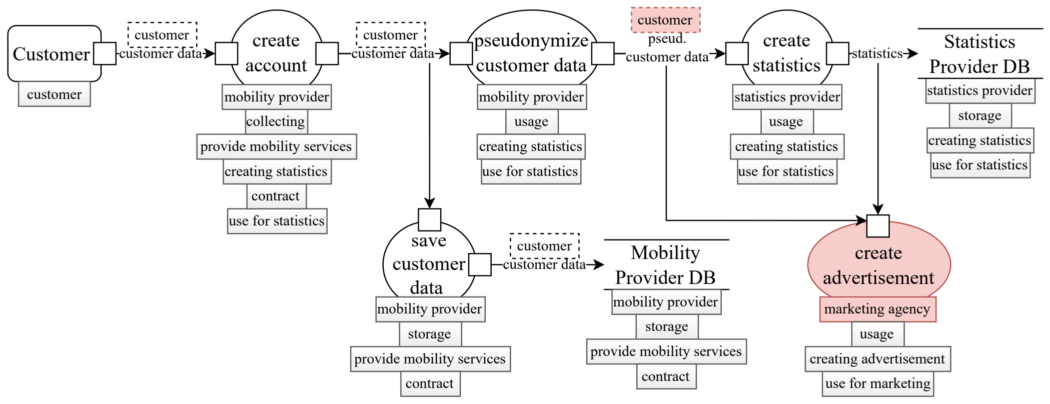


Figure 5.4.: Train Model DFD with Violations

In the results, the legal expert observes that a violation of the first constraint occurs, if the user is identifiable. A DFD marked with the problematic elements can be found in Figure 5.4. Therefore, they need to investigate further and might find out that the *marketing agency* possesses further data they use to depseudonymize customers and use the data in a non-conforming matter.

6. Evaluation

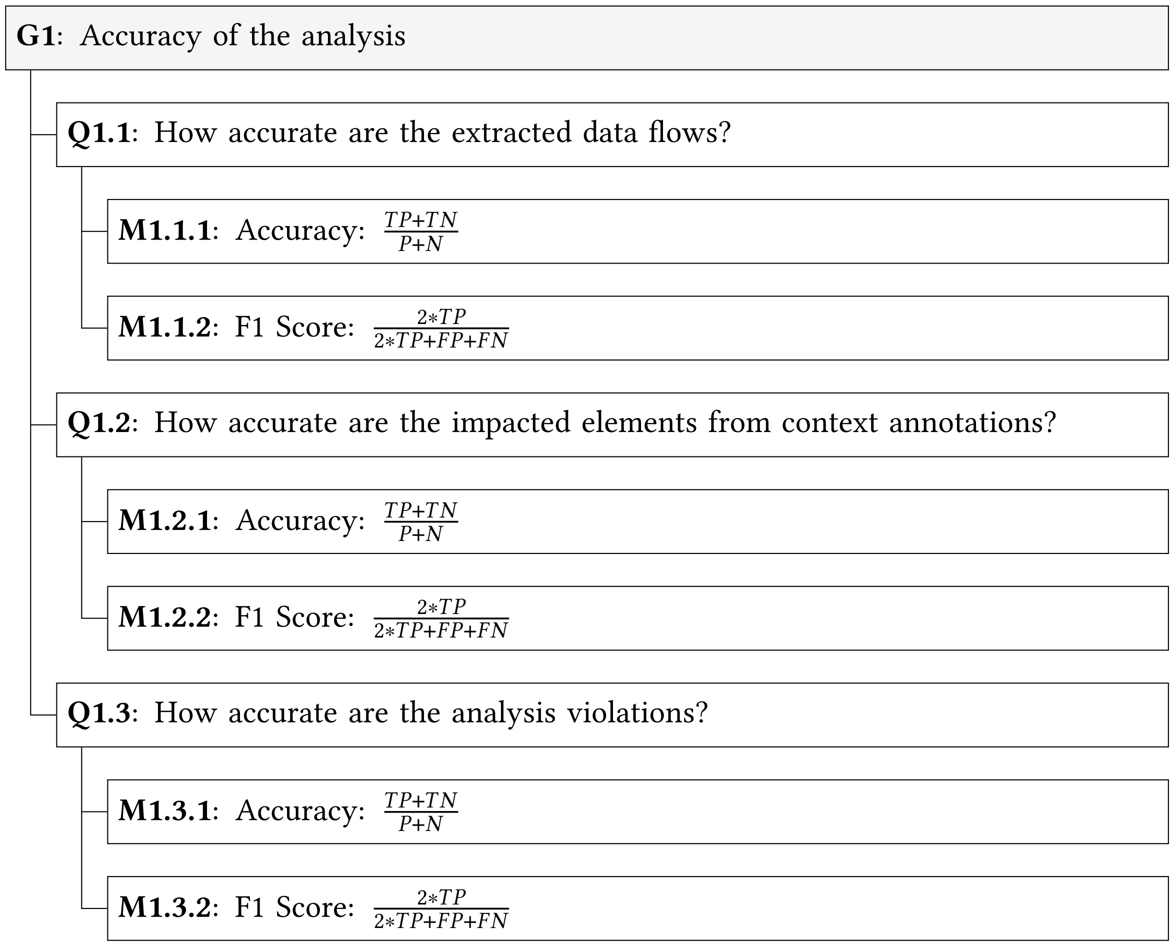
To evaluate the extended data flow analysis, the accuracy and scalability of the data flow analysis shall be evaluated. In general, this chapter aims to show the feasibility of the analysis in legal assessment frameworks, like the one outlined and expanded in Section 5.1. The following Sections describe the two goals of the proposed GQM plan according to Basili et al. [2]:

6.1. Goal 1: Accuracy

The first goal **G1** aims to evaluate the accuracy of the analysis results in different evaluation scenarios.

The question **Q1.1** aims to evaluate how accurate the extracted data flows in the form of TFGs are. Requirement to correctly find violations using constraints is to evaluate CDAs as outlined in Section 4.3 and determine the expected amount and contents of data flows. Therefore, I compare the created TFGs from the analysis with a list manually created following the description in Section 4.3. The metric **M1.1.1** and metric **M1.1.2** provide results based on the precision and recall of the extended analysis and counts the number of correct elements of TFGs. An element is counted as a true positive entry, if the actual and expected TFG has the element at the same position, with the same previous nodes and the same referenced GDPR element. An element is counted as a false positive entry, if the actual TFG contains an element at a position, with previous nodes and a referenced GDPR element, while the expected TFG does not contain the element or the position, previous elements or referenced GDPR element differ. An element is counted as a false negative entry, if the expected TFG contains an element at a position, previous nodes and referenced GDPR element, but the actual TFG does not contain the element with the same attributes.

The question **Q1.2** aims to evaluate how accurate the impact of CDA on data flows in the form of TFGs are. The set of impacted elements is defined as elements that are affected by at least one CDA or follow after another impacted element. Finding the correct impacted elements is requirement to correctly find violations. Therefore, I create the TFGs from the analysis, track impacted elements and compare them with a list manually created following the description in Section 4.3. The metric **M1.2.1** and metric **M1.2.2** provide results based on the precision and recall of the extended analysis and counts the number of correct elements of TFGs. An element is impacted by a CDA, if it had it behavior changed by it or a previous element has been impacted. An element is counted as a true positive entry, if the element is impacted in the expected and actual TFG. An element is counted as a false



G1: Accuracy of the analysis

Q1.1: How accurate are the extracted data flows?

M1.1.1: Accuracy: $\frac{TP+TN}{P+N}$

M1.1.2: F1 Score: $\frac{2*TP}{2*TP+FP+FN}$

Q1.2: How accurate are the impacted elements from context annotations?

M1.2.1: Accuracy: $\frac{TP+TN}{P+N}$

M1.2.2: F1 Score: $\frac{2*TP}{2*TP+FP+FN}$

Q1.3: How accurate are the analysis violations?

M1.3.1: Accuracy: $\frac{TP+TN}{P+N}$

M1.3.2: F1 Score: $\frac{2*TP}{2*TP+FP+FN}$

Figure 6.1.: GQM plan for Goal 1

positive entry, if the element is impacted in the actual TFG, but not in the expected TFG. An element is counted as a false negative entry, if the element has been impacted in the expected TFG, but has not been impacted in the actual TFG.

The question **Q1.3** aims to evaluate how accurate the violations of the analysis are, when CDAs are evaluated. To support legal experts in their legal assessments, the analysis results need to perform the design space exploration and result presentation as outlined in Section 5.2 and Section 5.4. Therefore, I create the TFGs from the analysis, run the label propagation, find violations with the constraint and compare the results with a list manually created following the description in Section 4.3 and the work by Boltz et al.[4]. The metric **M1.3.1** and metric **M1.3.2** provide results based on the precision and recall of the extended analysis and counts the number of correct elements of TFGs. An element is counted as a true positive entry, if the element has a violation in the expected and actual TFG. An element is counted as a false positive entry, if the element has a violation in the actual TFG, but not in the expected TFG. An element is counted as a false negative entry, if the element has a violation in the expected TFG, but does not in the actual TFG.

In the following, I describe the two different models that have been used to evaluate the accuracy of the software model:

6.1.1. Train Model

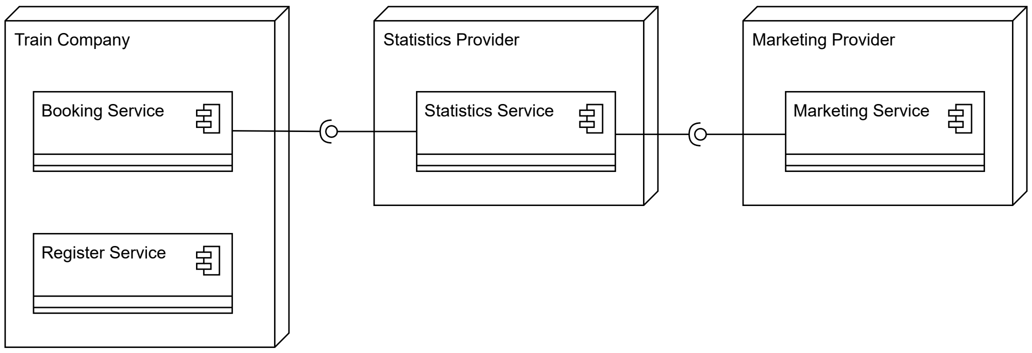


Figure 6.2.: Simplified component and deployment diagram of the train model

Source This model was created for use with the GDPR legal assessment framework.

Description The train model is used as a running example during this thesis and is described in Detail in Chapter 3.

Context Dependent Attributes The train model contains two different CDAs: The first CDA of identifiability is annotated to the *Customer*, which data is passed pseudonymized and unpseudonymized though the system. The CDA of identifiablity is true, when the unpseudonymized data is available, which corresponds only to elements in the responsibility of the *train company*. In the case of pseudonymized data, the CDA behaves like an **Uncertain CDA** as outlined in Section 4.3 and Section 5.3 and the *Property Value* of the CDA is unknown. The second CDA of transparency annotates a *Purpose* that annotates the *Processing* element of the *marketing provider* that creates advertisements from the pseudonymized customer data. The legal expert has not determined, whether the processing for that purpose is transparent to the user yet, so it is modelled as an **Uncertain CDA**.

GDPR Constraint The train model consists of two constraints: For the first constraint, the *marketing provider* may not receive data in which the user is identifiable. This may be required as the *marketing provider* may possess more data that can be used to depseudonymize the data and use it for non-compliant purposes in regard to the GDPR. The second constraint prohibits the processing of data with untransparent purposes, because Article 5(2) of the GDPR stipulates that "personal data shall be collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes"

6.1.2. Travel Planner

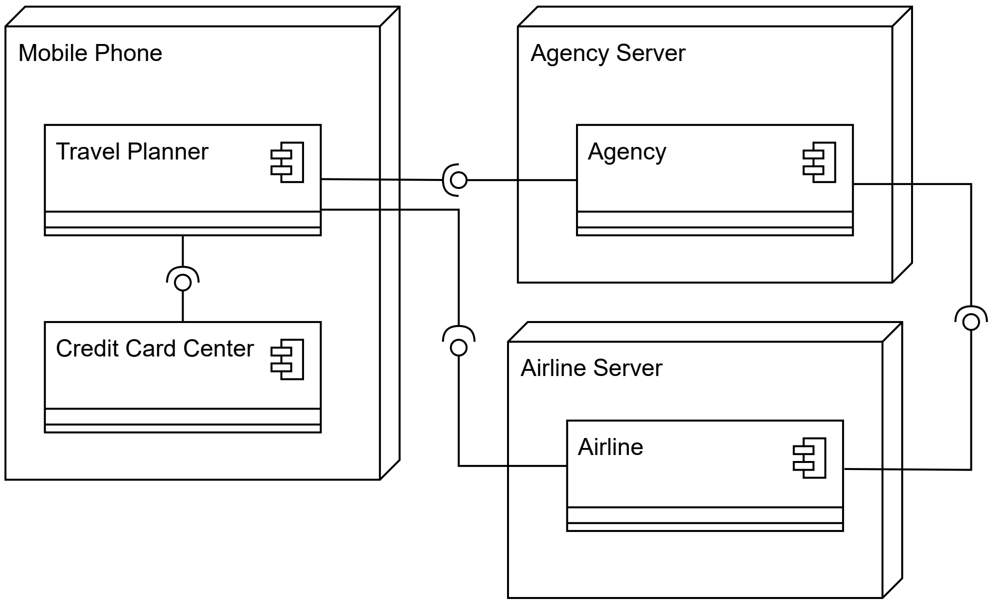


Figure 6.3.: Simplified component and deployment diagram of the Travel Planner

Source The Travel Planner evaluation model is originally from the iFlow project by Katkalov [9]. Its original purpose was the evaluation of an information flow analysis [10] and Palladio Component Model of this evaluation model has been created by Seifermann et al. [16]. The evaluation model has already been used in multiple validations of case studies [15, 8].

Description The Travel Planner evaluation model is shown in a simplified component and deployment diagram in Figure 6.3 and consists of three central entities: A *customer*, a *travel agency* and an *airline*. The *customer* uses the Travel planner on his mobile phone. The application communicates with a *travel agency* that queries flights from multiple different *airline*s and shows them to the *customer*. The queried flights can then be booked using a credit card, which is stored on the mobile phone.

Context Dependent Attribute For the Travel Planner evaluation model, I describe one CDA: The CDA **necessary** denotes whether the data is required for the purposes for which they are processed. This CDA is closely related to data minimization principle outlined in Article 5(3) of the GDPR.

GDPR Constraint Following the principles in Article 5(3) of the GDPR, the constraint is defined to forbid any flow of data that is not **necessary**.

6.1.3. Accuracy Results

The data of the accuracy evaluation of the extended analysis on the train example model, outlined in Chapter 3, can be found in Appendix A.2. The data of the accuracy evaluation of the extended analysis on the Travel Planner, outlined in Section 6.1.2, can be found in Appendix A.3.

For calculating the number of expected TFGs for the metric **M1.1.1** and metric **M1.1.2**, one needs to determine the TFGs in the converted GDPR model first: As outlined in the running example section of Section 4.1, the running example model contains three different TFGs. Additional TFGs are created for the three base TFGs according to the design space exploration outlined in Section 4.3. As only the third TFG includes the marketing provider, and in all other TFGs the defined CDAs are **Fully Resolvable CDA**, additional TFGs will only be created for the third TFG. As two **Uncertain CDA** exist for the third TFG, four TFG will be created from the third TFG. The six resulting TFGs are then split according to their responsibilities as outlined in Section 4.2 and their original TFGs preserved. The first TFG only contains one responsibility context, therefore no additional TFGs will be created. For the second TFG two TFGs will be created, as the TFG contains two different responsibility contexts. For the third TFGs, each TFG will result in three additional TFGs, as each contain the *train company*, *statistics provider* and *marketing provider* responsibilities. This results in a final total of 20 TFGs as the expected amount of TFGs. Evaluation of the analysis also shows that the analysis finds exactly 20 TFGs. Therefore, the Accuracy metric **M1.1.1** and F1-Score metric **M1.1.2** are both at 1.0 for the Train Example Model.

TFGs Amount	TP	FP	FN	Precision	Recall	F1-Score
20	20	0	0	1.0	1.0	1.0

A similar procedure for the Travel Planner model as outlined in Section 6.1.2 also results in 20 expected TFGs. The analysis also finds 20 TFGs, therefore the Accuracy metric **M1.1.1** and F1-Score metric **M1.1.2** are both at 1.0 for the Travel Planner model.

TFG Amount	TP	FP	FN	Precision	Recall	F1-Score
20	20	0	0	1.0	1.0	1.0

Determining the expected impacted elements for the Train Example Model is quite simple: As each *Processing* node references the *Customer* though *Customer Data* or *Pseudonymized Customer Data*, the CDA of *identifiability* is impacting each element. The analysis also finds that each element in the set of TFGs is impacted. Therefore, the Accuracy metric **M1.2.1** and F1-Score metric **M1.2.2** are both at 1.0 for the Train Example Model.

Total Elements	Impacted Elements	TP	FP	FN	Precision	Recall	F1-Score
35	35	35	0	0	1.0	1.0	1.0

Determining the expected impacted elements for the Travel Planner model follows the outlined process of determining the application of CDA, outlined in Section 4.3: As the *Collecting* node *Collect CCD* references the annotated element of the CDA **necessary**, each element following *Collect CCD* will be impacted by the CDA. The analysis finds that each element following *Collect CCD* in the two TFGs are impacted. Therefore, the Accuracy metric **M1.2.1** and F1-Score metric **M1.2.2** are both at 1.0 for the Travel Planner model.

Total Elements	Impacted Elements	TP	FP	FN	Precision	Recall	F1-Score
24	12	12	0	0	1.0	1.0	1.0

For evaluating question **Q1.3**, I compare the expected violating elements with the results produced by the analysis. For the Train Example model, violations only occur when looking at TFGs including the *marketing provider*. In this TFG violations only occur for the **transparency** CDA, when the *Create Advertisement Purpose* is annotated with **transparency** is `False`. For the **identifiability** CDA violations occur, when the responsible role for the *Processing* elements is the *marketing provider*. Therefore, 8 violating elements are contained in the TFGs and 8 are found by the analysis. Therefore, the Accuracy metric **M1.3.1** and F1-Score metric **M1.3.2** are both at 1.0 for the Train Example model.

Total Elements	Violating Elements	TP	FP	FN	Precision	Recall	F1-Score
35	8	8	0	0	1.0	1.0	1.0

For the Travel Planner model there only exists one violating node: In the case in which the CDA **necessary** is false, and the credit card details are transmitted though the *travel agency*, a violation occurs. The analysis also finds this violating node. Therefore, the Accuracy metric **M1.3.1** and F1-Score metric **M1.3.2** are both at 1.0 for the Travel Planner model.

Total Elements	Violating Elements	TP	FP	FN	Precision	Recall	F1-Score
24	1	1	0	0	1.0	1.0	1.0

6.2. Goal 2: Scalability

The goal **G2** aims to evaluate the scalability of the analysis results in different evaluation scenarios.

The question **Q2.1** aims to evaluate how the impact of the data flow length and number on the scalability of the analysis. This helps to ascertain, whether the analysis approach can be used on larger and detailed software models. Therefore, I measure the runtime on generated software models and scale the elements in the software model that impact the number and length of the data flow. First, a model is generated that has one data flow with an increasing

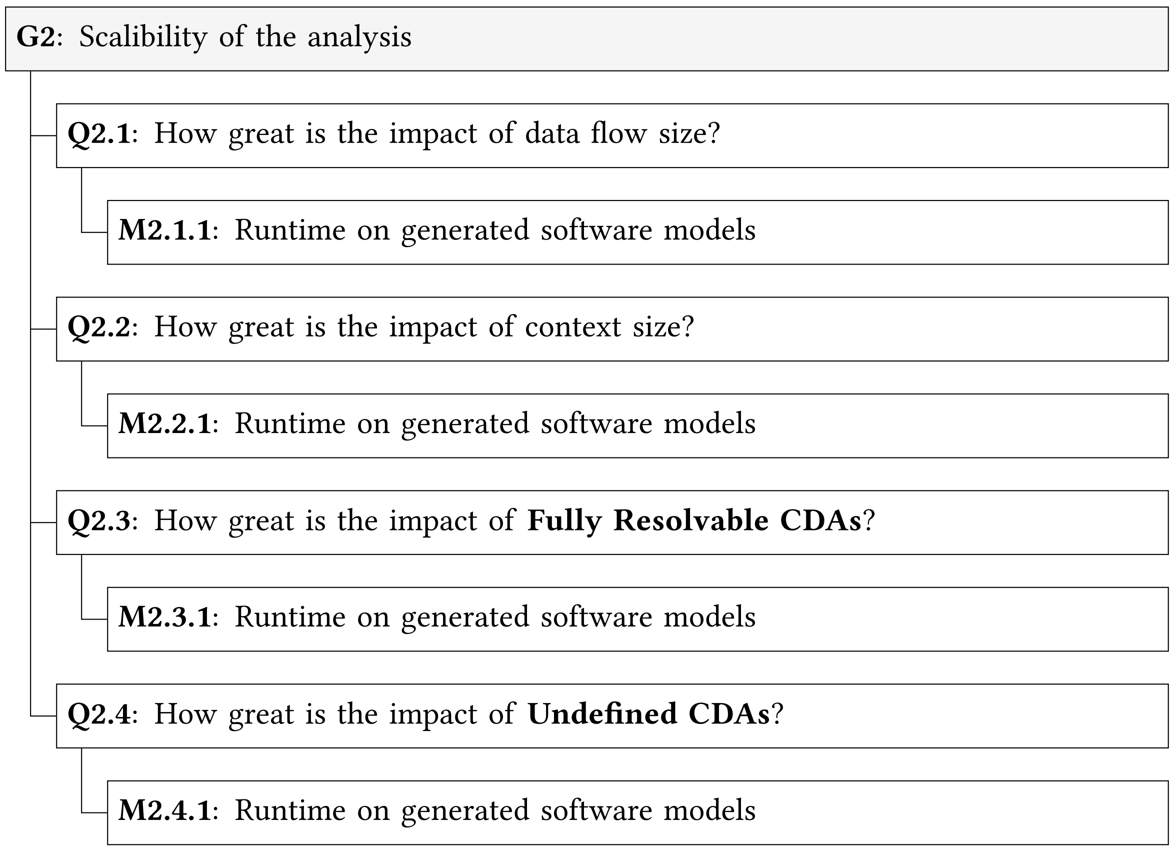

G2: Scalibility of the analysis
Q2.1: How great is the impact of data flow size?
M2.1.1: Runtime on generated software models
Q2.2: How great is the impact of context size?
M2.2.1: Runtime on generated software models
Q2.3: How great is the impact of **Fully Resolvable CDAs**?
M2.3.1: Runtime on generated software models
Q2.4: How great is the impact of **Undefined CDAs**?
M2.4.1: Runtime on generated software models

Figure 6.4.: GQM plan for Goal 2

length by scaling the number of *Processing* elements. Second, we scale the amount of *Data* elements in the GDPR model, resulting in additional effort during the propagation as each *Data* element may have different *Label* that need to be propagated. The metric **M2.1.1** provides an insight into the runtime of the analysis on growing software models and in this case the length and propagation effort needed during analysis.

The question **Q2.2** aims to evaluate how great the impact of the GDPR context is on the scalability of the analysis. This helps to ascertain, whether the analysis approach can be used on larger and detailed software models. Furthermore, this would allow legal experts to annotate elements in greater detail and investigate different responsibility subsystems as outlined in Section 4.2. Therefore, I measure the runtime on generated software models and scale the elements in the software model that impact the context size of TFG elements. I will provide results for the scaling of *Purpose* GDPR elements that only impact the context size of the TFG elements, but also *Role* GDPR elements that further impact the amount of created TFGs, as multiple responsibility boundaries are crossed. Furthermore, I scale the number of *Context Definition*s that check the context and in another scenario scale the number of elements in the context that is required by the *Context Definition.* The metric **M2.2.1** provides an insight into the runtime of the analysis on growing software models and in this case impact of context size and responsibility contexts.

The question **Q2.3** aims to evaluate how great the impact of **Fully Resolvable CDA**. This helps to ascertain, whether the analysis approach can analyze multiple CDA that might occur in the legal assessment of a system. Therefore, I measure the runtime on generated software models and scale the amount of **Fully Resolvable CDA**. The metric **M2.3.1** provides an insight into the runtime of the analysis in the case of a growing amount of **Fully Resolvable CDA**.

The question **Q2.4** aims to evaluate how great the impact of **Uncertain CDA**. This helps to ascertain, whether the analysis approach can analyze multiple CDA that might occur in the legal assessment of a system. Therefore, I measure the runtime on generated software models and scale the amount of **Uncertain CDA**. The metric **M2.4.1** provides an insight into the runtime of the analysis in the case of a growing amount of **Uncertain CDA**.

In the following section, I describe the results of the scalability evaluation in the scalability dimensions as outlined above. For each run, I increased the amount of model elements by 10 and evaluate the data flows across all elements in the TFGs. Each test is run 10 times and the median execution time is calculated to exclude any outliers or measurement anomalies. The scalability evaluation has been run on a dedicated VM with 4 AMD Opteron 8435 cores together with 97 GB RAM and runs Debian 11 with OpenJDK 17. This VM has also been used to evaluate the scalability in Boltz et al. [4] and Schwickerath et al. [15].

Results

The evaluation results for question **Q2.1** using the defined runtime metric **M2.1.1** can be seen in Figure 6.5. The graph showing the median execution time for *Processing* elements shows that the execution time grows approximately quadratic in the number of processing elements. As the process of label propagation is most likely the expensive subprocess of the analysis, the result shows that the propagation behaves similarly to the runtime of the base analysis [4]. Looking at the graph of the execution time for scaling *Data* elements, the impact is less severe as compared to increasing the amount of *Processing* elements. Therefore, it may be concluded that the impact of the propagated data elements is not high.

Figure 6.6 shows the evaluation results for question **Q2.2** using its defined runtime metric **M2.2.1**.

Increasing the size of the context, by increasing the amount of *Purpose* elements, does not seem to have a great impact on the runtime of the analysis. Most of the impact is likely mitigated by the filtering of CDAs, as described in Section 4.1.

The runtime of the analysis increases quadratic in the number of *Role*s. This may be due to the creation of **responsibility context-aware, TFGs** as outlined in Section 4.2. This creates overhead throughout the whole analysis as it increases the amount of TFGs that are propagated and checked for violations.

The number of *Context Definition*s querying the context of GDPR elements seems to have a growing impact the more *Context Definition*s are modelled. This overhead is most likely

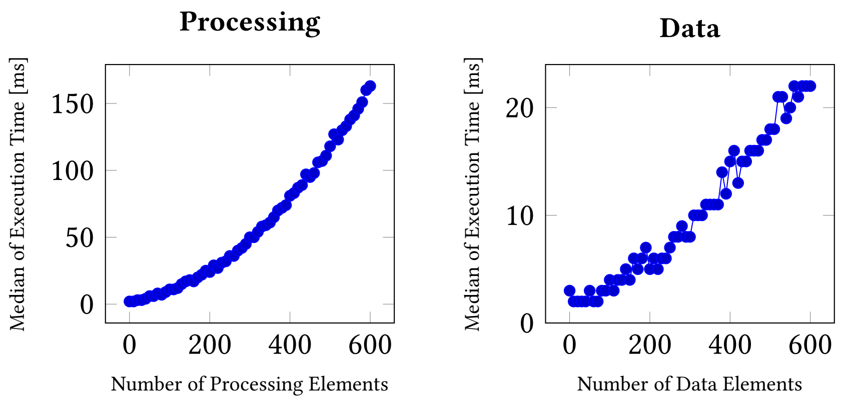


Figure 6.5.: Scalability Results regarding Question **Q2.1**

explained by the increasing amount of `Context Dependent Attribute Scenarios` that will be created for each *Context Definition*, as is outlined in Section 5.2.

The number of elements in a *Context Definition* requiring to check whether all GDPR elements in the context are contained, so its size, does not seem to impact the execution time.

For the next two figures, the scaling of both axes is different, as the number of nodes have been increased by powers of 2 instead of increased by 10 each step. Figure 6.7 shows the generally low runtime impact of **Fully Resolvable CDAs**. Regardless whether they modify the node or data *Label* or a TFG element, the runtime impact is negligible.

Figure 6.8 shows the increased runtime impact of **Uncertain CDAs**, especially in comparison to **Fully Resolvable CDAs**. Again, the runtime impact between **Uncertain CDAs** that modify node *Label* and **Uncertain CDAs** that modify data *Label* is identical. The increased impact of **Uncertain CDAs** in comparison to **Fully Resolvable CDAs** can be explained with design space exploration required to resolve **Uncertain CDAs**, as Section 4.3 and Section 5.3 outline. However, comparing the runtime impact of CDAs in general to the impact of additional *Processing* elements, it becomes clear that the runtime of the analysis is dominated by the model elements that impact the length of TFGs and not the amount of CDAs.

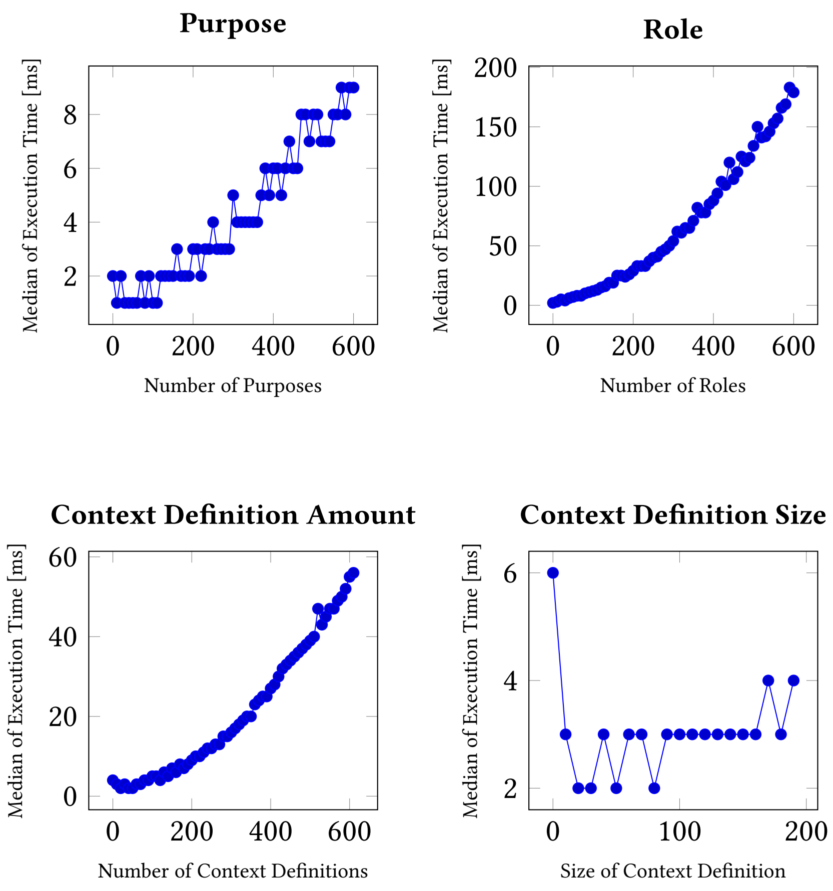


Figure 6.6.: Scalability Results regarding Question **Q2.2**

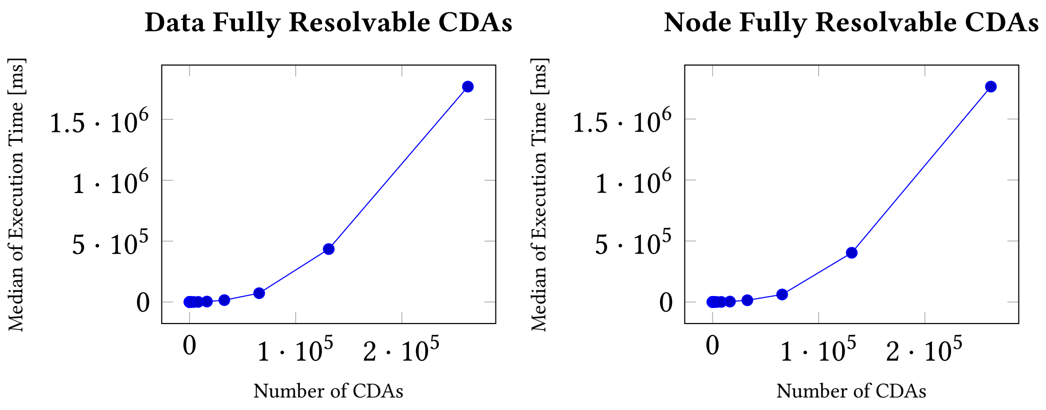


Figure 6.7.: Scalability Results regarding Question **Q2.3**

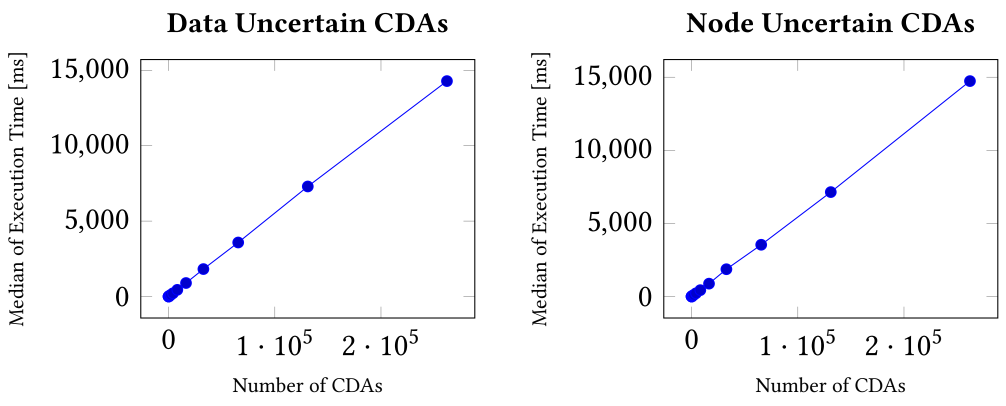


Figure 6.8.: Scalability Results regarding Question **Q2.4**

6.3. Threats to Validity

Threats to the validity of this thesis are evaluated based on the scheme proposed by Runeson et al. [14].

For ensuring construct validity, I used the GQM-plan outlined in this chapter and basing the input of the analysis on a software model that have been co-created by legal experts [3]. A threat to construct validity may be limited applicability to the legal domain, as the data flow extension has not been used in a user study with legal experts.

Regarding internal validity, a threat might arise from the created evaluation scenarios used to validate Goal **G1**: First, some of the evaluation scenarios and models were created by myself and might be created differently by experts. Furthermore, the constraints defined for the evaluation scenarios might be formulated and checked differently by other experts.

The primary threat to external validity is caused by the limited generalizability caused by the case study-based nature of the evaluation: Only a limited set of software models have been used during the accuracy evaluation of the thesis. Therefore, the implementation of the procedures outlined in Chapter 4 might not be correct or contain bugs.

7. Conclusion

This thesis presented an approach how GDPR models can be evaluated using the data flow analysis and how the existing conversion by Boltz et al. [5] from GDPR models to DFD models can be used in the process to evaluate GDPR models regarding legal assessments. It outlined important topics in the resolution of CDAs, like the (technical) context of an GDPR element or the distinction between **Fully Resolvable CDA** and **Uncertain CDA**. Moreover, it defines **responsibility context-aware TFGs** as a way for legal experts to look at the different responsibility contexts present in the system. Lastly, it outlines how CDA modify the behavior of DFD elements in a TFG and in what order they must be applied to correctly resolve them without greater overhead.

Additionally, this thesis described how the implementation might fit into the larger context of the legal assessment framework as presented by Boltz et al. [5]. It describes how the data flow analysis from Boltz, Hahner and Schwickerath et al. [15, 4] can be extended to allow the analysis of GDPR models. It outlines constructs, like `Context Dependent Attribute Sources` and `Context Dependent Attribute Scenarios`, that can be used to effectively resolve and apply CDA. They also can handle **Fully Resolvable CDA** and **Uncertain CDA** without additional processing during the resolving. Furthermore, I show that the provided results by the analysis can be used in legal assessment processes by legal experts. The results also support the interdisciplinary communication between software architects and legal experts and allows the communication of issues between the two parties.

The evaluation of this thesis shows that the analysis can be feasibly used for the legal assessment of systems, as it provides accurate results. Furthermore, the runtime of the extended analysis will be reasonably low for larger software models.

7.1. Future Work

The future work of this approach can be categorized in two categories:

The first category of future work includes the research into the further extension of the analysis. For example, the Legal Norm Rule Metamodel [5] can be converted into a constraint for the usage in this extended data flow analysis. As the formulation of constraints is currently only possible in Java and a Java DSL, it might be helpful for legal experts to express their constraints to the software model in a format closer to natural language. Other extensions might be the usage of machine learning to improve the analysis or its results. It may be able to create *Context Definition*s for CDA or created fixed software models after an violation by the analysis is found.

The second category of future work might include research into the connection with uncertainties. As **Uncertain CDA** already use design space exploration, a technique that is used to resolve Uncertainties, further research into their connection might yield promising results. A comparison between this work and work by Hahner et al. [8] might provide valuable insights into the connection of the two topics and might allow methods from one topic to be explored in the Other.

Bibliography

[1] Sushant Agarwal et al. “Legislative Compliance Assessment: Framework, Model and GDPR Instantiation”. In: *Privacy Technologies and Policy*. Ed. by Manel Medina et al. Cham: Springer International Publishing, 2018, pp. 131–149. ISBN: 978-3-030-02547-2.

[2] Gianluigi Basili, Victor R Caldiera, and H Dieter Rombach. “The goal question metric approach”. In: *Encyclopedia of software engineering* (1994), pp. 528–532.

[3] Nicolas Boltz et al. *A Model-Based Framework for Simplified Collaboration of Legal and Software Experts in Data Protection Assessments*. ISBN: 978-3-88579-720-3 Pages: 521–532 Published: INFORMATIK 2022. 2022. DOI: `10.18420/inf2022_44`.

[4] Nicolas Boltz et al. “An Extensible Framework for Architecture-Based Data Flow Analysis for Information Security”. In: *Software Architecture. ECSA 2023 Tracks, Workshops, and Doctoral Symposium*. Ed. by Bedir Tekinerdoğan et al. Cham: Springer Nature Switzerland, 2024, pp. 342–358. ISBN: 978-3-031-66326-0.

[5] Nicolas Boltz et al. “Bridging Legal and Technical Realms: An Architecture Model-Based Framework for Continuous Data Protection Legal Assessments”. In: *Journal of Systems and Software (JSS)* (2024). Preprint, to be submitted.

[6] Tom DeMarco. “Structured Analysis and System Specification”. In: *Software Pioneers: Contributions to Software Engineering*. 1979, pp. 529–560. ISBN: 978-3-642-59412-0. DOI: `10.1007/978-3-642-59412-0_33`. URL: `https://doi.org/10.1007/978-3-642-59412-0_33`.

[7] Naeem Esfahani, Sam Malek, and Kaveh Razavi. “GuideArch: Guiding the exploration of architectural solution space under uncertainty”. In: *2013 35th International Conference on Software Engineering (ICSE)*. 2013, pp. 43–52. DOI: `10.1109/ICSE.2013.6606550`.

[8] Sebastian Hahner et al. “Model-based Confidentiality Analysis under Uncertainty”. In: *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*. 2023, pp. 256–263. DOI: `10.1109/ICSA-C57050.2023.00062`.

[9] Kuzman Katkalov. “Ein modellgetriebener Ansatz zur Entwicklung informationsflusssicherer Systeme”. doctoralthesis. Universität Augsburg, 2017.

[10] Kuzman Katkalov et al. “Model-Driven Development of Information Flow-Secure Systems with IFlow”. In: *2013 International Conference on Social Computing*. 2013, pp. 51–56. DOI: `10.1109/SocialCom.2013.14`.

[11] Oliver Liu. “Design Space Evaluation for Confidentiality under Architectural Uncertainty”. 46.23.03; LK 01. Abschlussarbeit - Bachelor. Karlsruher Institut für Technologie (KIT), 2021. 41 pp. DOI: `10.5445/IR/1000139590`.

[12] Raimundas Matulevičius et al. “A Method for Managing GDPR Compliance in Business Processes”. In: *Advanced Information Systems Engineering*. Ed. by Nicolas Herbaut and Marcello La Rosa. Cham: Springer International Publishing, 2020, pp. 100–112. ISBN: 978-3-030-58135-0.

[13] Ralf H. Reussner et al., eds. *Modeling and Simulating Software Architectures – The Palladio Approach*. MIT Press, 2016. 408 pp. ISBN: 978-0-262-03476-0.

[14] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical Software Engineering* 14.2 (Apr. 2009), pp. 131–164. ISSN: 1573-7616. DOI: 10.1007/s10664-008-9102-8. URL: https://doi.org/10.1007/s10664-008-9102-8.

[15] Felix Schwickerath et al. *Tool-Supported Architecture-Based Data Flow Analysis for Confidentiality*. 2023. arXiv: 2308.01645 [cs.SE].

[16] Stephan Seifermann et al. “Detecting violations of access control and information flow policies in data flow diagrams”. In: *Journal of Systems and Software* 184 (2022), p. 111138. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2021.111138. URL: https://www.sciencedirect.com/science/article/pii/S0164121221002351.

[17] Laurens Sion, Dimitri Van Landuyt, and Wouter Joosen. “The Never-Ending Story: On the Need for Continuous Privacy Impact Assessment”. In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2020, pp. 314–317. DOI: 10.1109/EuroSPW51379.2020.00049.

[18] Jake Tom, Eduard Sing, and Raimundas Matulevičius. “Conceptual Representation of the GDPR: Model and Application Directions”. In: *Perspectives in Business Informatics Research*. Ed. by Jelena Zdravkovic et al. Cham: Springer International Publishing, 2018, pp. 18–28. ISBN: 978-3-319-99951-7.

[19] David Wright, Rachel Finn, and Rowena Rodrigues. “A comparative analysis of privacy impact assessment in six countries”. In: *Journal of Contemporary European Research* 9.1 (2013). DOI: 10.30950/jcer.v9i1.513.

[20] Reinhold Zippelius and Thomas Würtenberger. *Juristische Methodenlehre*. Vol. 4. Beck, 1999. ISBN: 978-3-406-63668-4.

A. Appendix

A.1. Legal Expressions

cummulative conflicting norms translated from the german "kummulative Normenkonkurrenz"

definitional elements translated from the german "Tatbestandteile"

higher order legislation translated from the german "Rangverhältnis"

legal consequence translated from the german "Rechtsfolge"

legal requirement translated from the german "Tatbestand"

matter of fact translated from the german "Tatsache"

specialization translated from the german "Spezialität"

subsumption translated from the german "Subsumtion"

A.2. Evaluation Results - Train Model

The Train Example Model from Chapter 3 contains 3 TFG per default. The design space exploration outlined in Section 4.3 grows the number of TFG from 3 to 6. Lastly, the number of TFG grows from 6 to 20 through the different responsibility contexts that are created as outlined in Section 4.2.

In the evaluation results **responsibility aware-TFG** are omitted, as they are contained in the larger TFG they originate from. The following TFG are found by the analysis. Elements marked in yellow are impacted by at least one CDA. Elements marked in orange are violating the constraints outlined in Chapter 3 and are impacted as well, unless otherwise denoted.

TFG1: Customer registers Account State: Train Company(TC) Context	TFG2: Statistics Provider uses data State: Train Company(TC) Context, Statistics Provider(SP) Context
RegisterCustomer, _0854cQhKEe67tpwR5G7Ipg	RegisterCustomer, _0854cQhKEe67tpwR5G7Ipg
TrainStoreCustomerData, _XxG3NAhLEe67tpwR5G7Ipg	Pseudonymization, _6JfSEAhKEe67tpwR5G7Ipg
	TransferToStatisticsProvider, _AJYX8ghLEe67tpwR5G7Ipg
	StatisticsProviderCalculateStatistics, _cWKWwwhLEe67tpwR5G7Ipg
	StatisticsProviderStoreData, _fQEjJAhLEe67tpwR5G7Ipg

TFG3: Marketing Provider receives data State: TC,SP,True@Identifiability,True@Transparency	TFG3: Marketing Provider receives data State: TC,SP,True@Identifiability,False@Transparency
RegisterCustomer, _0854cQhKEe67tpwR5G7Ipg	RegisterCustomer, _0854cQhKEe67tpwR5G7Ipg
Pseudonymization, _6JfSEAhKEe67tpwR5G7Ipg	Pseudonymization, _6JfSEAhKEe67tpwR5G7Ipg
TransferToStatisticsProvider, _AJYX8ghLEe67tpwR5G7Ipg	TransferToStatisticsProvider, _AJYX8ghLEe67tpwR5G7Ipg
StatisticsProviderCalculateStatistics, _cWKWwwhLEe67tpwR5G7Ipg	StatisticsProviderCalculateStatistics, _cWKWwwhLEe67tpwR5G7Ipg
TransferToMarketing, _Rk-8cjxHEe-7xYeJC8-49w	TransferToMarketing, _Rk-8cjxHEe-7xYeJC8-49w
MarketingCreateAdvertisement, _mMyfMzxNEe-aJKhk6DSkcA	MarketingCreateAdvertisement, _mMyfMzxNEe-aJKhk6DSkcA
MarketingStoreData, _wiKmVDxHEe-7xYeJC8-49w2	MarketingStoreData, _wiKmVDxHEe-7xYeJC8-49w2

TFG3: Marketing Provider receives data State: TC,SP,False@Identifiability,True@Transparency	TFG3: Marketing Provider receives data State: TC,SP,False@Identifiability,False@Transparency
RegisterCustomer, _0854cQhKEe67tpwR5G7Ipg	RegisterCustomer, _0854cQhKEe67tpwR5G7Ipg
Pseudonymization, _6JfSEAhKEe67tpwR5G7Ipg	Pseudonymization, _6JfSEAhKEe67tpwR5G7Ipg
TransferToStatisticsProvider, _AJYX8ghLEe67tpwR5G7Ipg	TransferToStatisticsProvider, _AJYX8ghLEe67tpwR5G7Ipg
StatisticsProviderCalculateStatistics, _cWKWwwhLEe67tpwR5G7Ipg	StatisticsProviderCalculateStatistics, _cWKWwwhLEe67tpwR5G7Ipg
TransferToMarketing, _Rk-8cjxHEe-7xYeJC8-49w	TransferToMarketing, _Rk-8cjxHEe-7xYeJC8-49w
MarketingCreateAdvertisement, _mMyfMzxNEe-aJKhk6DSkcA	MarketingCreateAdvertisement, _mMyfMzxNEe-aJKhk6DSkcA
MarketingStoreData, _wiKmVDxHEe-7xYeJC8-49w2	MarketingStoreData, _wiKmVDxHEe-7xYeJC8-49w2

A.3. Evaluation Results - Travel Planner

The Travel Planner model introduced in Section 6.1.2 contains 1 TFG per default. The design space exploration outlined in Section 4.3 grows the number of TFG from 1 to 2. Lastly, the number of TFG grows from 2 to 20 through the different responsibility contexts that are created as outlined in Section 4.2.

In the evaluation results **responsibility aware-TFG** are omitted, as they are contained in the larger TFG they originate from. The following TFG are found by the analysis. Elements marked in yellow are impacted by at least one CDA. Elements marked in orange are violating the constraints outlined in Section 6.1.2 and are impacted as well, unless otherwise denoted.

TFG1: Customer uses Travel Planner	TFG1: Customer uses Travel Planner
State: UserNecessity, True@Necessity	State: UserNecessity, False@Necessity
Look for Flights, _qKvdcoGkEe-as-PKi6C5xQ	Look for Flights, _qKvdcoGkEe-as-PKi6C5xQ
Request Flights From Airline, _28FB8oGkEe-as-PKi6C5xQ	Request Flights From Airline, _28FB8oGkEe-as-PKi6C5xQ
Read Flight DB, _5PKe4IGkEe-as-PKi6C5xQ	Read Flight DB, _5PKe4IGkEe-as-PKi6C5xQ
Return Flights From Airline, __Hm48oGkEe-as-PKi6C5xQ	Return Flights From Airline, __Hm48oGkEe-as-PKi6C5xQ
Return Flights, _BBtOIoGlEe-as-PKi6C5xQ	Return Flights, _BBtOIoGlEe-as-PKi6C5xQ
Collect CCD, _XOeAsYGkEe-as-PKi6C5xQ	Collect CCD, _XOeAsYGkEe-as-PKi6C5xQ
Book Flight, _unhx0oGkEe-as-PKi6C5xQ	Book Flight, _unhx0oGkEe-as-PKi6C5xQ
Ask Airline to Book Flight, _ERYPIoGlEe-as-PKi6C5xQ	Ask Airline to Book Flight, _ERYPIoGlEe-as-PKi6C5xQ
Book Flight, _QuvuwIGlEe-as-PKi6C5xQ	Book Flight, _QuvuwIGlEe-as-PKi6C5xQ
Return Booking Confirmation, _UCLOYYGlEe-as-PKi6C5xQ	Return Booking Confirmation, _UCLOYYGlEe-as-PKi6C5xQ
Return Booked Flight, _Vuu20oGlEe-as-PKi6C5xQ	Return Booked Flight, _Vuu20oGlEe-as-PKi6C5xQ
Store Booked Flight, _mfhTdIGlEe-as-PKi6C5xQ	Store Booked Flight, _mfhTdIGlEe-as-PKi6C5xQ