Methods and tools for GDPR Compliance through

Privacy and Data Protection 4Engineering

Requirements engineering methods for privacy and data protection v1

Project:	PDP4E
Project Number:	787034
Deliverable:	D4.4
Title:	Requirements engineering methods for privacy and data protection v1
Version:	1.0
Date:	16/07/2019
Confidentiality:	Public
Author:	Patrick Tessier Gabriel Pedroza Nicolás E. Diaz Ferreyra

Funded by

Table of Contents

Document History

Version	Status	Date
V0.1	First proposition	18/04/2019
V0.2	Contribution from Nicolas Díaz Ferreyra	
V0.3	Completing sections related to PDP4E method for requirements elicitation. Overall reviewing of the document.	25/06/2019
V0.4	Review the document	25/06/2019
V0.5	Review of the document	01/07/2019
V0.7	Change the title of the document and Modification about requirement pattern	01/07/2019
V0.8	Correction of typos and bibliography	12/07/2019
V1.0	Finalize document	16/07/2019

Approval		
	Name	**Date**
Prepared	Gabriel Pedroza, Patrick Tessier	18/04/2019
Reviewed	Victor Muntés-Mulero (Beawre)	09/07/2019
Reviewed	David Sanchez (Trialog)	11/07/2019
Authorised	Antonio Kung	17/07/2019

Circulation	
Recipient	**Date of submission**
Project partners	09/07/2019
European Commission	17/07/2019

List of Figures

Abbreviations and Definitions

Abbreviation	Definition
DFD	Data-Flow Diagram
EHS	Electronic Health System
GDPR	General Data Protection Regulation
ICT	Information and Communication Technologies
PDP4E	Privacy and Data Protection 4 Engineering
ProPan	Problem-based Privacy Analysis
RE	Requirements Engineering

Executive Summary

This document describes a method for the elicitation of privacy requirements in systems and software projects. Such method takes into account the legal obligations introduced by the EU General Data Protection Regulation (GDPR) and seeks to incorporate them into the project in the early stages of its development. This approach is inspired in the Problem-based Privacy Analysis method (ProPAn) [1] which was originally developed by researchers at the University of Duisburg Essen (UDE). This method is extended and adapted to the specific needs of PDP4E with additional requirement taxonomies and software artefacts in order to align it to the expectations of the project's stakeholders and in particular to engineers.

Overall, the core contributions of this deliverable are:

- Suitability analysis of the ProPAn method
- Identification of extension points of the ProPAn method
- Elaboration of a requirements engineering (RE) method specific to the needs of the PDP4E stakeholders and in particular to engineers.

1 Introduction

1.1 Objective of the document: global method

This document corresponds to the D4.4 on “Requirement engineering methods for privacy and data protection” within the scope of WP4 in the PDP4E project. It details the different stages of a method for generating privacy requirements out of functional requirements of a system-to-be. Likewise, it elaborates on a collection of system and software artefacts that support the different activities within the method. Particularly, (i) enriched data-flow diagrams (DFDs) for analysing the type of information being stored and processed in the system, (ii) privacy requirement meta-models to capture privacy requirements, and (iii) requirement templates to instantiate these requirements to specific systems and software projects.

1.2 Structure of the document

The document provides a first description of the PDP4E method for requirements elicitation. The main theoretical foundation is the ProPan method which is described in Section 2. The section includes detailed descriptions about the diagrams upon which ProPan relies to derive privacy-related requirements. Some strengths and limitations of ProPan are also highlighted as well as a summary of related work and approaches. After this theoretical review, Section 3 includes the method proposed in PDP4E to elicit privacy and data protection requirements. The method is mainly based upon a meta-model, a set of meta-requirements and taxonomies, all three derived from GDPR. A summary of the document finally comes in Section 4.

1.3 Relation with other deliverables

The method for elicitation of requirements specified in this document considers, in particular, the potential interactions with other methods and tools developed in the scope of PDP4E. More specifically, once elicited, the requirements can be taken as a basis to guide system design tasks (WP5). Consequently, an alignment with the method for Privacy and Data Protection by Design (D5.4) is foreseen. Also, we are aware that the Data Protection Risks Analysis (D3.4, D3.1), conducted in WP3, can also produce requirements which may guide system design tasks. In consequence, a harmonization between both methods and sources of requirements is foreseen. Finally, the tools developed to implement and support the method for requirements elicitation are specified in D4.1.

2 Background

Nowadays, developing privacy-aware software systems has become a challenge of public interest. Legal frameworks such as de EU GDPR [2] have triggered major concerns about how information systems should implement data-protection functionalities and safeguard de privacy rights of their users. Privacy engineering is a discipline that has taken care of these challenges through methods, techniques and tools that allow software developers to build and incorporate privacy-related functionalities in their projects. For this purpose, privacy must be considered as a primary aspect in every system and software development process. That is, it must be considered from the early stages of a systems' life cycle. In line with this premise, privacy requirements engineering seeks to define, and document requirements related to privacy and data protection for their later implementation. Following, we analyze ProPAn [1], a problem-based approach for the identification and documentation of privacy requirements. Furthermore, we provide an overview of the method with a three-fold goal:

- Detect the most salient characteristics of ProPan
- Circumvent a set of ProPan features to be reused within PDP4E
- Define a new method for requirements elicitation that (1) integrates GDPR specificities and (2) inherits selected ProPan features

To achieve this three-fold goal, we initially discuss the state of the art of methodologies for privacy requirement engineering. This minimal yet representative survey, helps to position ProPan within the landscape of related approaches and highlight its salient characteristics.

2.1 The ProPAn Method

ProPAn [1] is a systematic method that helps identifying the privacy needs of a software system based on a set of functional requirements. Overall, the method can be divided in two phases: **Identification of Privacy-Relevant Information Flows** and **Generation of Privacy Requirements**. In the first stage, functional requirements of the system-to-be are identified and expressed using the Problem Frame notation [3]. In the second, privacy requirement candidates are generated using a set of taxonomies that reflect privacy principles included in the GDPR and the ISO 29100 [4]. In this section, we describe each phase of the method in terms of steps, external/internal inputs and outputs together with the theoretical foundations of ProPAn, namely the problem frame notation. Likewise, we describe the most salient software artifacts that are generated throughout the method in order to estimate the overall documentation effort and identify areas of improvement. For illustration purposes, we will use a case study introduced by the industrial partners of the EU project *Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS).* This scenario is based on the German healthcare system which uses health insurance schemes for the accounting of treatments.

2.1.1 Problem diagrams

One of the core characteristics of ProPAn is the elicitation of functional requirements using problem diagrams. Such diagrams where introduced by Michael Jackson [3] as an approach to describe the environment in which a system-to-be must operate and the problem it must solve. In Jacksons' approach, software development consists of building a **Machine** (i.e. the system-to-be) that must be integrated in a certain Environment represented by a collection of **Domains** (e.g. humans, technical devices, data representations, etc.). In this sense, a **System** consists of the **Machine** (which is also a domain) together with its **Environment**. Depending on its nature, Jackson classifies domains into:

- Lexical: Describe some data representation (e.g. databases)
- Biddable: Represent humans and their roles (e.g. doctors)
- Causal: Describe physical laws (e.g. other systems or deterministic agents)

This classification can be extended and specialized to improve its expressiveness. For instance, Hatebur and Heisel [6] extended this classification with *Display* and *Connection* domains to represent knowledge that is relevant to perform a safety and security analysis of software projects.

The domains of a system (including the machine) are connected through interfaces that are used for sharing **phenomena** between them. Phenomena are observable by any of the connected domains (at least two) but only controlled by one of them. Jackson classifies phenomena into:

- Causal: Represents events, actions, messages, and operations.
- Symbolic: Represents data and states.

Problem diagrams visualize the relation between the machine's environment and a **functional requirement** [3]. For instance, the problem diagram of Figure 1 illustrates the relation between a requirement R6 and the domains representing an Electronic Health System (EHS). R6 describes that the EHS must monitor the vital signs of the patients through a mobile device and generate the corresponding Electronic Health Record (EHR). There are four domains in this problem diagram:

1. The EHR database (lexical domain).
2. The Mobile Device (causal).
3. The patient (biddable).
4. The Record system (machine).

Likewise, we can identify the following interfaces and phenomena

1. The patient's *vital signs* and *demographics* are shared to the mobile device (connection phenomena).
2. The mobile device sends the patient's *vital signs* to the record system (connection phenomena).
3. The record system sends *change* requests to the EHR database (connection domain).

where the domain that controls the phenomena is denoted by a "!".

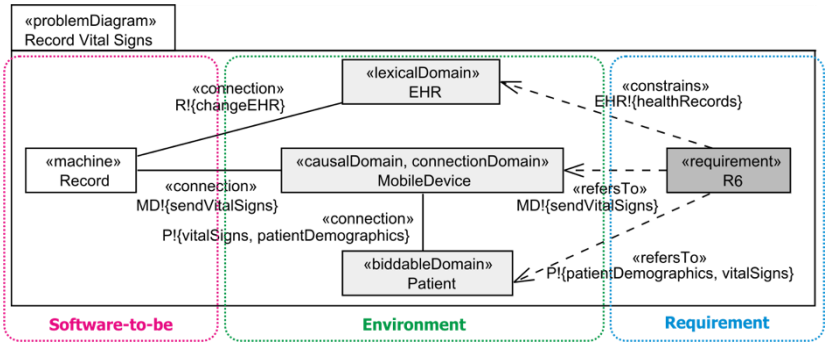

Figure 1: A problem diagram [6]

As it can be observed, a requirement can **refer to** or **constrain** a domain's phenomenon. Particularly, R6 *refers to* the patient from whom the vital signs are recorded and the mobile

device that forwards the vital signs. Further, R6 *constraints* the EHR database to store the recorded vital signs in the corresponding health record of the patient.

2.1.2 Identification of Privacy-Relevant Information Flows

Overall, this phase consists of four steps: *Context Elicitation*, *Graph Generation*, *Identification of Personal Data* and *Personal Data Flow Analysis* [1]. As illustrated in Figure 9, each step of the method draws on different external inputs and generates the outputs for the next step. The final output of this phase is a collection of diagrams that, in sum, describe which private information is stored and processed by the system-to-be. In the first step of the method stakeholders and information flows between them are elicited using questionnaires and documented using a *Context Diagram.* Likewise, functional requirements are documented as *Problem Diagrams* and *Domain Knowledge*. Next, these artifacts are passed forward to the *Graph Generation* step in order to build a *Detail Stakeholder Information Flow Graph* that describes how phenomena flows across the interfaces of the system-to-be and how stakeholders (represented as biddable domains) exchange information. This flow graph is used in the next step to identify which data of the stakeholders has been put into the system and the relations between these data. The result of this step is a Personal Information Diagram for each stakeholder. In the final step, information flows of personal data are closely analyzed using the diagrams previously generated to identify the availability and linkability of personal data at the system's domains. This is documented using *Available information Diagrams*.

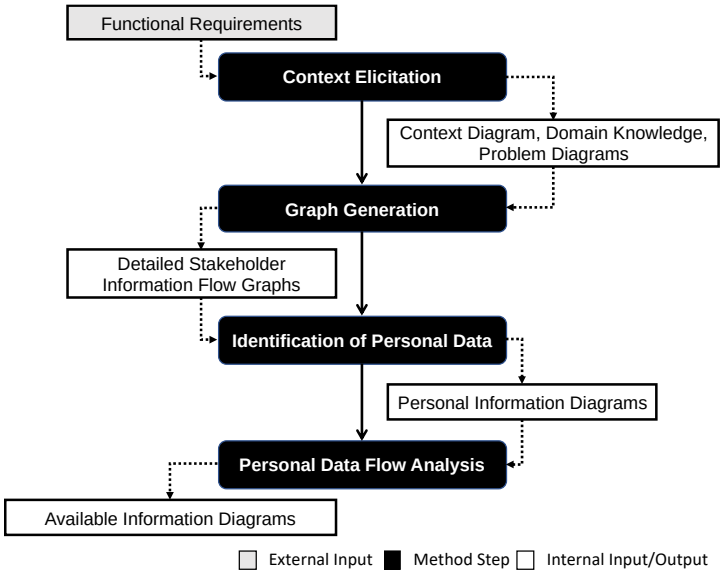

Figure 2: Identification of Privacy-Relevant Information Flows

2.1.2.1 Context Diagram

A Context Diagram (CD) describes the environment in which the machine will operate. It consists of domains and interfaces. However, it does not contain requirements (Figure 3).

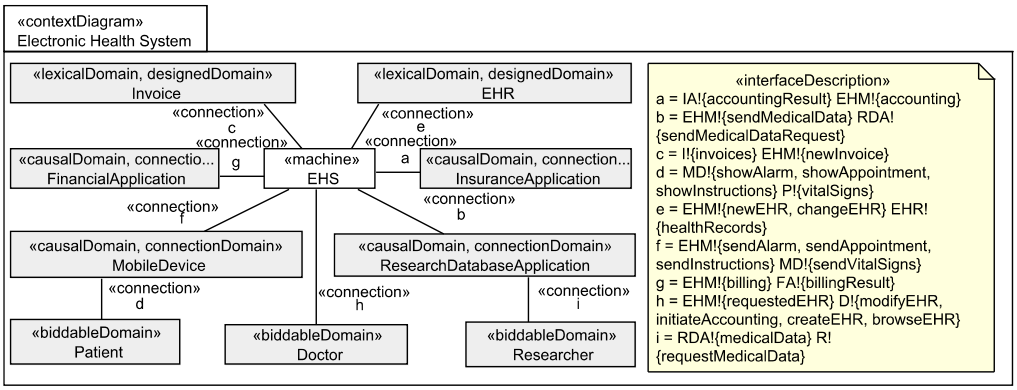

Figure 3: Context Diagram [7]

2.1.2.2 Domain Knowledge

The Domain Knowledge (DK) consists of facts and assumptions about a particular domain in the CD (Figure 4). This is done to describe the expected behaviour of another machine (i.e. system) over which we have no control. Similar to requirements, facts and assumptions *constraint* and *refer to* domains.

- Fact: A statement that is always true.
- Assumption: A statement that, under certain circumstances, may not be true.

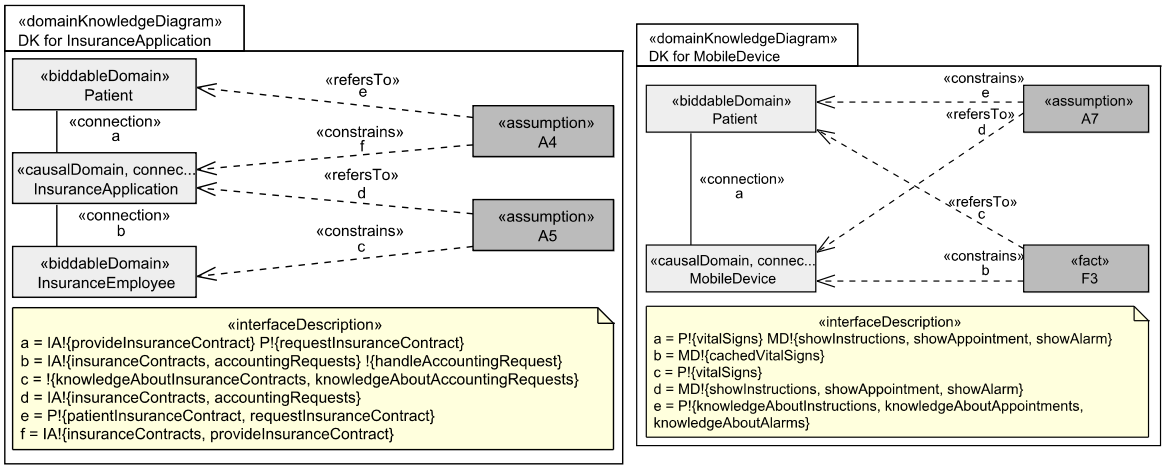

Figure 4: Domain Knowledge [7]

2.1.2.3 Detailed Stakeholder Information Flow Graph

A Detailed Stakeholder Information Flow Diagram (DSIFD) is generated **automatically** out of the PDs and the DK following this reasoning strategy: *In a problem diagram, **statements** (i.e. requirements, assumptions and facts) "refer to" and "constraint" domains of the machines' environment. If a domain is "referred to" by a statement, then this implies that such domain is a potential information source. Likewise, if a domain is "constrained" by a statement, then this implies that there is a change in such domain based on the information from the referred domain. Hence, there is a potential information flow from the "referred to" domains to the "constrained" one* [8]. Figure 5 illustrates a DSIFD.

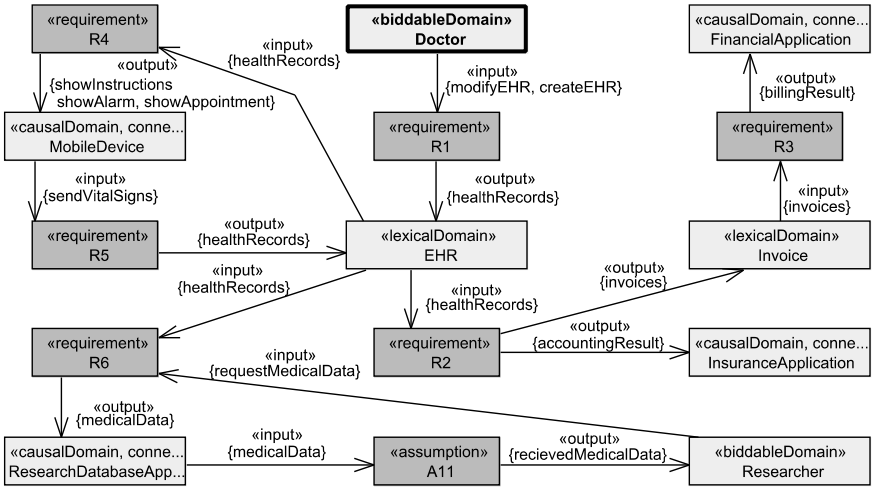

Figure 5: Detailed Stakeholder Information Flow Diagram [9]

2.1.2.4 Personal Information Diagram

These diagrams capture which personal data of a *biddable domain* (i.e. data subject) is processed by the system-to-be and how such data is related and collected from their owner. A Personal Information Diagram (PID) is done **for each biddable domain** that has been identified in the CD. To elaborate a PID, we must connect all phenomena that represent personal data of the considered biddable domain with a dependency of the type `<<relatedTo>>`[1]

This type of diagrams is generated semi-automatically [10]. That is, the **user** (i.e. a requirement engineer) must analyse the *phenomena* annotated in the edges of the DSIFD starting from the biddable domain that corresponds to the stakeholder as follows:

1. If the phenomenon is *symbolic*, the tool creates a connection between such phenomenon and the corresponding biddable domain in the domain's PID.
2. If the phenomenon is *causal,* then the user must first check whether it contains/transmits personal data and specify it as symbolic phenomenon. Then, the tool creates a connection between the phenomenon specified by the user and the corresponding biddable domain in the domain's PID.

*Example: **modifyEHR** and **createEHR** are causal phenomena connected to the biddable domain Doctor. Therefore, they must be specified as symbolic phenomenon. We will assume that both contain the symbolic phenomena **doctorContactInformation**, **doctorDetails**, **treatments**, **diagnosis**, and **notes**. Hence, we connect these phenomena to the Doctor in the Doctor's PID (Figure 6).*

[1] In the DSIFD, *referredTo* phenomena are translated as *inputs*, and *constrained* phenomena as *outputs*.

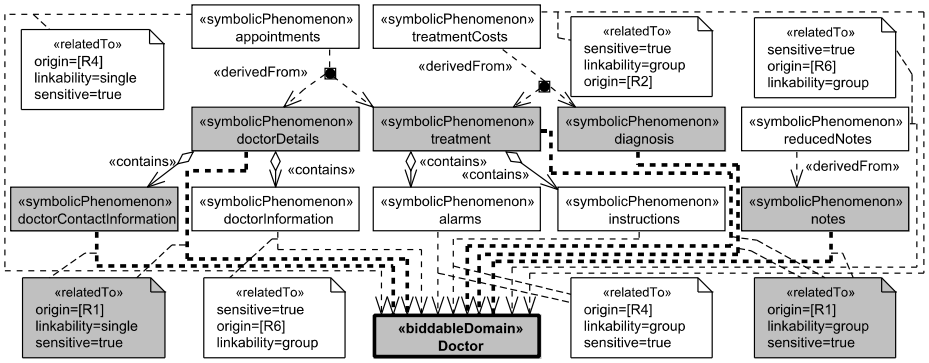

Figure 6: Personal Information Diagram [9]

3. Create a dependency of the type `<<relatedTo>>` for each connection in the PID. This is done for documenting the relation between the phenomenon and the biddable domain. The `<<relatedTo>>` stereotype provides the following attributes for documentation purposes:
   - `Origin`: This is automatically set to those statements which refer to the respective phenomenon.
   - `Sensitive`: This attribute is set to *true* when the information represented by the phenomena is of sensitive nature and *false* in any other case. This and the rest of the attribute values are specified by the user.
   - `Linkability`: The value *single* is used to indicate that the data can only identify the individual it belongs to, the value *subgroup* when it identifies a potential subgroup of individuals, and the value *anonymous* when it does not provide any link to the data subject.
   - `Collection`[2]: The value *direct* indicates that the stakeholder provides the information herself, the value *indirect* when the information is collected by observing the stakeholder's behaviour, *reused* when the data has been previously collected for another purpose (e.g. another project), and *external* in cases where the data is gathered through third parties.

*Example: The phenomenon **doctorContactInformation** has its origin in R6, is of sensitive nature, and has a single linkability. The initially identified relations for the Doctor are highlighted in grey.*

4. Symbolic phenomena can be related with each other when a phenomenon contains another phenomenon, or when it can be derived from another phenomenon. These relations between phenomenon are identified during a later iterative analysis using the stereotypes `<<derivedFrom>>` and `<<contains>>`.

2.1.2.5 Available Information Diagram (personal data flow analysis)

After the PID is made, we proceed to analyse how the personal data of the stakeholder flows through the system-to-be. This is done by analysing the stakeholder's available information in the different *statements* of the DISFG. That is, we create for every statement in the DISFG a projection of the symbolic phenomena included in the stakeholder's PID. Such projection is depicted through an Available Information Diagram (AID) of the stakeholder in the statement under consideration. Such AID consists of an *information* view and a *linkability* view which

[2] As it can be observed in Figure 6, this attribute is sometimes omitted in the literature. However, we consider it as mandatory within the context this work package because of its importance.

describe the information available in the statement and the relations between such information, respectively.

2.1.2.5.1 Initialization of Personal Data Flow Analysis

An initial *information* view of the AID is created at the statements which have the stakeholder as an input (Figure 7). Such AID consists of `<<availableAt>>` dependencies between the *statement* under consideration and the *symbolic phenomena* that represents the stakeholder's personal information (i.e. the ones included in the stakeholder's PID). The `<<availableAt>>` stereotype provides the following attributes for documentation purposes:

- `Duration`: How long the information shall be available at the statement. It can adopt the value *forAction* when the information is only available to perform an action, *untilDeleted* when it is deleted at some point in time which is not necessarily after an action is completed, or *unlimited* to express that once the information is available at the statement it will not be deleted.
- `Origin`: The statements from which the phenomena flows to the statement under consideration.
- `Purpose`: To which statements the phenomena is flowing to.

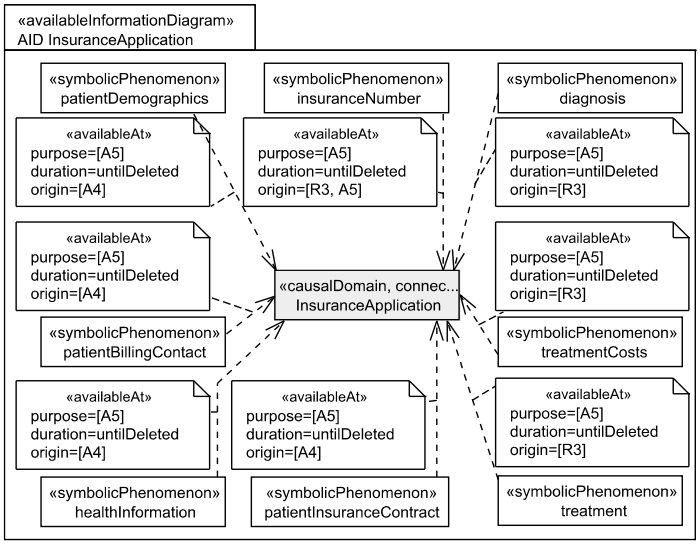

Figure 7: Available Information Diagram (information view) [7]

In the initialization phase, the duration attribute is set by default to *unlimited*, the origin is set to the statements from which the phenomenon is flowing from, and purpose is an *empty set*. These values must be checked by the user and modified when necessary. For the attribute purpose the user must decide for each output domain which of the available information is transmitted.

It may happen that the user identifies some symbolic phenomena that is inferred within the statement into consideration (e.g. the doctor's **reducedNotes** is derived from **notes**). This can be documented in the corresponding PID using the stereotypes `<<contains>>` or `<<derivedFrom>>` to indicate that a symbolic phenomenon is contained or derived from another one. Like the rest of the phenomena in the PID, the new phenomena must be connected to the corresponding biddable domain with a `<<relatedTo>>` association. In this case, the value *origin* is set to the statement under consideration.

2.1.2.5.2 Iterative Analysis of the Flow of Personal Data

After the initial AID is created, the user decides which other statements of the DSIFG she wants to investigate. That is, which and how personal information available at the statement under consideration flows to another statement. For this purpose, ProPAn creates a path of statements that still have to be considered. Such statements are the ones which appear in a path between the current statement and the statements the user is interested in. For those statements, an AID must be created.

The available information at a statement corresponds to those symbolic phenomena available at the *input statement* which have the statement into consideration as *purpose*. Therefore, when creating the AID of the statement under consideration, the *origin* of these available phenomenon is set to the corresponding *input statement*.

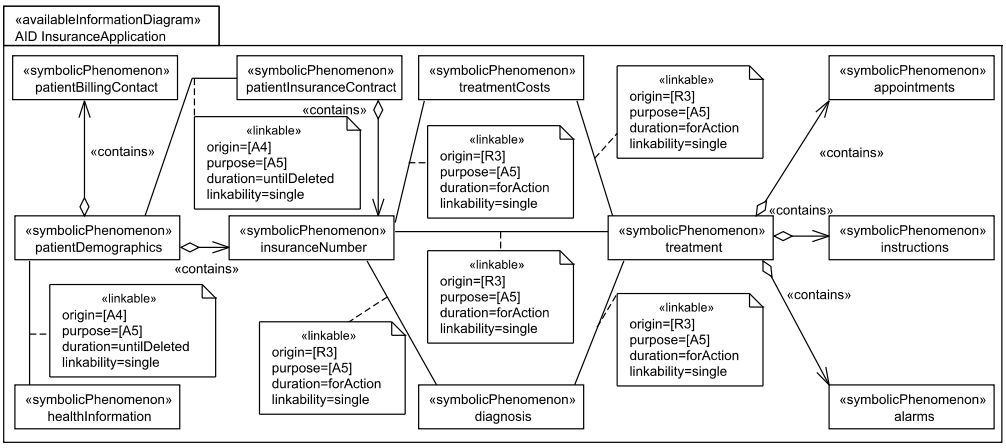

Figure 8: Available Information Diagram (linkability view) [7]

Another thing that is documented in an AID is how symbolic phenomena are linked with each other within the statement. That is, which and how the personal information available in the statement into consideration is linked with each other. This is documented in a *linkability view* of the AID (Figure 8) using associations with the stereotype `<<linkable>>` between the phenomena available in the statement's AID. The linkable stereotype offers the same attributes included in `<<availableAt>>` plus a *linkability* attribute which documents with which certainty the data can be linked to each other. It is worth mentioning that in this case, the attribute *purpose* represents which output statements are able to link the data.

2.1.3 Generation of Privacy Requirements

In this phase of the ProPAn method, privacy requirements are generated out of the privacy-relevant information flows identified in the first phase [1]. As illustrated in Figure 9, the method consists of four steps which are *Requirement Information Deduction*, *Generation of Privacy Requirement Candidates*, *Adjust Privacy Requirements*, and *Validate Privacy Requirements*. Overall, requirement candidates are generated for different privacy goals. Particularly, ProPAn follows the privacy protection goals introduced by Hansen [11], namely *confidentiality*, *integrity*, *availability*, *unlinkability*, *transparency*, and *intervenability*. For each goal, ProPAn provides a taxonomy (i.e. a metamodel) of requirements that can be instantiated using the information contained inside the *functional requirement artifacts* generated during the first phase (i.e. Problem and Context Diagrams, Domain Knowledge, DSIFDs, PIDs, and AIDs). Moreover, each requirement taxonomy provides a collection of semantic templates with placeholders that allow to document the generated privacy requirements. We will provide examples of taxonomies and semantic templates in the next sections.

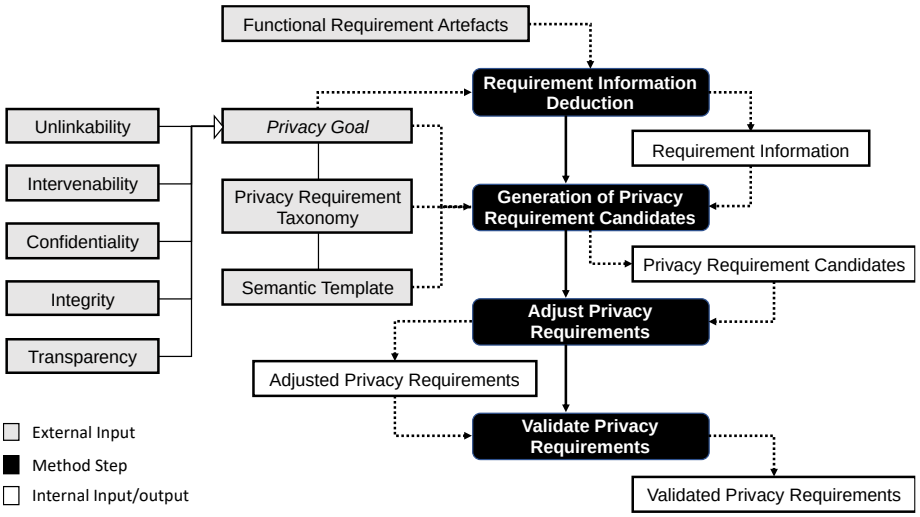

Figure 9: Generation of Privacy Requirements

In the first step, the information necessary to instantiate the taxonomy of a particular privacy goal is deduced using the *functional requirement artifacts* generated during the first phase of ProPAn. For instance, to generate unlikability requirements one must know which information from a stakeholder S should be undetectable for a counter-stakeholder C. This can be deduced using the PID of S and the AID of C. Once this requirement information is deduced, we can proceed to instantiate the corresponding taxonomy metamodel of the privacy goal under consideration and derive a set of requirement candidates using the semantic templates. This last derivation activity corresponds to the step *Generation of Privacy Requirement Candidates*.

Requirement candidates may be incomplete or result too strong/weak for the system-to-be under analysis. For instance, it may happen that an undetectability requirement can be relaxed allowing a counter-stakeholder to know that a piece of personal data exists in the system without disclosing the actual value of such data (i.e. replacing an *undetectability* requirement with a *confidentiality* one). For this reason, the user must review, complete and adjust the generated requirement candidates manually. Afterwards, privacy requirement candidates must be *validated* to check whether they are still consistent with the flow and availability of personal data at the different domains of the system-to-be. Depending on the outcome of this validation activity, some requirements will need to be adjusted and others accepted.

2.1.3.1 Privacy Requirement Taxonomies

Taxonomies are obtained from an in-depth analysis of the privacy principles introduced in ISO 29100 and the body of the GDPR. Both, standard and law, are analyzed through the lens of each of the privacy protection goals introduced by Hansen. Taxonomies consist of a hierarchy of meta-requirements expressed through UML class diagrams and implemented as UML profiles. The taxonomy of Figure 10 corresponds to the privacy goal *unlinkability* and refines it into the privacy meta-requirements pseudonymity, unlinkability and undetectability. In this case, the sub-requirement unlinkability has been further refined into data unlinkability (i.e. when certain personal data shall not be linkable to each other) and anonymity (i.e. when certain personal data shall not be linkable to the corresponding data subject).

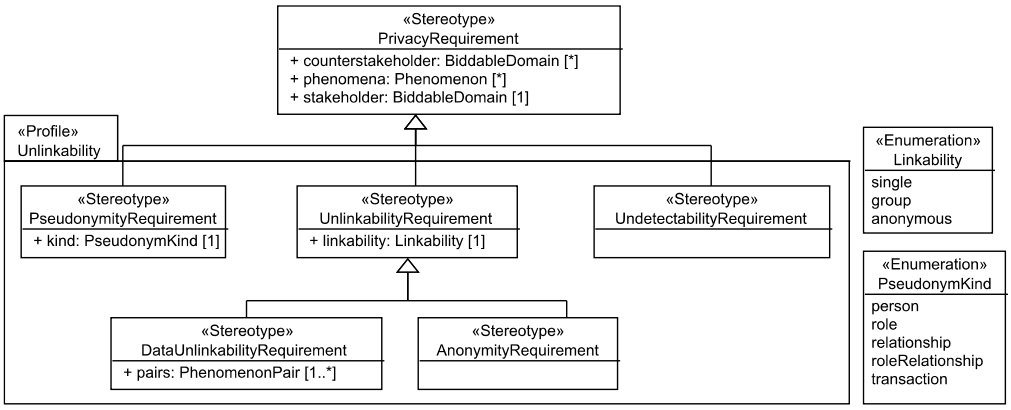

Figure 10: Unlinkability Requirements Taxonomy [7]

2.1.3.2 Semantic Templates

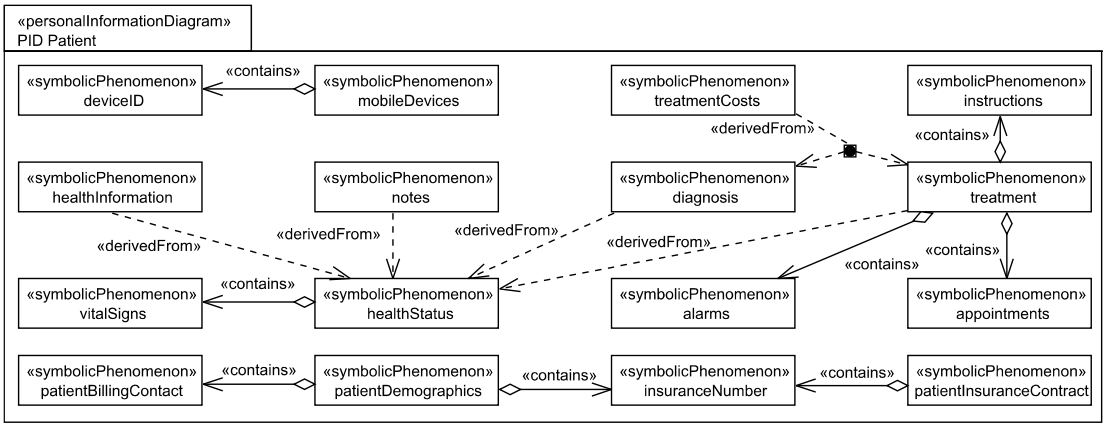

Figure 11: PID of the data subject Patient [7]

As already mentioned, the privacy requirement taxonomies can be instantiated by reasoning over the information contained in the functional requirement artifacts obtained during the first phase of the ProPAn method. For instance, let us assume that we want to generate an undetectability requirement for the stakeholder *patient* with regard to the counter-stakeholder *insurance employee*. We can use the information inside the patient's PID and the employee' AID to instantiate the corresponding class of the taxonomy. In this particular case, the *deduction strategy* consists of checking those phenomena in the patient's PID that are not available at the employee's AID. Then, we create an instance of **UndetectabilityRequirement** for such phenomena with the patient and insurance employee as the corresponding stakeholder and counter-stakeholder class attributes.

> The <*counterstakeholder*>s shall not be able to sufficiently distinguish whether the personal information <*phenomena*> of the <*stakeholder*> exists or not.

Figure 12: Semantic Template for Undetectability Requirements [7]

Figure 12 shows the semantic template corresponding to UndetectabilityRequirement. Basically, it is a text snippet with placeholders that refer to the attributes of the corresponding class in the taxonomy[3]. If we consider the PID of Figure 11 and the AID of Figure 8, we can observe that the phenomena *healthStatus*, *mobileDevices*, *deviceId*, *vitalSigns* and *notes* should be undetectable by the insurance employee. Hence, we create an instance of UndetectabilityRequirement that can be expressed as in Figure 13.

[3] This snippet in particular follows the *undetectability* definition introduced by Pfitzmann and Hansen [20]

The *insurance employee* shall not be able to sufficiently distinguish whether the personal information *healthStatus, mobileDevices, deviceId, vitalSigns, and notes* of the *patient* exist or not.

Figure 13: Undetectability Requirement Candidate [7]

2.1.4 Strengths and Limitations

As it can be observed, the first phase of ProPAn offers a collection of artefacts that allow software engineers to capture and document privacy-relevant knowledge in a system-to-be. In general, a software project can be expressed and decomposed into:

- 1 Context Diagram,
- N Problem Diagrams,
- 1 Detailed Stakeholder Information Flow Diagram (DSIFD),
- M Personal Information Diagrams (PIDs),
- 2*M Available Information Diagrams (linkability and information view),

from which only the DSIFD is generated automatically and the rest must be elaborated manually by engineers and privacy experts. This can result in large amounts of documentation, particularly in software projects of middle and large size. Consequently, the integration of ProPAn into an Agile development process may be difficult because of its documentation overhead. Moreover, the jargon adopted by the method (i.e. words like “biddable” and “lexical”) may alienate those who are not familiar with it and, consequently, hinder the method’s usability.

In order to apply ProPAn to PDP4E, we propose to reduce its documentation overhead on the first phase and employ a terminology closer to the one used in a software development

2.2 Other approaches to generate GDPR requirements

So far, several methods have been proposed for the generation of privacy and data protection requirements in software projects [1] [12] [13] [14] [15], each building upon a particular Privacy Conceptual Model (PCM) and Privacy Normative Framework (PNF). Basically, the PCM defines what privacy means in the context of the method (e.g. Hansen’s privacy goals, privacy principles), whereas the PNF represents binding regulations or privacy obligations to be considered (e.g. GDPR, ISO 29100) [10]. Methods employ different notations for the specification of requirements (e.g. textual, UML, use-case diagrams), and modelling languages for representing the system (or system-to-be) under consideration (e.g. BPMN, data-flow diagrams). Generally speaking, these methods consider privacy as a quality attribute or soft-goal that must be refined into a set of functional requirements [10]. For example, Dennedy et al. [15] propose to model a system-to-be through use cases and business activity diagrams enhanced with metadata related to the actors and information being processed by such system. In this approach, patterns (i.e. generic use cases) are used in combination with interpretation guidelines of the OECD privacy principles[4] to identify and instantiate privacy use cases. Such instances guide the selection of Privacy Enhancing Technologies (PETs) which are prescribed by the method for each OECD principle. Hoepman et al. [16] and Colesky et al. [14] elaborate on a set of privacy patterns for the development of software architectures that consider certain levels of privacy protection. Colesky et al. [14] identified a set of privacy protection needs from the body of the GDPR and other legal

[4] http://oecdprivacy.org

frameworks like the Privacy Shield Agreement[5]. Such protection needs are refined into privacy design tactics and linked to specific privacy patterns (represented in natural language) and PETs.

Requirement engineering methods can also contemplate the identification of privacy threats and the definition of functional requirements that mitigate privacy risks [10]. For instance, LINDDUN [13] is a framework for privacy threat analysis that extends the security approach proposed by the STRIDE [12] method. Overall, LINDUNN differentiates between *hard privacy properties* such as unlinkability, anonymity and confidentiality; and *soft privacy properties* like consent awareness and compliance. The method proposes to model the system-to-be as a DFD with trust boundaries to differentiate those information flows which are subject to threat analysis to the ones that are not. Then, misuse cases are identified for those untrusted data flows based on i) LINDDUN's privacy threats (which are negations of the privacy properties), ii) a mapping between DFD elements and such privacy threats, and iii) threat trees patterns which are similar to attack trees. Another privacy- and security- by design methodology is PRIPARE [17]. This method identifies the functional requirements of a system-to-be together with stakeholders, sub-systems, domains and the relations between them. Additionally, it identifies which personal data is ought to be processed by the system and up to which extent the method should be applied, this last one depending on the size of the organization. Based on this model of the system, a checklist is used to identify relevant privacy principles that must be considered and select initial privacy controls. Following, the identified principles are refined into privacy requirements that represent their conformance criteria. This activity is supported through a collection of conformance criteria for all privacy principles of ISO 29100. LINDDUN can be applied in combination with PRIPARE to carry on the corresponding risk elicitation and treatment selection activities.

Privacy regulations and the obligation to comply with privacy laws are driving factors for considering privacy as a software quality [10]. Privacy principles and goals are often introduced in requirement engineering methods to make these obligations more accessible and comprehensible for practitioners in the computer science domain. Privacy principles and regulations are also considered as guidelines during the ProPAn method to refine privacy goals such as transparency and intervenability. In this sense, ProPAn is not limited to particular principles and regulations such as the ISO 29100 and the GDPR but expects to be adaptable to other privacy legislations and standards. Regarding data representation, text-based approaches like PRIPARE are easier to adapt in the practice since users are not required to learn a specific modelling notation or formal language [10]. Moreover, Meis et al. [10] show in a literature review on requirement engineering methods for privacy that a large number of methods rely solely on textual documentation. However, informal notations make automation and consistency checking harder to perform, hence, a model-driven approach is preferable for these purposes.

[5] https://www.privacyshield.gov/

3 Requirement Elicitation Method for PDP4E

So far, we have introduced the ProPAn method for capturing a system's functional requirements and generate the corresponding privacy requirements. We have discussed the method's strengths and limitations in the context of PDP4E. In this section, we present a requirement engineering method for PDP4E which is inspired in the principles outlined by ProPAn. That is, the definition of privacy requirement meta-models and artifacts for capturing the system's privacy-related functionalities. We emphasize the need for coverage of the data protection principles introduced in the GDPR in order to achieve compliance. For this reason, we elaborate on a protocol oriented to systematically generate requirement meta-models (also referred as meta-requirements) from legal sources. This includes the extraction of the fundamental notions that are introduced in a legal document to create an ontology of terms that can pertain to the legal arena or technical terms introduced by the legal document (e.g. data subject, processor, controller, process, consent, right, lawful). Such ontology or vocabulary will set the basis for the elaboration of requirement meta-models.

3.1 Method overview

Figure 14 illustrates the proposed requirement elicitation method for PDP4E. As it can be observed, this method is an extension of the second phase of ProPAn "Generation of privacy requirements". Hence, we assume as input either a set of software artifacts like the ones generated in the first phase of ProPAn (i.e. problem diagrams, domain knowledge, DSIFD, etc.), or a data structure containing the same information. As we mentioned in section 2.1.4, this last one consists of a DFD annotated with privacy-related information equivalent to the elicited using PIDs and AIDs (e.g. duration, origin, and purpose of personal information). Since DFDs are data structures that are planned to be used in the different WPs of PDP4E, we will elaborate an integrated DFD vision that considers the different viewpoints and necessities of the different WPs. For instance, WP3 elaborates on the LINDDUN method for risk assessment which makes use of DFDs to analyse the privacy threats in a system (see deliverable D3.4). The resulting DFD if this WP is expected to be aligned with the one of WP3 to provide a common and consistent interface between them.

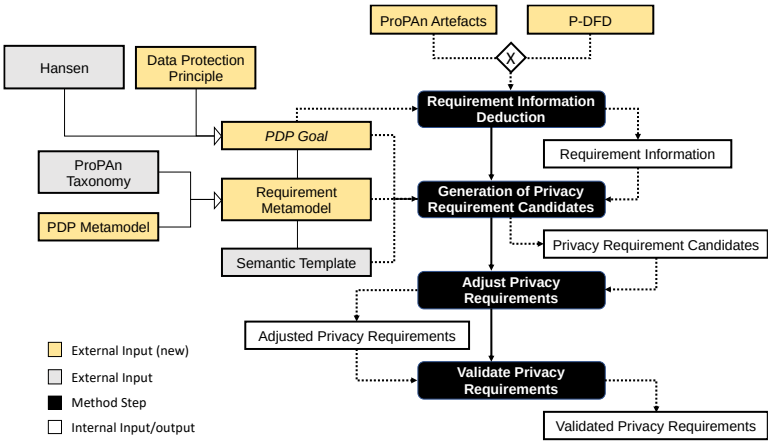

Figure 14: PDP4E Requirement Elicitation Method

Another extension point introduced in the PDP4E method corresponds to the meta-models used to generate privacy and data protection requirements. As we described in sections 2.1.3.1 and 2.1.3.2, privacy requirements in ProPAn are derived using requirement taxonomies and semantic templates. Such taxonomies and templates are created by conducting an analysis of the GDPR and the ISO 29100. Such analysis is performed through the lens of the Hansen's privacy goals.

That is, both law and standard are parsed into a set of taxonomies and semantic templates that represent each of the Hansen's goals. For instance, in [18] Meis elaborates on the requirement taxonomy corresponding to the goal intervenability. Likewise, in [7] the authors introduce taxonomies corresponding to the goals unlinkability and transparency. We believe that, for compliance purposes, such approach might pass over some critical aspects included in the sources of such taxonomies. That is, although the authors claimed having analysed the GDPR and ISO 29100 using the Privacy goals of Hansen, there is no guarantee that this strategy is sufficient for covering all the legal requirements introduced by these sources. This is the case of Article 8 of the GDPR which prescribes legal obligations when processing children's data. Expressing GDPR requirements only in terms of a limited set of principles (e.g., intervenability, unlikability and transparency) introduces a risk of methodological bias on the choice of principles. For this reason, we introduce two extension points to achieve coverage and correctness during requirements elicitation. The first one is a protocol for extracting legal notions from the GDPR to capture in a structured vocabulary those concepts that can help us modelling *Data Protection Principles* (such protocol is described in Section **¡Error! No se encuentra el origen de la referencia.**). The second one is the possibility of adding new taxonomies or meta-requirements to the ones already introduced by ProPAn in order to achieve full coverage and avoid any potential bias. This is done by adding the extension point *PDP Goal* in the requirement elicitation method of Figure 14, which considers the incorporation of new data protection principles complementary to the goals of Hansen. Therefore, we consider the new PDP4E taxonomies as Requirement Meta-models and propose representing new data protection principles through PDP Meta-models. Such PDP Meta-models follow the same principle of the taxonomies introduced by ProPAn but instead of being associated with a privacy goal, they are associated to a privacy principle and defined using the vocabulary of legal notions.

3.2 Method for elicitation of GDPR requirements

As explained in previous sections, the method proposed in PDP4E for elicitation of requirements pursues the main following objectives:

1. Consider, circumvent and inherit the most salient features of ProPan. Referred features are a basis upon which the PDP4E method is defined
2. Define additional method features in order to elicit requirements from GDPR ensuring a certain coverage, the integration of main GDPR specificities and avoiding potential biases.
3. Leverage Model Driven Engineering (MDE) approaches in order to support non-savvy privacy engineers during the elicitation of requirements to ensure privacy and data protection. The leveraging is conducted following an "as simple as possible" policy.

To achieve these objectives, and following the MDE perspective, a meta-model needs to be defined. A meta-model is a model that captures and structures the fundamental concepts and notions of a given problem space. Problem spaces are often related to application domains or even specific use cases where a language, knowledge and domain expertise meet all together. The meta-model is thus a mean to structure referred elements and constitutes a basis for reasoning w.r.t. addressed concerns and possible ways to analyze and find solutions. In the case of PDP4E, our main source to extract and structure fundamental notions is the GDPR. As it is shown in Figure 15, three main outcomes are expected from the analysis of GDPR: a meta-model,

new taxonomies aligned to the ones already defined in ProPan, and a mapping to elicit requirements. Regarding the meta-model, it is build following a protocol that is described in the following subsection 3.2.1. Along with meta-model definition, the protocol also helps to create new taxonomies of requirements (also called categories).

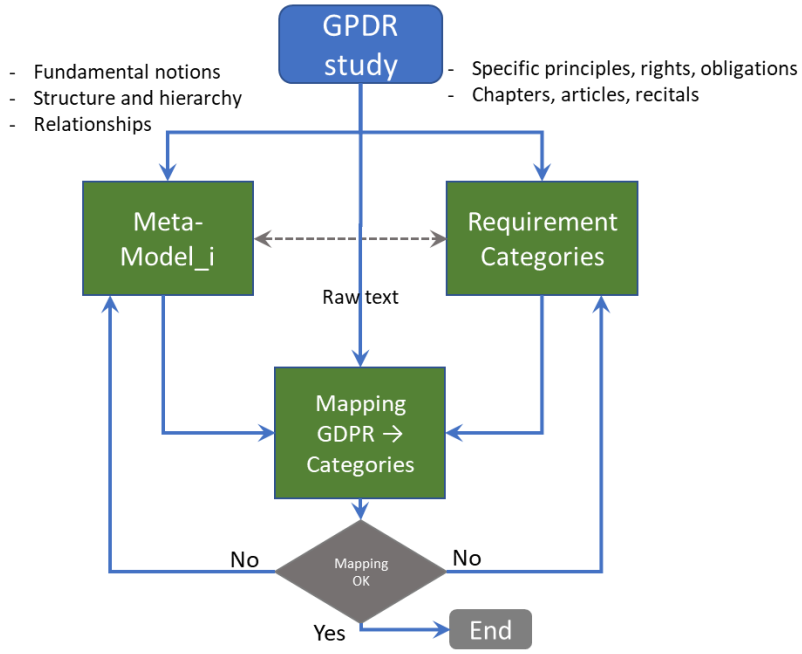

Figure 15. Main relationships between PDP4E meta-model and GDPR taxonomies

After a first lecture of GDPR, it is expected that the taxonomies will be aligned with the principles in GDPR (Processing of personal data, Lawfulness, Consent, Child's consent, Special categories, etc.). Each category shall be defined by a set of "signatures" which are related to principles. Each signature follows a syntactical pattern "*If pre-conditions, the <system> shall post-conditions*" [19]. Since these patterns can be instantiated for a specific system-to-be or use case, they are also referred as meta-requirements. The *Pre-conditions* and *post-conditions* in the meta-requirements include explicit references to the fundamental notions within the meta-model. Those correlations allow to introduce behavioral and semantical models[6] useful in particular when fine-grained/detailed versions of requirements are specified. Pre and post-conditions can be typed by certain kinds of abstract elements found in GDPR like actions (inform, ask for consent), subjects (process, data subject), and roles (controller, processor). Action types can in turn be associated with an operational semantics (behavioral models), a level of stringency (mandatory, optional), and temporal attributes (occurrence interval, validity period). The mapping between the meta-model and the taxonomies is defined by construction. The mapping is later reused during the elicitation of requirements for a specific system-to-be or use case.

3.2.1 Protocol for GDPR meta-model creation

The meta-model for requirements elicitation aims to ensure the integration of GDPR specificities and its coverage whereas the potential biases should still be avoided. To ensure these features, the protocol shown in Figure 16 is followed. After a lecture and analysis of GDPR, a first version of the meta-model is obtained. This version is to be validated w.r.t. the taxonomies including the meta-requirements (*If pre-conditions then post-conditions*) which are accordingly written to

[6] For now, referred behaviours and semantics have not been selected and are out of the scope of PDP4E.

cover GDPR articles, recitals and paragraphs. The referred validation occurs via the expression of meta-requirements in terms of the elements within the meta-model. The meta-model remains "as is", as long as it is complete and suitable enough to specify the meta-requirements for each category. Otherwise, the meta-model should evolve. The potential evolutions may consist in:

(a) adding GDPR elements (notions, concepts, etc.) necessary to specify the meta-requirements,
(b) adding auxiliary elements (e.g., abstract classes) necessary to specify the meta-requirement,
(c) adding/removing associations, dependencies, between existing elements in the meta-model,
(d) integrating new attributes to existing elements,
(e) defining or redefining the elements rationale.

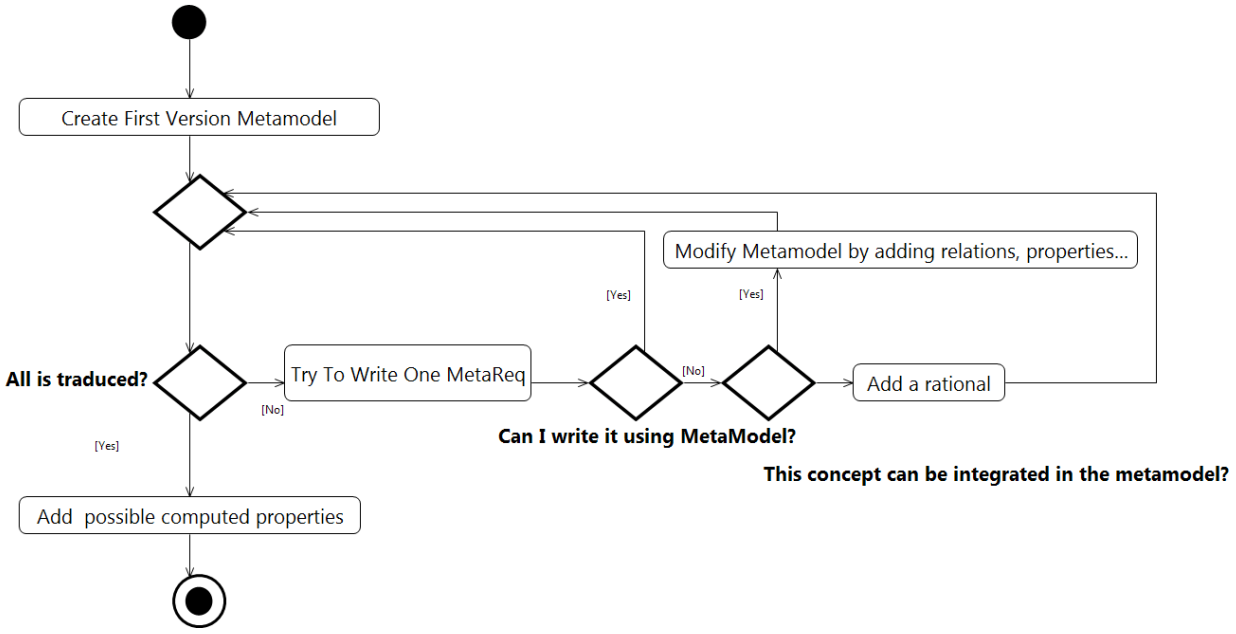

Figure 16. Protocol adopted in PDP4E for GDPR meta-model creation

For now, the evolutions are manual since they aim to introduce new conceptual, structural and semantical elements thus capturing the essential contents of GDPR. In a first approach, all parts of the regulation are considered. However, according to PDP4E goals, we are mainly interested on parts that finally impact the typical systems and software engineering development cycle. More concretely, the rules concerning stakeholders in the development cycle are primarily in the scope of the meta-model whereas those involving governments (country, states, cities) are potentially out of scope. Since there is no such as a unique meta-model and its optimality is rather hard to achieve/prove, the coverage of GDPR is, for now, the main criterion for meta-model design. In addition, since the ISO 29100 privacy framework is closer to the technical arena and its jargon more familiar to engineers, a potential alignment with the standard is foreseen. During the consolidation phases of the project, it is expected that legal experts, an in particular KUL partners, conduct an internal review in order to consolidate this work.

3.3 Definition of PDP Meta-models

Following the MDE approach, the meta-model is developed relying upon UML Class diagrams. A UML Class includes three compartments including attributes, methods, and nested-classifiers. The attributes are defined with a name and selecting a basic type, like string, integer, double,

boolean, or a user defined type. The attributes support the definition of variables and parameters within the model. The methods in the second compartment can convey information about functions, interfaces, and also behaviors. Inputs and outputs of methods can also be declared relying upon basic and user-defined types. The compartment for nested classifiers can contain different UML classifiers which become an internal part of the Class. The nested classifier compartment can be used to store behavioral diagrams like UML activities or sequences thus adding -informal- operational semantics. Figure 17 shows an overview of the UML Class diagram containing the meta-model of GDPR, Article 5. This model shows a generic element named *Principle* that refers in particular to *Data Subjects*. As they are specified in GDPR, the *Principles* can be classified into two categories, one related to *Processes* and the other related to *Personal Data* and *Purposes* for processing them. Within the *Process* category, the principles settle properties for the *Processes* to be endorsed with, like for instance *Lawful*, *Fairly*, *Transparency*. Regarding the *Personal Data* and *Purposes* category, the principles refer to properties related to *Data Minimization*, *Purpose Limitation*, *Accuracy*, and *Storage Limitation*. The principles can be documented and their textual specifications stored within the meta-model. The main attributes of the principles are defined as Boolean variables. This choice aligns the meta-model for the specification and instantiation of meta-requirements as it is explained in the following subsection 3.4.

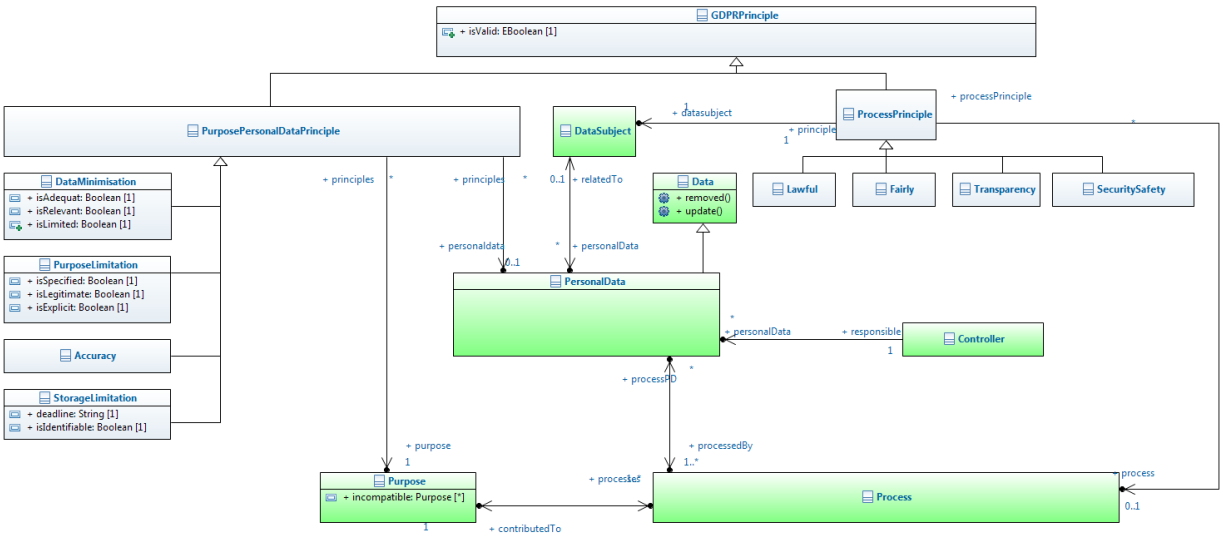

Figure 17: Meta-model related to GDPR, Article 5

The associations in the meta-model appear as connectors between elements (directed and non-directed). The associations come with a given semantics to express different kinds of relationships, for instance, associations can express a hierarchy between general and specific notions (white arrows). Some associations support quantifiers useful to constraint the number of elements concerned by the association. This helps to express one-to-many, many-to-many relationships. The semantics of directed associations declare a sense for navigability also useful for meta-model lecture. The features of the meta-model are later reused and leveraged during the implementation of the language as a profile.

3.4 Definition of GDPR meta-requirements

The meta-model of GDPR defines a language including the fundamental notions, their structure and main attributes. Thus, it is more than a list of terms or vocabulary: the meta-model includes potential hierarchies between concepts, dependencies, and relational constraints. In PDP4E, we

plan to exploit UML meta-model features as well as other MDE mechanisms in order to create a set of meta-requirements. As previously explained, the meta-requirements adopt the syntax "*If pre-conditions then post-conditions*". For the meta-requirements to be defined, it is assumed that *pre-conditions* and *post-conditions* can be typed. Among the instances of typed elements, we can mention the following (non-exhaustive list):

- **Actions:** process, request, provide, send,
- **Subjects:** data subject,
- **Roles:** processor, controller, data protection officer,
- **Objects:** data, personal data, sensitive data
- **Abstract elements:** purpose, tasks, process
- **Qualities, properties:** lawfully, fairly, transparency.

Given the syntax *"If pre-conditions then post-conditions",* one of the main goals during the definition of meta-requirements is to find patterns that correlate typed elements in pre-conditions with those specified in post-conditions. The identified patterns shall help to define categories related to GDPR principles. The process of meta-requirements definition is illustrated in the following instance.

The text below is an excerpt from GDPR, article 5, paragraph (a).

> Personal data shall be:
> (a) processed lawfully, fairly and in a transparent manner in relation to the data subject ('lawfulness, fairness and transparency');

The rule states that, in all cases, the processes involving personal data shall exhibit certain desired properties. To translate it as a meta-requirement, the pre-conditions and post-conditions are first identified. In this case, the statement refers to all processes involving personal data as pre-condition. The elements for *process* and *personal data* are thus located within the meta-model. As post-condition, it is expected that all *processes* referred in the pre-condition are endorsed with *lawful*, *fairly* and *transparent* properties. The respective elements should also be found within the meta-model. Of course, in case of missing elements, the meta-model should be accordingly completed (meta-model evolution). Finally, the meta-requirement is written by following the structure, relationships and attributes of concerned elements inherited from the meta-model. The instance below shows the meta-requirement associated to the GDPR paragraph above.

> **IF** process **<self.processPD.size()>0** process personal Data of **< self.processPD>** **THEN** the Process **<self>** shall be lawfull **<self.principles<LawFull>(self.processPD.DataSubject)->.value=true**, fairly **< self.principles<Fairly>(self.processPD.DataSubject)->.value=true>** and transparent **<self.principles<Transparency>(self.processPD. DataSubject)->.value=true >.**

The process of meta-requirements specification can be assisted relying upon the meta-model features and UML mechanisms, e.g., links to meta-model elements, lists of available attributes, default values, etc. The meta-requirement syntax already shows some of those mechanisms. In particular, the elements within placeholders "< >" make reference to meta-model elements and internal attributes. The structure of meta-requirements is aligned and facilitates other development phases like requirements validation. Indeed, as a basic but useful mechanism, the properties specified as boolean attributes can be set to true thus reflecting that the quality has been endorsed.

4 Summary

In this document we have specified a first draft of a method for requirements elicitation in the context of PDP4E. In a first approach, the method considers ProPan as its main background and basis. ProPan is indeed a method for requirements elicitation that relies upon (1) a collection of diagrams containing contextual and system-to-be information, (2) taxonomies defined via privacy principles and patterns for inference, and (3) semi-automatic/interactive mechanisms used to infer privacy-related requirements given an instance of a system-to-be. Nonetheless, ProPan exhibits some drawbacks mainly regarding its complexity, evaluated in terms of quantity, contents, and jargon of diagrams. The proposed method aims to circumvent and inherit the most salient features of ProPan. Moreover, additional features are defined in order to elicit requirements from GDPR ensuring a certain coverage, the integration of main GDPR specificities and avoiding potential biases. To do so, the method leverages some MDE techniques to support non-savvy privacy engineers during the elicitation of requirements. The method mainly relies upon a meta-model of GDPR, a set of meta-requirements correlated to the meta-model, and new taxonomies defined by patterns of typed requirements. The progress in the method and tool architecture specifications show that the approach for requirements elicitation is feasible. Since some of the phases need to be implemented to ensure feasibility as well as harmonization with other PDP4E tools and methods, the approach will be deployed and accordingly consolidated.

5 References

[1] K. Beckers, S. Faßbender, M. Heisel and R. Meis, "A Problem-Based Approach for Computer Aided Privacy Threat Identification," in *Privacy Technologies and Policy*, 2014.

[2] European Comission, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46," *Official Journal of the European Union (OJ),* vol. 59, p. 294, 2016.

[3] M. Jackson, Problem frames: analysing and structuring software development problems, Addison-Wesley, 2001.

[4] International Organisation for Standardization and International Electrotechnical Comission (ISO/IEC), "ISO/IEC 29100:2011 Information Technology-Security Techniques-Privacy Framework," [Online]. Available: https://www.iso.org/standard/45123.html. [Accessed 3 June 2019].

[5] Network of Excellence (NoE) on Engineering Secure Future Internet Software Services and Systems (NESSoS), [Online]. Available: http://http://www.nessos-project.eu/. [Accessed 3 June 2019].

[6] D. Hatebur and M. Heisel, "A UML profile for requirements analysis of dependable software," in *International Conference on Computer Safety, Reliability, and Security*, 2010.

[7] R. Meis and M. Heisel, "Computer-Aided Identification and Validation of Privacy Requirements," *Information,* vol. 7, no. 2, pp. 1-32, 2016.

[8] R. Meis and M. Heisel, "Systematic identification of information flows from requirements to support privacy impact assessments," in *10th International Joint Conference on Software Technologies (ICSOFT)*, 2015.

[9] R. Meis and M. Heisel, "Supporting privacy impact assessments using problem-based privacy analysis," in *10th International Joint Conference on Software Technologies(ICSOFT)*, 2015.

[10] R. Meis, *Problem-based Privacy Analysis (ProPAn): Computer-aided Privacy Requirements Engineering Method (PhD. Thesis),* 2018.

[11] M. Hansen, M. Jensen and M. Rost, "Protection goals for privacy engineering," in *IEEE Security and Privacy Workshops (SPW)*, 2015.

[12] M. Howard and S. Lipner, The security development lifecycle, vol. 8, Microsoft Press Redmond, 2006.

[13] M. Deng, K. Wuyts, R. Scandariato, B. Preneel and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering,* vol. 16, no. 1, pp. 3-32, 2011.

[14] M. Colesky, J.-H. Hoepman and C. Hillen, "A critical analysis of privacy design strategies," in *IEEE Security and Privacy Workshops (SPW)*, 2016.

[15] M. Dennedy, J. Fox and T. Finneran, *The Privacy Engineer's Manifesto: Getting from Policy to Code to QA to Value,* Apress, 2014.

[16] J.-H. Hoepman, "Privacy design strategies," in *IFIP International Information Security Conference*, 2014.

[17] N. Notario, A. Crespo, Y.-S. Martín, J. M. Del Alamo, D. Le Métayer, T. Antignac, A. Kung, I. Kroener and D. Wright, "PRIPARE: Integrating privacy best practices into a privacy engineering methodology," *IEEE Security and Privacy Workshops (SPW),* pp. 151-158, 2015.

[18] R. Meis and M. Heisel, "Computer-Aided Identification and Validation of Intervenability Requirements," *Information,* vol. 8, no. 1, pp. 1-30, 2017.

[19] K. Pohl and C. Rupp, Requirements engineering fundamentals: A study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant, Rocky Nook, Inc., 2016.

[20] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," 2010.