

PDP-ReqLite: A lightweight approach for the elicitation of privacy and data protection requirements

Nicolás E. Díaz Ferreyra, Patrick Tessier, Gabriel Pedroza, Maritta Heisel

▶ **To cite this version:**

Nicolás E. Díaz Ferreyra, Patrick Tessier, Gabriel Pedroza, Maritta Heisel. PDP-ReqLite: A lightweight approach for the elicitation of privacy and data protection requirements. Data Privacy Management, Cryptocurrencies and Blockchain Technology. DPM 2020, CBT 2020., pp.161-177, 2020, 978-3-030-66172-4. 10.1007/978-3-030-66172-4_10 . cea-03264121

HAL Id: cea-03264121

https://cea.hal.science/cea-03264121

Submitted on 17 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PDP-ReqLite: A Lightweight Approach for the Elicitation of Privacy and Data Protection Requirements

Nicolás E. Díaz Ferreyra[1a], Patrick Tessier[2b], Gabriel Pedroza[2c], and Maritta Heisel[1d]

[1] Department of Computer Science and Applied Cognitive Science, University of Duisburg-Essen, Oststr. 99, Duisburg, 47057, Germany
{[a]nicolas.diaz-ferreyra, [d]maritta.heisel}@uni-due.de
[2] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
{[c]gabriel.pedroza, [b]patrick.tessier}@cea.fr

Abstract. With the introduction of the EU General Data Protection Regulation (GDPR), concerns about compliance started to arise among software companies inside and outside Europe. In order to achieve high compliance, software developers must consider those privacy and data protection goals defined across the different legal provisions in the GDPR. Prior work has introduced methods to systematically extract taxonomies of privacy requirements out of the GDPR's legal provisions. That is, a hierarchy of meta-requirements that can be instantiated for each specific software project. Particularly, ProPAn is a requirements elicitation method which leverages such taxonomies with the aim of achieving high levels of compliance. However, despite of its benefits, the method presents a high documentation overhead and redundancy across the artifacts it generates. In this work, we introduce a lightweight method named PDP-ReqLite initially inspired from ProPAn that introduces new artifacts for the documentation of personal data and information flows in a system-to-be. The purpose of PDP-ReqLite is to improve usability and applicability by reducing documentation overhead and complexity, and by introducing means to automate tasks, *e.g.*, automated requirements elicitation. In particular, this improved method provides additional features for incorporating new meta-requirements thus enlarging existing taxonomies.

Keywords: GDPR · privacy requirements engineering · data protection

1 Introduction

Nowadays, developing privacy-aware software systems has become a challenge of public interest. Legal frameworks such as the EU General Data Protection Regulation (GDPR) [17] have triggered major concerns about how information systems should implement data-protection functionalities and safeguard the 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. Particularly, there is a growing understanding that privacy must be considered as a primary aspect in every system and software development process. That is, it must be considered right from the early stages of a systems' design process and throughout its whole life cycle. Therefore, under this premise, an adequate elicitation of privacy requirements results critical for developing software systems that guarantee the data protection rights of their end-users.

Privacy requirement engineering methods seeks to define, and document requirements related to privacy and data protection for their later implementation. Overall, these methods define a set of privacy and data protection goals like transparency or integrity which guide the elicitation process of such requirements. Furthermore, many of them introduce conceptual elements extracted from current standards or legal provisions in order to achieve high levels of compliance. Such is the case of ProPAn, a model-driven method for the elicitation of privacy requirements that incorporates principles introduced by well-established data protection frameworks and standards. Particularly, ProPAn comprises a collection of meta models (or requirements taxonomies) that are systematically extracted from the body of the EU GDPR and the ISO 29100. Such meta models guide the elicitation process of privacy requirements taking as input a collection of functional requirements that the system-to-be is expected to meet.

All in all, requirements engineering methods use 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). In the particular case of ProPAn, functional requirements are specified using Jackson's Problem Frames notation. Such approach consists of describing the contextual elements with which the system-to-be must interact. In principle, this approach is suitable for the systematic elicitation of functional requirements. Nevertheless, it does not provide adequate means for a fine-grained documentation of the personal information that the system under consideration is expected to process. Hence, the method introduces a considerable amount of documentation overhead in order to document and analyze the information flows of the system and, thereby, generate the corresponding privacy and data protection requirements. Furthermore, even when such requirements are successfully generated, a high level of compliance cannot be ensured after their incorporation since the meta requirements used by the method do not cover all the legal provisions stated in the GDPR.

This paper introduces PDP-ReqLite, a lightweight method for the elicitation of privacy and data protection requirements. Particularly, PDP-ReqLite seeks to overcome the drawbacks identified in ProPAn regarding documentation overhead and compliance through new elicitation artifacts and requirements meta models. Such artifacts consist of data-flow and personal information diagrams that are independent from the Problem Frame notation employed by ProPAn but allow capturing the necessary information for the generation of privacy requirements. On the other hand, PDP-ReqLite introduces the possibility of defining additional

meta requirements and models for achieving a broader scope and compliance with legal provisions and standards. Furthermore, a protocol for the systematic elicitation of such meta requirements is also introduced and illustrated alongside the different steps of PDP-ReqLite.

The rest of the paper is structured as follows. Section 2 gives the related work. The Section 3 introduces the theoretical background necessary to understand the proposed approach. The PDP-ReqLite method is presented in Section 4. The method applicability is demonstrated relying upon a Smart Grid case study in Section 5. The conclusions and perspectives finally come in Section 6

2 Related work

So far, several methods have been proposed for the generation of privacy and data protection requirements in software projects, each employing different notations and modelling languages [1, 8, 4, 3, 5]. Generally speaking, these methods consider privacy as a quality attribute or soft-goal that must be refined into a set of functional requirements [12]. For example, Dennedy et al. [5] 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 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. In line with this, Hoepman et al. [7] and Colesky et al. [3] elaborated a set of privacy patterns for the development of software architectures considering certain levels of privacy protection. Particularly, Colesky et al. [3] identified a set of privacy protection needs from the body of the GDPR and other legal frameworks like the Privacy Shield Agreement and refined them into a collection of privacy design tactics. Such tactics are then linked to specific privacy patterns (represented in natural language) and PETs for their later application in the design of software architectures.

Privacy regulations and the obligation to comply with privacy laws are driving factors for considering privacy as a software quality [12]. 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. Furthermore, many of these approaches have been coined under the principles and premises of model-driven software engineering [12]. This is the case of ProPAn, a method for the systematic elicitation of privacy and data protection requirements which is driven by legal provisions and privacy goals related to transparency and intervenability, among others. Particularly, ProPAn is grounded in the provisions and guidelines included in the EU GDPR [17] and the ISO 29100 standard [9]. Nevertheless, the method exhibits extension points that makes it adaptable to other privacy legislations and data protection standards. Regarding data representation, text-based approaches like PRIPARE [14] are easier to adapt in the practice since users are not required to learn a spe-

cific modelling notation or formal language. Moreover, Meis et al. [12] 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.

3 Theoretical background

In this section, we describe the theoretical foundations upon which PDP-ReqLite is elaborated. Particularly, a brief introduction to ProPAn and its most salient software artifacts is given in the following subsections. Alongside, an analysis of the benefits and drawbacks of this method is conducted in order to identify potential areas of improvement.

3.1 The ProPAn Approach

ProPAn [1] is a systematic method that helps identifying the privacy needs of a software system based on a set of functional requirements. Particularly, one of the core characteristics of ProPAn is the elicitation of functional requirements using *problem diagrams*. Such diagrams where introduced by Michael Jackson [10] 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.) connected by interfaces through which they exchange certain *Phenomena* (i.e. events, actions, messages, and operations). Overall, the ProPAn method consists of two phases: *Identification of Privacy-Relevant Information Flows* and *Generation of Privacy Requirements*. In the first phase, the functional requirements of the system-to-be are identified and expressed through a collection of problem diagrams. In addition, the phenomena specified in such diagrams is further analysed and described in terms of personal information and data flows. For this, ProPAn introduces elicitation instruments additional to the ones proposed by Jackson with the purpose of documenting the personal information being processed by the system along with their corresponding data subjects (or system stakeholders) and data flows. These Functional Requirement Artifacts (FRAs) generated during the first stage of ProPAn become the input for the method's second stage.

In the second stage of the method, privacy requirement candidates are generated using a set of taxonomies that reflect privacy principles included in the GDPR and the ISO 29100 [9]. Overall, requirement candidates are generated for different privacy goals. Particularly, ProPAn follows the privacy protection goals introduced by Hansen [6], namely confidentiality, integrity, availability, unlinkability, transparency, and intervenability [1]. For each goal, the method provides a taxonomy (i.e. a metamodel) of privacy requirements and a collection of semantic templates with placeholders for their documentation. For example Fig 1 depicts

the taxonomy corresponding to the privacy goal unlinkability and the semantic template used to generate undetectability requirements. As it can be observed, the information available in the taxonomy (i.e. the attribute values) is leveraged by the semantic template to instantiate the corresponding requirement.

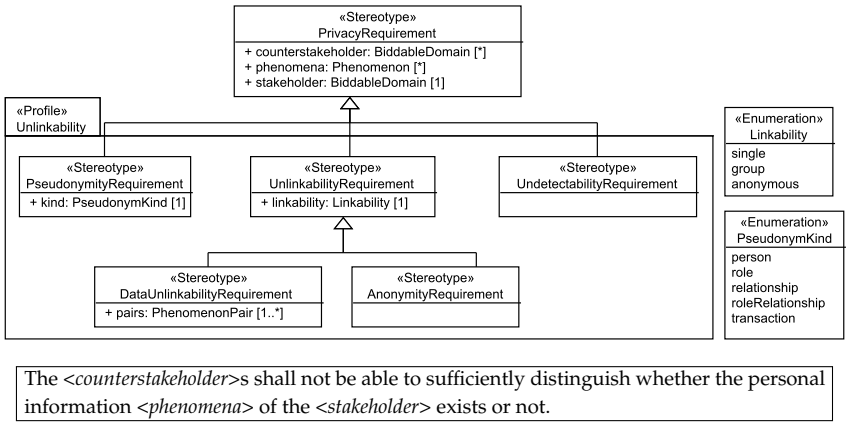


Fig. 1. Unlinkability Requirements Taxonomy (up) and Semantic Template for Undetectability Requirements (down) [13]

The second stage of ProPAn consists of four steps which are *Requirement Information Deduction*, *Generation of Privacy Requirement Candidates*, *Adjust Privacy Requirements*, and *Validate Privacy Requirements*. In the first step, the information necessary to instantiate the taxonomy of a particular privacy goal is deduced from the FRAs of the first stage. 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 by analysing the data flows and stakeholder information obtained during the first stage of ProPAn [12]. Once this information is deduced, a set of requirement candidates can be derived by instantiating the taxonomy and semantic templates of the privacy goal under consideration (Step 2). Since requirement candidates may be incomplete or result too strong/weak for the system-to-be under analysis, the user must review, complete and adjust the generated requirement candidates manually (Step 3). For instance, it may happen that an undetectability requirement must be relaxed and replaced with a confidentiality one. Once these adjustments are done, privacy requirement candidates are validated to check whether they remain consistent with the flow and availability of personal data at the different domains of the system-to-be (Step 4). Depending on the outcome of this validation activity, some requirements will be accepted and others may need to be adjusted.

3.2 Requirements Elicitation Artifacts

As described in Section 3.1, the first phase of ProPAn offers a collection of FRAs that allow software engineers to capture and document privacy-relevant knowledge of a system-to-be. More precisely, the software artifacts generated during this phase are:

- *(1) Context Diagram*: Describes the domains interacting with the machine, their corresponding interfaces, and exchanged phenomena.
- *(N) Domain Knowledge Diagrams*: Documents facts and assumptions about the context in which the machine will operate.
- *(N) Problem Diagrams*: Represent functional requirements that must be satisfied by the contextual elements of the machine (i.e. domains and phenomena).
- *(1) Detailed Stakeholder Information Flow Diagram (DSIFD)*: Describes how different phenomena flow across the domains of the system.
- *(N) Personal Information Diagrams*: Identifies which data of the stakeholders will be processed by the system and the relations between these data.
- *(N) Available Information Diagrams*: These diagrams consist of two views. The first one identifies the stakeholder data available at the different domains of the system, and the second documents their linkability nature at such domains (e.g. simple or anonymous).

from which only the DSIFD is generated automatically and the rest must be elaborated manually by engineers and privacy experts. To a certain extent, many of these artifacts are introduced to refine the information captured by the problem diagrams into data flows. This is because the phenomena represented in such diagrams describe mainly events and not the information that such events enclose. Hence, this approach 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 "domains" and "phenomena") may alienate those who are not familiar with it and, consequently, hinder the method's usability.

4 PDP-ReqLite: A lightweight method for privacy requirements engineering

All in all, ProPAn counts on multiple artifacts for capturing and documenting privacy-relevant information flows in a system-to-be. Such artifacts provide key input for conducting privacy analyses and, ultimately, for generating privacy requirements. However, their high syntactical correspondence with Jackson's terminology can jeopardize their usability for average software developers. Moreover, the overall documentation overhead in ProPAn can impact the development process of the system-to-be under analysis and eventually delay its commissioning. Hence, we introduce PDP-ReqLite, a lightweight method for

the elicitation of privacy and data protection requirements which incorporates new artifacts for the documentation and analysis of privacy-relevant information flows. Alongside, we describe a protocol for the systematic elicitation of privacy and data protection taxonomies for achieving a broader scope and compliance with legal provisions and privacy standards to come.

4.1 Method Overview

Fig.2 illustrates the proposed requirement elicitation method named PDP-ReqLite. As it can be observed, this method keeps similarities with the second phase of ProPAn. Hence, to keep compatibility, PDP-ReqLite can receive 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 alternative ones containing the same information. Particularly, our new lightweight method introduces two software artifacts as an alternative to the ones from ProPAn. The first one is a Requirements Data-Flow Diagram (RDFD) that describes requirements related to data processing and storage and the information flows between such requirements. Such a diagram resembles many aspects of ProPAn's DSIFD and AID but remains independent from the Problem Frames notation (i.e. it is not specified in terms of *phenomena* or *domains*). The other artifact introduced in this new approach is a Personal Information Diagram (PID). As described in Section 3.2, such a diagram is already present in ProPAn and, like the DSIFD, it its expressed through the jargon of Problem Frames. Conversely, this new PID is expressed in terms of stakeholders, data, and the linkability relations between them. Thereby, we look to overcome the issues related to usability and documentation overhead of the original method.

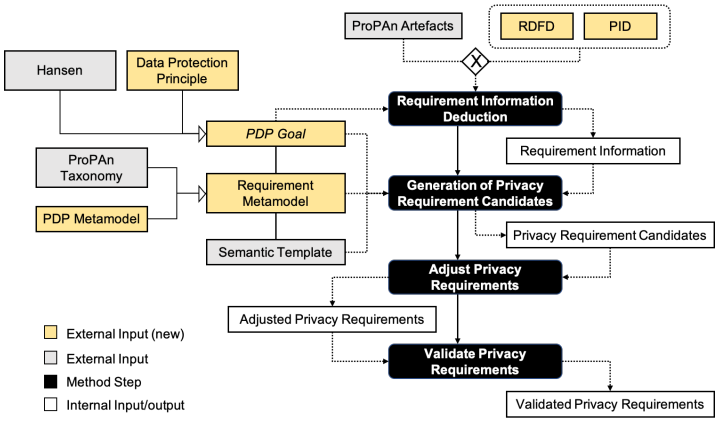


Fig. 2. PDP4E Requirement Elicitation Method

Another element introduced by the PDP-ReqLite corresponds to the meta-models used to generate privacy and data protection requirements. As we de-

scribed in Section 3.1, 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. Nevertheless, expressing GDPR requirements only in terms of a limited set of principles (e.g., intervenability, unlikability and transparency) introduces a risk of methodological bias related to the choice of such principles. For this reason, we introduce two conceptual artifacts 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 4.2). 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 "Privacy and Data Protection Goal" (PDP Goal) in the requirement elicitation method of Fig. 2, which considers the incorporation of new data protection principles complementary to the goals of Hansen. Therefore, we consider the new PDP-ReqLite 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.

Requirements Data-Flow Diagram (RDFD) Fig. 3 (left) describes the meta-model of a Requirements Data-Flow Diagram (RDFD) which was introduced to replace mainly ProPAn's Detailed Stakeholder Information Flow Diagram (DSIFD), but it also contains information originally specified inside an Available Information Diagram (AID). This allow to generate only one global RDFD instead of one DISFD and multiple AIDs for each component inside the DISFD. Following, we detail the most salient components of the RDFD meta-model:

- *RDFD_Element*: Abstract Class. The main elements of the RDFD inherit from this class.
- *Data*: A piece of data is an individual unit of information. All data in a system-to-be has an `origin` which is the RDFD element in which it was originated (i.e. created for the very first time).
- *Record*: A record is a piece of information which has associated a certain `retention_time` in a data storage. Records are composed by data.
- *Data Record Requirement (DRR)*: It represents a requirement related to data storage. Particularly, it represents information structures that are implemented later on in a data base. A DRR is composed by a list of data records (`records_list`) which are pieces of information which have associated a cer-

tain retention time. The attribute `traceability` is a list of the functional requirements that the DRR contributes to.
- *Data Process Requirement (DPR)*: Represents activities that are performed over data records. Similar to a DRR, a DPR is also contains a list of the data (`data_list`) it is required to process. However, unlike DRRs, such data does not have a specific retention time associated. That is, it will be retained in memory during the processing time. In addition, a DPR has a `requestor_list` of the stakeholders that are allowed to execute it.
- *Data Flow Requirement (DFR)*: This element represents the exchange of information between RDFD elements. The attribute `data_list` represents the data flowing from one element to another.

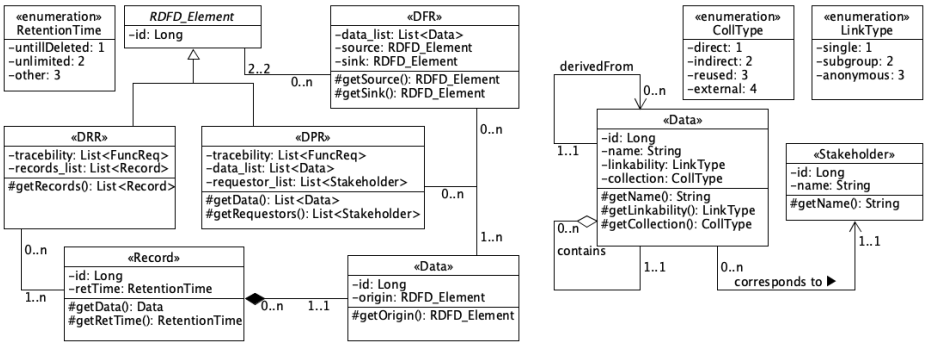


Fig. 3. RDFD Metamodel (left) and PID Metamodel (right).

Personal Information Diagram (PID) As it can be observed, the RDFD merges into one documentation artifact the information which was originally spread across ProPAn's DSIFD and AIDs. This improves the overall documentation navigability and interpretability. Fig.3 (right) introduces the meta-model of a Personal Information Diagram (PID) which is a simplified version of the one from ProPAn. The purpose of this new PID is also the documentation of the stakeholders' personal data that the system must process. The most salient components of the PID meta-model are:

- *Stakeholder*: An individual whose data is processed by the system-to-be.
- *Data*: An individual unit of information that has a `name`, `linkability` and `collection` type. Both, `linkability` and `collection` type have the same semantic as in ProPAn. A piece of data can be `derived_from` or `contain` other pieces of data.

The attribute `linkability` can adopt the values *single*, *subgroup*, or *anonymous*. The value *single* indicates that the data can only identify the individual it

belongs to, *subgroup* when it identifies a potential subgroup of individuals, and *anonymous* when it does not provide any link to the data subject. On the other hand, `collection` can be *direct*, *indirect*, *reused*, or *external*. 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.

4.2 Elicitation of Requirement Candidates

The automated elicitation of requirements is composed by two fundamental blocks: a protocol for covering GDPR directives and an algorithm for requirement candidates generation.

Protocol for covering GDPR directives. The protocol leverages several MDE techniques. First, a meta-model capturing fundamental GDPR notions and privacy principles was developed. The GDPR meta-model defines a language and syntax which is a basis to define so named meta-requirement categories. A meta-requirement is a pattern that results from translation of plain-text GDPR directives into predicates of the form *Pre-conditions* ⇒ *Post-conditions*. *Pre* and *Post-conditions* define the pattern that can be instantiated according to a given system model. As shown in the example bellow, the meta-requirements include placeholders to be filled according to the information included in the RDFD and PID model instance.

IF process <*self.processPD.size()* greater than 0> processes personal Data of <*self.processPD*> **THEN** the Process <*self*> shall be lawful <*self.principles*<*LawFul*> *(self.processPD. DataSubject)* ⇒ *.value=true*>

The protocol followed to create the set of meta-requirements and the respective categories is as follows. First, a GDPR paragraph is taken as input and the goal is to translate it as a meta-requirement relying upon (1) the GDPR meta-model structure and contents and (2) the *Pre* and *Post-conditions* syntax. If the meta-model is not sufficient to do so, then it is extended by integrating the missing GDPR terms, notions and relationships. Once extended and the respective meta-requirement created, a new GDPR paragraph or principle is targeted. This iterative process is repeated till the coverage of GDPR articles and principles is ensured. This protocol ensures the introduction of new privacy principles coming, for instance, from updates or new regulations. Not having any dependency with existing privacy principles, the protocol prevents any bias whereas still achieves a good balance between legal and technical jargon.

Generation of privacy requirements candidates. The generation is done by leveraging the information inside the RDFD and the PIDs. To illustrate how

```
List<UndetectReq> computeUndetectReq(Stakeholder S, Stakeholder C)
{
  List<UndetectReq> reqList = new List<UndetectReq>();
  List<Data> personalData = new List<Data>();

  List<Data> stakeholderData = getStakeholderData(PID,S);
  List<DPR> processList= getProcesses(RDFD,C);

  foreach(p in processList)
  {
    personalData.add(getProcessData(p));
  }

  List<Data> undetectableData = stakeholderData - personalData;

  foreach(u in undetectableData)
  {
    reqList = new UndetectReq(S,C,u);
  }

  return reqList;
}
```

```
The <C> shall not be able to sufficiently distinguish whether the
        personal information <u> of the <S> exist or not.
```

Fig. 4. Snippet for computing undetectability requirements (up) and the corresponding textual template (down)

this can be achieved, we analyse the generation of undetectability requirements. Basically, Pfitzmann et al. [16] define the undetectability of an information item as an attacker's inability to identify if such item exists or not. In other words, if some personal information `PI` of a stakeholder `S` is not accessible to another stakeholder `C` (and `PI` does not contain any other personal information accessible to `C`), then we assume that `PI` is undetectable to `C`. Note that we consider `C` as a stakeholder but also as a potential attacker. That is, as a "counter-stakeholder". Such analysis is described in the snippet of Fig. 4 and starts with the generation of an empty list of requirements (`reqList`) and another of personal data (`personalData`). Additionally, it creates a list of all the personal data of `S` which is kept inside the system-to-be. Such list is computed by the function `getStakeHolderData` with the help of the information inside the PID. Following, a list of all the processes in the RDFD for which `C` appears as requester is computed by the function `getProcesses` and stored inside `processList`. Next, algorithm iterates over the elements of this list, gathers all personal data involved in each process and adds it to `personalData`. Then, it proceeds to compute a list of undetectable data for `C` (`undetectableData`) by subtracting the elements inside `personalData` from `stakeholderData`. Finally, for each element inside `undetectableData` it generates a new `undetectability` requirement using and adds it to `reqList` and returns it afterwards. The text template of Fig. 4 is used for instantiating undetectability requirements. As one can observe, the generation of privacy requirements becomes smooth thanks to the privacy-relevant information captured across PDP-ReqLite's software artifacts.

4.3 Implemented method

The PDP-ReqLite method is implemented as summarized in the following items:

S1. Specify functional requirements. In this phase, the engineer should specify the requirements associated to the functions to be covered by the system (functional requirements). To do so, the engineer can rely on well-known methods as suggested by the International Council of Systems Engineering (INCOSE), e.g., consider [18]. The specification is written in natural language, following good practices like for instance "one single sentence per system function". This step is a mean to simplify the next phase in the elicitation process. In particular, the encapsulation of functional requirements via minimal statements eases their correct transformation into a RDFD model.

S2. Transform functional requirements into a RDFD. In this phase, each functional requirement is transformed into a Requirement Process in the model which includes input/output data and the respective function tasks (i.e., the RDFD). Modelling the involved data types is an activity conducted in parallel. The data are required not only for completing the specification of the Requirement Process elements but also for identifying and labelling Personal Data (i.e., the PID). This phase of the elicitation process should be conducted manually since the expert should examine the form and contents of the functional requirements, and refine/adapt them, prior to transform them into the RDFD.

S3. Validation of the RDFD model and improvement. The validation phase yields outcomes regarding the model completeness and correctness. Model completeness is validated considering the language and syntax defined at meta-model level and determined by mandatory/optional attributes, their multiplicity and the associations types. Model correctness is validated by crosschecking the contents within and between RDFD and PID and the consistency between their attributes. Pre-defined rules are implemented to help the engineer to conduct an assessment validation and infer the solutions in case of conflicts. Indeed, the validation results essentially come as warnings and errors including a description of the issue. Then, the engineer can fix and improve the model so as to make it acceptable for the next phase. Since the process is iterative, going back to previous phases may be necessary.

S4. Privacy and GDPR requirements generation. Once the RDFD and PID model has been validated and accepted, according to the engineer evaluation, the specific privacy and GDPR requirements can be automatically generated: the generation is performed at the push of a button. All the GDPR and privacy principles instantiated either as meta-requirements or as meta-privacy patterns are considered during the generation. The respective algorithms are already implemented as built-in features of the PDP-ReqLite tool. They allow to obtain one requirement per meta-pattern including a full description. The requirements candidates should be validated by the engineer prior to follow other phases of the development cycle.

5 Automated support for PDP-ReqLite application

A tool support has been developed in the scope of the PDP4E project [15]. The PDP-ReqLite tool is developed on top of Papyrus [2] and covers the different modeling and analysis phases of the method as specified in Section 4.3. The tool addresses different aspects regarding the need for automation during requirements generation, usability and re-usability of industry-size models. The main features of the tool are demonstrated in the analysis of a Smart Grid case study.

5.1 Application to a Smart Grid case study

To illustrate the PDP-ReqLite implementation described in Section 4.3, we apply it to analyze a Smart Grid case study. As shown in Figure 5, the Smart Grid design is described by a model composed by seven functions. The GDPR and privacy requirements analysis mainly targets the *Billing* function. The resulting functional path includes the following dependencies. The *Measurement of Consumption* function samples and provides electricity measures which are later gathered by the *Grid Management* including the meter ID and client ID. Overall electricity consumption over time per client are then computed and stored by the *Consumption Data Processing* function. The information are finally accessed by the *Billing* function to generate and submit an invoice relying upon personal data of consumers. Further details of the Smart Grid architecture are provided in this paper [11]. The model in Figure 5 is used to specify the functional requirements

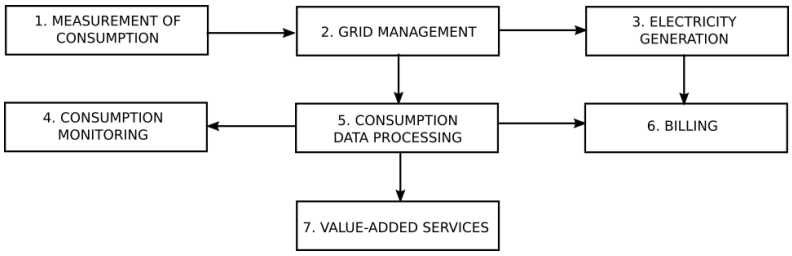


Fig. 5. Reference functions of the Smart Grid System

the system should satisfy (Step 1, Section 4.3). The PDP-ReqLite modeling features support the engineer to map the functional requirements into a RDFD as shown in Figure 6 (Step 2, Section 4.3). The model covers the functional path in the Smart Grid system going from consumption measuring up to generation of the customer bill. The rounded rectangles represent `ProcessRequirements` whereas the normal rectangles represent the `DataStores`. These elements are connected by directed edges representing data flows. The ports attached to the `ProcessRequirements` are typed with conveyed data. Referred types are modelled and structured in a dedicated PID view. The Figure 7 shows an overview

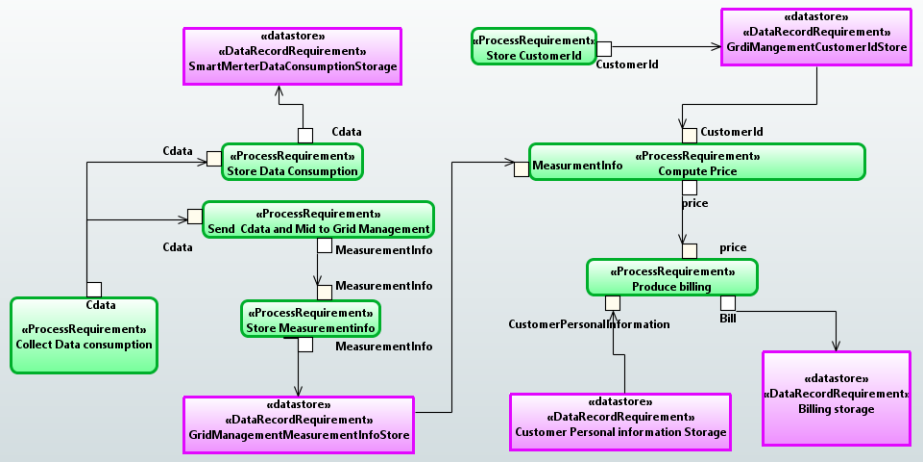


Fig. 6. RDFD model of the Billing process within the Smart Grid system

of the PID used to model the data involved in the Billing processing. The PID includes composite structures that contain information about the electricity measurements (meter ID, consumption), the computed price per customer, and the final bill. More importantly, the PDP-ReqLite tool comes with dedicated types

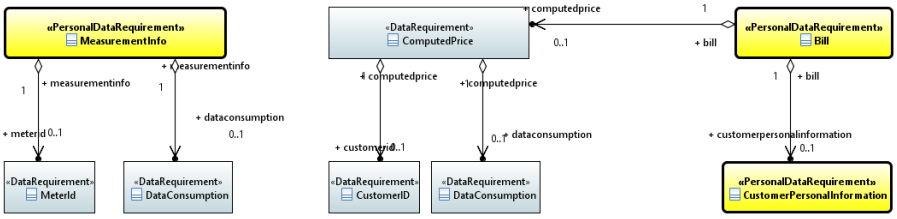


Fig. 7. Model view showing the PID of the Smart Grid System

that can be used to identify personal information what is crucial to correctly elicit GDPR and privacy requirements. Once a first version of the RDFD and PID model was achieved, a round of validation and debugging was conducted (Step 3, Section 4.3). The resulting warnings and errors contain plain text messages that rely upon the syntax and rules defined in the GDPR meta-model. They indicate model discrepancies thus pointing out concerned elements by name and details on the specific issue. So, the completeness and correctness of the model were thus accordingly ensured. The final step in the method application generates the set of GDPR and privacy requirements (Step 4, Section 4.3). The requirements are automatically generated since the GDPR meta-requirement

categories and the generation algorithm introduced in Section 4.2 have been implemented as built-in features of the PDP-ReqLite tool. As shown in Figure

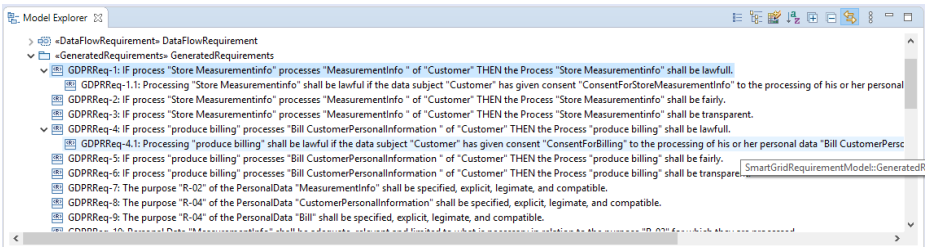


Fig. 8. Overview of the GDPR generated requirements

8, the generated requirements become part of the model and can be reused in other phases of the system development cycle. For instance, during the design phase, requirements are supposed to be satisfied by elements within the design model and validated through tests or similar validation instances. Last yet note the least, since there is no hard link between generated requirements and the RDFD and PID model, an update of the later may demand an iteration in order to obtain an updated version of requirements.

6 Conclusions and perspectives

In this paper we have introduced a lightweight method named PDP-ReqLite to guide and support engineers during the elicitation of GDPR and privacy requirements in systems and software projects. The method has been proposed given that existing approaches impose certain restrictions mainly regarding the coverage of the GDPR regulation and certain biases potentially introduced during requirements elicitation. In this context, we have considered ProPAn as a reference method. By doing so, its main salient features and limitations have been assessed. So far, it has been concluded that ProPAn presents a high documentation overhead, complexity and redundancy across the artifacts it generates. It has been also identified a potential risk of bias since the method relies upon a limited set of principles (e.g., intervenability, unlikability and transparency) what reduces the choices the engineer has during the interpretation of GDPR directives. Overall, certain key ProPAn features, like the usage of privacy taxonomies and a generation algorithm, have been considered to develop extension points in the new method. As novelties of PDP-ReqLite, we can mention the following. The method only requires two modeling artifacts as inputs, the RDFD and PID, what significantly reduces overhead and complexity. The PDP-ReqLite method also introduces and implements a protocol that ensures full coverage of GDPR directives whereas still avoids the need for mapping them

into existing privacy principles. The protocol produces GDPR taxonomies composed by meta-requirements which are predicate patterns defined in terms of GDPR directives and principles. The meta-requirements are instantiated during the requirements elicitation process. These feature also facilitates the integration of updates or even forthcoming privacy regulations. A tool support for the PDP-ReqLite method has been also implemented. The tool leverages several model-driven engineering techniques thus supporting RDFD and PID modelling and also automatic generation of candidate requirements. The method an tool features were finally demonstrated by analyzing a Smart Grid case study.

As work perspectives, we need to improve the automation of GDPR texts processing during the creation of meta-requirements since this task is for now human-based and highly time consuming. Since a high number of requirements can be obtained, even for simple system models, we should consider methods to structure, prioritize and treat them. To consolidate the approach, other privacy regulations can be targeted. Since the PDP-ReqLite method is positioned within a development life cycle, the interfaces with other cycle phases, like design, still need to be developed and consolidated.

Acknowledgments. This work has been conducted in the scope of the project PDP4E - Methods and tools for GDPR compliance through Privacy and Data Protection Engineering. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 787034.

References

1. Beckers, K., Faßbender, S., Heisel, M., Meis, R.: A problem-based approach for computer-aided privacy threat identification. In: Annual Privacy Forum. pp. 1–16. Springer (2012)
2. CEA-LIST: The Eclipse Papyrus project. https://www.eclipse.org/papyrus/ (2020)
3. Colesky, M., Hoepman, J.H., Hillen, C.: A critical analysis of privacy design strategies. In: 2016 IEEE Security and Privacy Workshops (SPW). pp. 33–40. IEEE (2016)
4. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requirements Engineering **16**(1), 3–32 (2011)
5. Dennedy, M., Fox, J., Finneran, T.: The privacy engineer's manifesto: getting from policy to code to QA to value. Apress (2014)
6. Hansen, M., Jensen, M., Rost, M.: Protection goals for privacy engineering. In: 2015 IEEE Security and Privacy Workshops. pp. 159–166. IEEE (2015)
7. Hoepman, J.H.: Privacy design strategies. In: IFIP International Information Security Conference. pp. 446–459. Springer (2014)
8. Howard, M., Lipner, S.: The security development lifecycle, vol. 8. Redmond: Microsoft Press. Google Scholar Google Scholar Digital Library Digital Library (2006)

9. Information Technology-Security Techniques-Privacy Framework. Standard ISO/IEC 29100:2011, International Organization for Standardization, Geneva, CH (December 2011), https://www.iso.org/standard/45123.html
10. Jackson, M.: Problem frames: analysing and structuring software development problems. Addison-Wesley (2001)
11. Laakkonen, J., Annala, S., Jäppinen, P.: Abstracted architecture for smart grid privacy analysis. In: 2013 International Conference on Social Computing. pp. 637–646 (2013)
12. Meis, R.: Problem-Based Privacy Analysis (ProPAn): A Computer-aided Privacy Requirements Engineering Method. Ph.D. thesis, DuEPublico: Duisburg-Essen Publications [online], University of Duisburg-Essen, Germany (October 2018). https://doi.org/10.17185/duepublico/47797
13. Meis, R., Heisel, M.: Computer-aided identification and validation of privacy requirements. Information **7**(2), 28 (2016)
14. Notario, N., Crespo, A., Martín, Y.S., Del Alamo, J.M., Le Métayer, D., Antignac, T., Kung, A., Kroener, I., Wright, D.: Pripare: integrating privacy best practices into a privacy engineering methodology. In: 2015 IEEE Security and Privacy Workshops. pp. 151–158. IEEE (2015)
15. PDP4E: The PDP4E Project. https://www.pdp4e-project.eu/ (2020)
16. Pfitzmann, A., Hansen, M.: A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management (Aug 2010), http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf, v0.34
17. Regulation, G.D.P.: 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) **59**(1-88), 294 (2016)
18. Ryan, M., Wheatcraft, L., Zinni, R., Dick, J., Baksa, K.: Guide for writing requirements. INCOSE, California, USA (2017)