A Foundation for Requirements Analysis of Privacy Preserving Software

Kristian Beckers, Maritta Heisel

A Foundation for Requirements Analysis of Privacy Preserving Software*

Kristian Beckers, Maritta Heisel

paluno - The Ruhr Institute for Software Technology – University of Duisburg-Essen
{firstname.lastname}@paluno.uni-due.de

Abstract. Privacy requirements are difficult to elicit for any given software engineering project that processes personal information. The problem is that these systems require personal data in order to achieve their functional requirements and privacy mechanisms that constrain the processing of personal information in such a way that the requirement still states a useful functionality.
We present privacy patterns that support the expression and analysis of different privacy goals: anonymity, pseudonymity, unlinkability and unobservability. These patterns have a textual representation that can be instantiated. In addition, for each pattern, a logical predicate exists that can be used to validate the instantiation. We also present a structured method for instantiating and validating the privacy patterns, and for choosing privacy mechanisms. Our patterns can also be used to identify incomplete privacy requirements. The approach is illustrated by the case study of a patient monitoring system.

Keywords: privacy, common criteria, compliance, requirements engineering

1 Introduction

Westin defines privacy as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [1]. A number of guidelines for privacy are available. *The Fair Information Practice Principles* – or short FIPs) [2] – are widely accepted, which state that a person's informed consent is required for the data that is collected, collection should be limited for the task it is required for and erased as soon as this is not the case anymore. The collector of the data shall keep the data secure and shall be held accountable for any violation of these principles. The FIPs were also adapted into the Personal Information Protection and Electronic Documents Act in Canada's private-sector privacy law. In the European Union the *EU Data Protection Directive*, *Directive 95/46/EC*, does not permit processing personal data at all, except when a specific legal basis explicitly allows it or when the individuals concerned consented prior to the data processing [3]. The U.S. have no central data protection law, but separate privacy laws, e.g., the Gramm-Leach-Bliley Act for financial information, the Health Insurance Portability and Accountability Act for medical information, and the Children's Online Privacy Protection Act for

* This research was partially supported by the EU project Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS, ICT-2009.1.4 Trustworthy ICT, Grant No. 256980).

data related to children [4]. These legal guidelines must be implemented by any given software system for which the guidelines apply.

However, in order to comply with these guidelines the privacy requirements for a given software system have to be elicited. In order to do this we have to formulate specific privacy goals. We use two distinct approaches that specify privacy terms that can used for this purpose, namely the terminology by Pfitzmann and Hansen [5] and the privacy specification in the ISO 15408 standard - Common Criteria for Information Technology Security Evaluation (or short CC) [6]. Pfitzmann and Hansen [5] introduced a terminology for privacy via data minimization. They define central terms of privacy using items of interest (IOIs) , e.g., subjects, messages and actions. *Anonymity* means that a subject is not identifiable within a set of subjects, the anonymity set. *Unlinkability* of two or more IOIs means that within a system the attacker cannot sufficiently distinguish whether these IOIs are related or not. *Undetectability* of an IOI means that the attacker cannot sufficiently distinguish whether it exists or not. *Unobservability* of an IOI means undetectability of the IOI against all subjects uninvolved in it and anonymity of the subject(s) involved in the IOI even against the other subject(s) involved in that IOI. A pseudonym is an identifier of a subject other than one of the subject's real names. Using pseudonyms means *pseudonymity*. The CC contains the privacy requirements anonymity, pseudonymity, unlinkability, and unobservability [6, pp. 118-125].

In this paper, we provide patterns for these privacy requirements, building on the problem frame terminology, and we explain the difference between the privacy notions in the CC and the Pfitzmann and Hansen terminology.

The rest of the paper is organized as follows. Section 2 presents the problem frame approach, and Sect. 3 explains how to work with privacy patterns. We introduce our patterns in Sect. 4 and illustrate them using a case study. Section 5 contains related work, and Sect. 6 concludes.

2 Problem Frames

We use a problem frames approach to build our privacy patterns on, because problem frames are an appropriate means to analyze not only functional, but also dependability and other quality requirements [7, 8].

Problem frames are a means to describe software development problems. They were proposed by Jackson [9], who describes them as follows: *"A problem frame is a kind of pattern. It defines an intuitively identifiable problem class in terms of its context and the characteristics of its domains, interfaces and requirement."*. It is described by a *frame diagram*, which consists of domains, interfaces between them, and a requirement. We describe problem frames using class diagrams extended by stereotypes as proposed by Hatebur and Heisel at Safecomp 2010 [10](see Fig. 1). All elements of a problem frame diagram act as placeholders, which must be instantiated to represent concrete problems. Doing so, one obtains a problem description that belongs to a specific kind of problem.

The class with the stereotype machine represents the thing to be developed (e.g., the software). The other classes with some domain stereotypes, e.g., *CausalDomain* or *BiddableDomain* represent *problem domains* that already exist in the application environment. Domains are connected by interfaces consisting of shared phenomena.

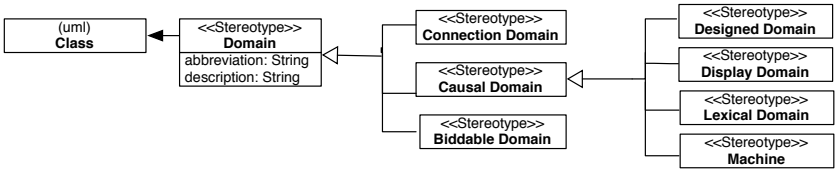

Fig. 1. Inheritance Structure of different Domain Types

Shared phenomena may be events, operation calls, messages, and the like. They are observable by at least two domains, but controlled by only one domain, as indicated by an exclamation mark. These interfaces are represented as associations, and the name of the associations contain the phenomena and the domains controlling the phenomena.

Jackson distinguishes the domain types ***CausalDomain***s that comply with some physical laws, ***LexicalDomain***s that are data representations, and ***BiddableDomain***s that are usually people. According to Jackson, domains are either ***designed***, ***given***, or ***machine*** domains. The domain types are modeled by the subclasses *BiddableDomain*, *CausalDomain*, and *LexicalDomain* of the class *Domain*. A lexical domain is a special case of a causal domain. This kind of modeling allows one to add further domain types, such as *DisplayDomain*s as introduced in [11] (see Fig. 1).

Problem frames support developers in analyzing problems to be solved. They show what domains have to be considered, and what knowledge must be described and reasoned about when analyzing the problem in depth.

Software development with problem frames proceeds as follows: first, the environment in which the machine will operate is represented by a *context diagram*. Like a frame diagram, a context diagram consists of domains and interfaces. However, a context diagram contains no requirements. Then, the problem is decomposed into subproblems. If ever possible, the decomposition is done in such a way that the subproblems fit to given problem frames. To fit a subproblem to a problem frame, one must instantiate its frame diagram, i.e., provide instances for its domains, phenomena, and interfaces. The instantiated frame diagram is called a *problem diagram*.

Since the requirements refer to the *environment* in which the machine must operate, the next step consists in deriving a *specification* for the machine (see [12] for details). The specification describes the machine and is the starting point for its construction.

3 Working with Privacy Patterns

We developed a set of patterns for expressing and analyzing privacy requirements, which are presented in more detail in Sect. 4. An important advantage of these patterns is that they allow privacy requirements to be expressed without anticipating solutions. For example, we may require data to be anonymized before these are transmitted without being obliged to mention k-Anonymity [13], which is a means to achieve anonymity.

The benefit of considering privacy requirements without reference to potential solutions is the clear separation of problems from their solutions, which leads to a better

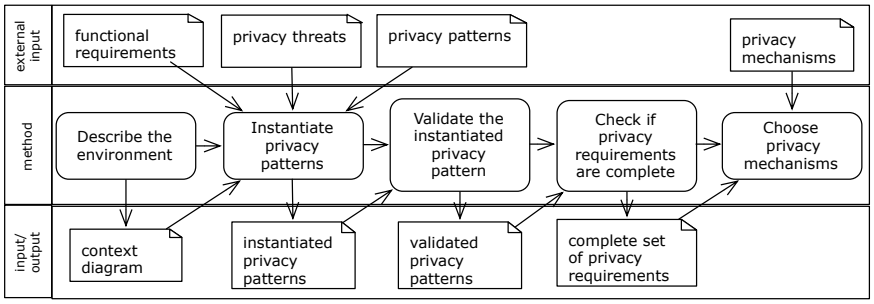

Fig. 2. A Method for Using Privacy Patterns

understanding of the problems and enhances the re-usability of the problem descriptions, since they are completely independent of solution technologies.

We provide a method for privacy requirements elicitation using our privacy patterns. The entire method is depict in Fig. 2. This approach helps to elicit and complete privacy requirements within a given software engineering process. We use the resulting set of requirements to choose privacy-enhancing mechanisms.

The first step in our method is to *describe the environment*, because privacy requirements can only be guaranteed for a specific intended environment. For example, a software may preserve privacy of its users in a social media setting, where it is only exchanging information about concert tickets, but not for a medical setting exchanging more sensible information about the health status of its stakeholders. This step results in a context diagram (see Sect. 2) of the intended software system.

We apply our patterns after a privacy threat analysis has been conducted, e.g., according to the PIA method of the Australian Government [14], and functional requirements of the system have been elicited. The requirements describe how the environment should behave when the machine is in action. The description of the requirements must consist of domains and phenomena of the environment description. The functional requirements and privacy threats, as well as our privacy patterns are used to *instantiate privacy patterns*. For each privacy threat, a textual representation of a privacy pattern is instantiated using the context diagram. This results in an initial set of privacy requirements. These are linked to the previously described functional requirements.

The next step is to *validate the instantiated privacy patterns* using the corresponding privacy predicates. Our privacy requirements patterns are expressed in natural language and logical predicates. The natural text has parts written in **bold**, which have to be instantiated with ***domains*** of a context diagram for the system. These present a structured analysis of the required elements in order to specify the requirement. In addition, the logical predicates refer to **domain types** in a context diagram that have a kind-of relationship to the domains in the natural language pattern. This allows us to check if the instantiated domains have the correct domain types.

Privacy requirements are separated from functional requirements. On the one hand, this limits the number of patterns; on the other hand, it allows one to apply these pat-

terns to a wide range of problems. For example, the functional requirements for data transmission or automated control can be expressed using a problem diagram. Privacy requirements for anonymity, unlinkability, unobservability, and pseudonymity can be added to that description of the functional requirement, as shown in Sect.4.

The predicate patterns are expressed using the domain types of the meta-model described in Figure 1, i.e., ***Domain***, ***BiddableDomain***, ***CausalDomain***, and ***LexicalDomain***. From these classes in the meta-model, subclasses with special properties are derived. We explain the domain types that are relevant for this step in the method:

- A ***Stakeholder*** is a ***BiddableDomain*** (and in some special cases also a ***CausalDomain***) with some relation to stored or transmitted personal information. It is not necessary that a stakeholder has an interface to the machine.
- A ***CounterStakeholder*** is a ***BiddableDomain*** that describes all subjects (with their equipment) who can compromise the privacy of a ***Stakeholder*** at the machine. We do not use the term attacker here, because the word attacker indicates a malicious intent. Privacy of stakeholders can also be violated by accident.
- ***PersonalInformation*** is a ***CausalDomain*** or ***LexicalDomain*** that represents personal information about a ***Stakeholder***. The difference between these domains is that a ***LexicalDomain*** describes just the stored information, while a ***CausalDomain*** also includes the physical medium the data is stored upon, e.g., a hard drive.
- ***StoredPersonalInformation*** is ***PersonalInformation***, which is stored in a fixed physical location, e.g., a hard drive in the U.S.
- ***TransmittedPersonalInformation*** is ***PersonalInformation***, which is transmitted in-between physical locations, e.g., data in a network that spans from Germany to the U.S.
- ***InformationAboutPersonalInformation*** is is a ***CausalDomain*** or ***LexicalDomain*** that represents information about ***PersonalInformation***, e.g., the physical location of the name and address of a stakeholder.

The *check if privacy requirements are complete* is a check that all the textual gaps are instantiated. For example, a requirement that has to be instantiated with a ***Stakeholder*** has to name an instance of a biddable domain from the context diagram, e.g., ***Patient***s. Several privacy patterns require instantiation with sets of biddable domains. For example, a privacy pattern might not only be directed towards ***Patient***s, but also towards ***Visitor***s of patients. In this case we can reason for all biddable domains in the context diagram if they are a ***Stakeholder*** or not.

In addition, a requirement might be missing, e.g. because of an incomplete threat analysis. In order to execute this check all personal information in the system has to be elicited. For each ***CausalDomain*** or ***LexicalDomain***, we have to check if these are ***StoredPersonalInformation***, ***TransmittedPersonalInformation*** or ***InformationAboutPersonalInformation***. If this is the case, we check if these were considered in the privacy threat analysis. If this is not the case, we have to re-do the privacy threat analysis and start our process again.

The last step of our approach is to *choose a privacy mechanism* that solves the problem. For example, to achieve pseudonymity, a privacy-enhancing identity management systems [15] can be chosen.

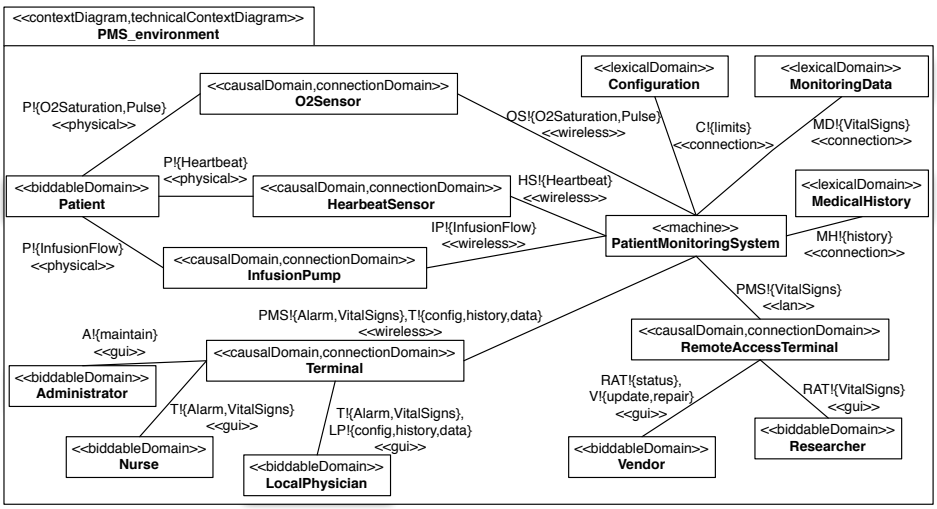

Fig. 3. Patient Monitoring System Context Diagramm

4 Privacy Patterns

We illustrate our method with the development of a *patient monitoring system (PMS)*, which monitors the vital signs of patients, reports these to physicians and nurses, and controls an infusion flow according to specified rules. The context diagram of the system is shown in Fig. 3. The configuration is stored in a database of the system, as well as the monitoring data of each patient over time and the medical history of each patient. This history shall help physicians to determine the correct settings for the PMS. Nurses and physician have access to alarm messages of the system and the information about the vital signs of patients. An administrator has access to the system for maintenance purposes. The system also allows a remote access for the vendor in order to receive status messages and send updates or repair the system. In addition, the system provides information of monitored data to researchers. The **privacy threat** to be avoided is that a counterstakeholder reveals personal information of the patient to one or more unauthorized persons.[1] Examples for the **described environment** are the properties of the infusion pump and the heartbeat and O2 flow sensors. Further examples are the assumed opportunities of a counterstakeholder to gather personal information about patients and to distribute it. A possible counterstakeholder can be a stakeholder of the system. In addition, we assume here that a counterstakeholder, who is not also a stakeholder of the system, can only access the WAVE/WLAN interface. Stakeholders and counterstakeholders are biddable domains in the context diagram shown in Fig. 3. The elicitation of stakeholders and counterstakeholders from biddable domains is part of the instantiation process. Hence, the reasoning of which biddable domain is either a stakeholder or counterstakeholder or both is essential. The information from the privacy threat serves as a

[1] The privacy threat analysis is left out here.

support in this task. The **functional requirement** of the PMS is to keep the infusion flow controlled according to the configuration.

R1 The PMS shall control the infusion flow of a ***Patient*** using the ***InfusionPump*** according to the ***Configuration*** given by a ***Physician*** based upon the vital signs of the ***Patient***, which are transmitted using a wireless network (WAVE/WLAN interface).
R2 The PMS should raise an alarm for ***Physician***s and ***Nurse***s using the ***Terminal***, if the vital signs of a ***Patient*** exceed the limits defined in the ***Configuration***.
R3 ***Physician***s can change the ***Configuration***, according to the ***MonitoringData*** of the vital signs and the ***MedicalHistory*** of a ***Patient*** in the PMS using the ***Terminal***, which sends the ***Configuration*** to the PMS using a wireless network (WAVE/WLAN interface).
R4 ***Researcher***s can use the collected data about ***Patient***s' vital signs for a long term medical study using the ***RemoteAccessTerminal*** that sends the data over a wired network (LAN interface) from the PMS. The ***Vendor*** can query the status of the PMS and send patches to the PMS using the ***RemoteAccessTerminal***.
R5 ***Administrator***s can use the ***Terminal*** to maintain the PMS.

We **instantiate privacy requirements** as a next step. We describe privacy patterns as textual patterns. The parts of the pattern's textual description printed in ***bold and italics*** should be instantiated according to the concrete problem. Hence, the instantiation should use only domains from the context diagram. We complement the textual description of the patterns with predicates. They must be described in such a way that it is possible to demonstrate that the privacy predicate holds for all objects of this class. The instantiated predicates are helpful to analyze conflicting requirements and the interaction of different privacy requirements, as well as for finding missing privacy requirements.

4.1 Anonymity

For the functional requirement **R4**, we formulate a privacy requirement for anonymity. The textual pattern for anonymity requirements is:

> Preserve anonymity of ***Stakeholder***s and prevent disclosure of their identity by ***CounterStakeholder***s.

The privacy requirement pattern can be expressed by the anonymity predicate:

$$anon_{cs} : BiddableDomain \times \mathbb{P}\, BiddableDomain \rightarrow Bool$$

The suffix "cs" indicates that this predicate describes a requirement considering a certain *C*ounter*S*takeholder. The definition of anonymity by Pfitzmann and Hansen [5] states that a stakeholder shall not be identifiable from a set of stakeholders. This is the so-called *anonymity set*, which is represented in our pattern by a biddable domain and in turn a specific type of biddable domain called ***Stakeholder***. We interpret a domain according to Jackson [9] also as a set. Hence, all the persons that are ***Stakeholder***s used to instantiate the anonymity pattern form the anonymity set. The second argument of

the predicate $anon_{cs}$ must be instantiated with a *set* of biddable domains, because we might have more than one kind of counterstakeholder. We analyze the context diagram (see Fig. 3) for counterstakeholders that might gain personal information using the ***RemoteAccessTerminal***. This leads to the ***Researcher***.

> Preserve anonymity of ***Patient***s and prevent disclosure of their identity by ***Researcher***s.

In this case all instances of ***Patient***s are elements of the anonymity set.
We **validate the instantiated privacy pattern** by checking if predicate is instantiated with domains belonging to the required domain types: The ***Patient*** is a biddable domain and the ***Researcher*** is also a biddable domain, which forms the only member of the set of counterstakeholder. Hence, the anonymity predicate is instatiated in a type-correct way. As a next step we **check if the privacy requirement is complete**.

The counterstakeholders are a set of biddable domains. This demands that we inspect the context diagram (see Fig. 3) again for further possible counterstakeholders. Another biddable domain has access to the ***RemoteAccessTerminal***, the ***Vendor***. Therefore, we add the ***Vendor*** to the list of counterstakeholders. We **choose a privacy mechanism** as a final step for anoymity, e.g., based upon the work in [16].

$$anon_{cs} : Patients \times \{Researcher, Vendor\} \rightarrow Bool$$

Common Criteria The common criteria divides requirements for anonymity into two categories. The first one is just called anonymity and the second one is a refinement of the first and called anonymity without soliciting information. The first demands from a privacy mechanism that it shall ensure that a set of "users" or "subjects" are unable to determine the "real name" related to a set of "subjects or operations or objects" [6, p. 119]. This demand is similar to the requirement above. The set of users is translated into ***CounterStakeholder***s and subjects are ***Stakeholder***s. The second kind of anonymity the CC considers is the so-called "anonymity without soliciting information" about the real user name for a set of services and subjects [6, p. 119]. This requirement needs the first requirements as a prerequisite.

We formulate an anonymity without soliciting personal information requirement :

> Preserve anonymity of ***Stakeholder***s via not soliciting ***personal information*** via ***ConstrainedDomain***s and, thus, prevent disclosure to certain ***CounterStakeholder***s.

where a ***ConstrainedDomain*** is a ***CausalDomain*** or a ***ConnectionDomain*** that is constrained by a functional or privacy requirement.
The privacy requirement pattern can be expressed by the anonymity without soliciting personal information predicate:

$$anonCC_{cs} : BiddableDomain \times \mathbb{P}\, LexicalDomain \times \mathbb{P}\, ConstrainedDomain \rightarrow Bool$$

The first kind anonymity requirement of the CC is instantiated with the requirement stated above. We instantiate also the textual pattern for the second kind of requirement using the context diagram. We instantiate the ***ConstrainedDomain*** with the ***RemoteAccessTerminal*** and the ***personal information*** with ***MedicalHistory***:

> Preserve anonymity of ***Patient***s via not soliciting ***MedicalHistory*** via ***RemoteAccessTerminal*** and, thus, prevent disclosure to ***Researcher***s and ***Vendor***s.

4.2 Unlinkability

The textual pattern for unlinkability requirements is:

> Preserve unlinkability of two or more ***ConstrainedDomain***s for ***Stakeholder***s and prevent ***CounterStakeholder***s of disclosing that the ***ConstrainedDomain***s have a relation to the ***Stakeholder***.

The privacy requirement pattern can be expressed by the unlinkability predicate:

$$unlink_{cs} : \mathbb{P}\, CausalDomain \times BiddableDomain \times \mathbb{P}\, BiddableDomain \rightarrow Bool$$

We also **instantiate privacy requirements** for the functional requirement **R3**. The following privacy requirement for unlinkability can be stated using the textual pattern:

> Preserve unlinkability of ***Configuration***s, ***MonitoringData***, and ***MedicalHistory*** for a ***Patient*** and prevent ***Nurse***s and ***Administrator***s of disclosing that the ***Configuration***s, ***MonitoringData***, and ***MedicalHistory*** have a relation to the ***Patient***.

We again **validate the instantiated privacy pattern** by checking if predicate is instantiated with domains belonging to the required domain types: The domains ***Configuration***s, ***MonitoringData***, and ***MedicalHistory*** are lexical domains according to the context diagram (see Fig. 3) and these are refined causal domains (see Fig. 1). The ***Patient*** is a biddable domain and ***Nurse***s and ***Administrator***s are also biddable domains. Hence, the unlinkability predicate is instatiated in a type-correct way.

We **check if the privacy requirement is complete**. The ***LocalPhysician*** also has access ***Terminal***, but **R3** states that the ***LocalPhysician*** requires access to these data. Thus, ***LocalPhysician***s are excluded from the unlinkability requirement. We **choose a privacy mechanism** for unlinkability, e.g., based upon the work in [17].

Common Criteria The common criteria lists requirements for unlinkability of just one category. This demands from a privacy mechanism that it shall ensure that a set of "users" or "subjects" are unable to determine if "operations" are used by the same users or have other recurring relations [6, p. 122].

4.3 Unobservability

The textual pattern for unobservability requirements is:

> Preserve unobservability of ***ConstrainedDomain***s that are used by ***Stakeholder***s and prevent ***CounterStakeholder***s from recognizing that the ***ConstrainedDomain***s exist.

The privacy requirement pattern can be expressed by the unobservability predicate:

$$unobserv_{cs} : \mathbb{P}\, CausalDomain \times BiddableDomain \times \mathbb{P}\, BiddableDomain \rightarrow Bool$$

An example of an instantiated unobservability requirement is:

> Preserve unobservability of a ***MedicalHistory*** that is used by ***LocalPhysician***s and prevent ***Administrator(s),Nurse(s)*** from recognizing that the ***MedicalHistory*** exists.

Common Criteria The common criteria divides requirements for unobservability into four categories. The first demands from a privacy mechanism that it shall ensure that a set of "users" or "subjects" are unable to observe certain "operations" on "objects" [6, p. 123-125]. This first requirements is equivalent to the requirement stated previously.

The second kind of unobservability the CC considers is the so-called "allocation of information impact unobservability". The CC demands a privacy mechanism that ensures that "unobservability related information" is distributed to different parts of the machine, such that specific conditions hold, which ensure allocation of information impact unobservability [6, p. 123-125]. The standard does not specify these conditions. This kind of unobservability needs the first kind of unobservability requirement as a prerequisite.

We formulate an allocation of information impact unobservability requirement:

> Distribute ***InformationAboutPersonalInformation*** of ***Stakeholder***s on the machine, such that a ***CounterStakeholder*** cannot recognize its existence.

Specific conditions have to be derived in order to be able to check whether the requirement is fulfilled. The privacy requirement pattern can be expressed by the unobservability related information predicate:

$$unobservRelInf_{cs} : CausalDomain \times BiddableDomain \times \mathbb{P}\, BiddableDomain \rightarrow Bool$$

The third kind of unobservability the CC considers is the so-called "unobservability without soliciting information". This demands unobservability without soliciting information about personal information [6, p. 123-125].
We formulate an unobservability without soliciting personal information requirement:

> Preserve unobservability without soliciting personal information of ***Stakeholder***s via not soliciting ***PersonalInformation*** and, thus, prevent disclosure to a certain ***CounterStakeholder***.

The privacy requirement pattern can be expressed by the unobservability without soliciting personal information predicate:

$$unobservWoSol_{cs} : BiddableDomain \times LexicalDomain \times \mathbb{P}\, BiddableDomain \rightarrow Bool$$

The fourth kind of unobservability the CC considers is the so-called "authorized user observability". The standard demands a solution that offers a list of "authorized users" that can observe the usage of "resources and services" [6, p. 123-125].

We formulate an authorized user observability unobservability requirement :

> Provide access of authorized ***Stakeholder***s to ***InformationAboutPersonalInformation***.

We also **instantiate privacy requirements** for the functional requirement **R3**. The privacy requirement pattern can be expressed by the unobservability related information predicate:

$$unobservRelInf_{cs} : \mathbb{P}\, BiddableDomain \times LexicalDomain \rightarrow Bool$$

4.4 Pseudonymity

A ***Pseudonym*** is a ***LexicalDomain*** used as an identifier of a ***Stakeholder*** without revealing ***PersonalInformation***. An ***Authorized User*** is a ***Stakeholder*** who is allowed to know the identity of the ***Stakeholder*** the ***Pseudonym*** belongs to.

The textual pattern for pseudonymity requirements is:

> Preserve pseudonymity of ***Stakeholder***s via preventing ***CounterStakeholder***s from relating ***Pseudonym***s to ***Stakeholder***s.

The privacy requirement pattern can be expressed by the pseudonymity predicate:

$$pseudo_{cs} : LexicalDomain \times BiddableDomain \times \mathbb{P}\, BiddableDomain \rightarrow Bool$$

For the functional requirement **R5**, we formulate a privacy requirement for pseudonymity:

> Preserve pseudonymity of ***Patient***s via preventing ***Administrator***s from relating ***Pseudonym***s to ***Patient***s.

Common Criteria The common criteria divides requirements for pseudonymity into three categories [6, p. 120-121]. The first demands from a privacy mechanism that it shall ensure that a set of "users" or "subjects" are unable to determine the real user name of "subjects or operations or objects" or "operations" or "objects". In addition, the privacy mechanism shall use "aliases" of the real user name for "subjects". Moreover, the privacy mechanism shall decide which "alias" the user gets assigned. The "alias" has to conform to an "alias metric". The following kinds of pseudonymity requirements have the first kind of pseudonymity requirements as a prerequisite.
The second kind of pseudonymity the CC considers is the so-called "reversible pseudonymity". This demands that authorized users can determine the user identity under a list of conditions [6, p. 120-121]. The standard does not specify these conditions further.

We formulate an reversible pseudonymity requirement:

> Preserve pseudonymity of ***Stakeholder***s via preventing that a certain ***CounterStakeholder*** from relating a ***Pseudonym*** to its ***Stakeholder***. An ***Authorized User*** shall be able to relate a ***Pseudonym*** to its ***Stakeholder***.

The privacy requirement pattern can be expressed by the reversible pseudonymity predicate:

$$pseudoRe_{cs} : LexicalDomain \times BiddableDomain \times \mathbb{P}\, BiddableDomain \times BiddableDomain \rightarrow Bool$$

The third kind of pseudonymity the CC considers is the so-called "alias pseudonymity". This demands if a ***Stakeholder*** gets a ***Pseudonym*** assigned it shall either be always the same or the two ***Pseudonym***s shall not be related at all [6, p. 120-121].

We formulate an alias pseudonymity requirement:

> Provide the same ***Stakeholder*** with the same or completely unrelated ***Pseudonym***s.

The privacy requirement pattern can be expressed by the alias pseudonymity predicate:

$$pseudoAl_{cs} : LexicalDomain \times BiddableDomain \rightarrow Bool$$

5 Related Work

The authors Deng et al. [18] generate a threat tree for privacy based upon the threat categories: linkability, identifiablitiy, non-repudiation, detectability, information disclosure, content unawareness, and policy/consent noncompliance. These threats are modeled for the elements of an information flow model, which has data flow, data store, processes and entities as components. Privacy threats are described for each of these components. This method can complement our own. The results of the privacy threat analysis from Deng et al. can be used as an input for our method.

The PriS method elicits privacy requirements in the software design phase. Privacy requirements are modeled as organizational goals. Further privacy process patterns are used to identify system architectures, which support the privacy requirements

[19]. The PriS method starts with a conceptual model, which also considers enterprise goals, stakeholders, privacy goals, and processes [19]. In addition, the Pris method is based upon a goal oriented requirements engineering approach, while our work uses a problem based approach as a foundation. The difference is that our work focuses on a description of the environment as a foundation for the privacy analysis, while the Pris method uses organizational goals as a starting point.

Hafiz described four privacy design patterns for the network level of software systems. These patterns solely focus on anonymity and unlinkability of senders and receivers of network messages from protocols e.g. http [20]. The patterns are specified with several categories. Among them are intent, motivation, context, problem and solution, as well as forces, design issues and consequences. Forces are relevant factors for the applicability of the pattern e.g. number of users or performance. Design issues describe how the forces have to be considered during software design. For example, the number of stakeholders have to have a relevant size for the pattern to work. Consequences are the benefits and liabilities the pattern provides. For example, an anonymity pattern can disguise the origin of a message, but the pattern will cause a significant performance decrease [20]. This work focuses on privacy issues on the network layer and can complement our work in this area.

Hatebur and Heisel proposed similar patterns for expressing and analyzing dependability requirements [7].

6 Conclusions

In this paper, we have presented a set of patterns for eliciting and analyzing privacy requirements. These patterns are separated from the functional requirements and expressed without anticipating solutions. They can be used to create re-usable privacy requirements descriptions for a wide range of problems.

Our work also includes a structured method for instantiating privacy patterns and validates the correctness and completeness of the instantiated patterns in a systematic way. The patterns are based upon natural language privacy goals: anonymity, unlinkability, unobservability, and pseudonymity. The instantiated parameters of the patterns refer to domains of the environment descriptions and are used to describe the privacy requirements precisely. Predicates exist for each of the patterns, which can be used to validate that the instantiated parameters from the environment description have the correct domain types. In addition, several values of the privacy patterns can be instantiated with a set of subclasses of a specific domain type. Hence, we can reason about all domains of that type in the environment description, if they are part of this set or not.
In summary, our pattern system has the following advantages:

- The privacy patterns are re-usable for different projects.
- A manageable number of patterns can be applied on a wide range of problems, because they are separated from the functional requirements.
- Requirements expressed by instantiated patterns only refer to the environment description and are independent from solutions. Hence, they can be easily re-used for new product versions.

- The patterns closely relate textual descriptions and predicates. The textual description helps to instantiate the privacy requirements, while the predicates are used for validation of the instantiation.
- The patterns help to structure and classify the privacy requirements. For example, requirements considering anonymity can be easily distinguished from unlinkability requirements. It is also possible to trace all privacy requirements that refer to one domain.
- The patterns also have variations to satisfy the privacy requirements stated in the common criteria.

In the future, we plan to elaborate more on the later phases of software development. For example, we want to apply our patterns to software components to show that a certain architecture enforces privacy for its intended usage. Additionally, we plan to systematically search for privacy requirements using existing specifications (e.g., public privacy statements). Moreover, we want to evaluate a chosen privacy mechanism against the capabilities of the known counterstakeholders in order to evaluate its usefulness.

Acknowledgements

We thank Stephan Faßbender and Denis Hatebur for their extensive and valuable feedback on our work.

References

1. Westin, A.F.: Privacy and Freedom. Atheneum, New York (1967)
2. OECD: OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. Technical report, Organisation for Economic Co-operation and Development (OECD) (1980)
3. EU: Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Technical report, European Community(EU) (1995)
4. Hansen, M., Schwartz, A., Cooper, A.: Privacy and Identity Management. Security & Privacy, IEEE **6**(2) (2008) 38–45
5. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization: Anonymity, unlinkability, unobservability, pseudonymity, and identity management - version v0.34. Technical report, TU Dresden and ULD Kiel (2011)
6. ISO and IEC: Common Criteria for Information Technology Security Evaluation – Part 2 Security functional components. ISO/IEC 15408, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) (2009)
7. Hatebur, D., Heisel, M.: A foundation for requirements analysis of dependable software. In Buth, B., Rabe, G., Seyfarth, T., eds.: Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP). Volume 5775 of LNCS., Springer Berlin / Heidelberg / New York (2009) 311–325
8. Alebrahim, A., Hatebur, D., Heisel, M.: A method to derive software architectures from quality requirements. In Thu, T.D., Leung, K., eds.: Proceedings of the 18th Asia-Pacific Software Engineering Conference (APSEC), IEEE Computer Society (2011) 322–330
9. Jackson, M.: Problem Frames. Analyzing and structuring software development problems. Addison-Wesley (2001)

10. Hatebur, D., Heisel, M.: A UML profile for requirements analysis of dependable software. In: SAFECOMP. (2010) 317–331
11. Côté, I., Hatebur, D., Heisel, M., Schmidt, H., Wentzlaff, I.: A systematic account of problem frames. In: Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP 2007), Universitätsverlag Konstanz (2008)
12. Jackson, M., Zave, P.: Deriving specifications from requirements: an example. In: Proceedings 17th Int. Conf. on Software Engineering, Seattle, USA, ACM Press (1995) 15–24
13. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **10** (2002) 571–588
14. Australian Government - Office of the Privacy Commissioner: Privacy Impact Assessment Guide. Australian Government. (2010) http://www.privacy.gov.au/materials/types/download/9509/6590.
15. Clauß, S., Kesdogan, D., Kölsch, T.: Privacy enhancing identity management: protection against re-identification and profiling. In: Proceedings of the 2005 workshop on Digital identity management. DIM '05, ACM (2005) 84–93
16. Cormode, G., Srivastava, D.: Anonymized data: generation, models, usage. In: Proceedings of the 35th SIGMOD international conference on Management of data. SIGMOD '09, ACM (2009) 1015–1018
17. Kapadia, A., Naldurg, P., Campbell, R.H.: Distributed enforcement of unlinkability policies: Looking beyond the chinese wall. In: Proceedings of the POLICY Workshop, IEEE Computer Society (2007) 141–150
18. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requir. Eng. **16** (March 2011) 3–32
19. Kalloniatis, C., Kavakli, E., Gritzalis, S.: Addressing privacy requirements in system design: the pris method. Requir. Eng. **13** (August 2008) 241–255
20. Hafiz, M.: A collection of privacy design patterns. In: Proceedings of the 2006 conference on Pattern languages of programs. PLoP '06, ACM (2006) 7:1–7:13