A Problem-Based Approach for Computer-Aided Privacy Threat Identification *

Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Rene Meis

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

Abstract. Recently, there has been an increase of reported privacy threats hitting large software systems. These threats can originate from stakeholders that are part of the system. Thus, it is crucial for software engineers to identify these privacy threats, refine these into privacy requirements, and design solutions that mitigate the threats.
In this paper, we introduce our methodology named Problem-Based Privacy Analysis (ProPAn). The ProPAn method is an approach for identifying privacy threats during the requirements analysis of software systems using problem frame models. Our approach does not rely entirely on the privacy analyst to detect privacy threats, but allows a computer aided privacy threat identification that is derived from the relations between stakeholders, technology, and personal information in the system-to-be.
To capture the environment of the system, e.g., stakeholders and other IT systems, we use problem frames, a requirements engineering approach founded on the modeling of a machine (system-to-be) in its environment (e.g. stakeholders, other software). We define a UML profile for privacy requirements and a reasoning technique that identifies stakeholders, whose personal information are stored or transmitted in the system-to-be and stakeholders from whom we have to protect this personal information. We illustrate our approach using an eHealth scenario provided by the industrial partners of the EU project NESSoS.

Keywords: privacy, threat analysis, problem frames, requirements engineering

1 Introduction

Identifying privacy threats to a software system is difficult, because of a lack of structured approaches for identifying stakeholders that have privacy requirements in a system. In addition, finding methods to fulfill these requirements, and fulfilling the functional requirements of the system-to-be at the same time, is even more challenging.

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" [23]. The privacy specification in the ISO 15408 standard - Common

* 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).

Criteria for Information Technology Security Evaluation (or short CC) [16] defines four privacy goals. These goals can be refined into privacy requirements for a given software system. *Anonymity* means that a subject is not identifiable within a set of subjects, the anonymity set. *Unlinkability* of two or more items of interest (IOI) means that within a system the attacker cannot sufficiently distinguish whether these IOIs are related or not. *Unobservability* of an IOI means that an IOI is not detectable by any subject 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*.

In this paper, we introduce our methodology named Problem-Based Privacy Analysis (ProPAn). The ProPAn method provides assistance for the initial steps of any given privacy analysis, which is to figure out those parts of the system, where personal information, we have to protect, can be disclosed by counterstakeholders. We use the problem frame [17] requirements engineering approach to model the machine (system-to-be) in its environment (e.g. stakeholders, other software). We extend the UML4PF framework [5] with a UML profile for privacy requirements and a reasoning technique. This reasoning technique identifies the domains, in which personal information is stored or to which personal information is transmitted. Additionally, our technique identifies the domains, to which counterstakeholders have access. From these identified domains, our technique derives the possible privacy threats of the system-to-be. We illustrate our approach using an eHealth scenario provided by the industrial partners of the EU project *Network of Excellence (NoE) on Engineering Secure Future Internet Software Services and Systems (NESSoS)*[1].

A number of guidelines for privacy are available. *The Fair Information Practice Principles* (– or short FIPs) [20] – are widely accepted. They 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 in 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 [10]. 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 data related to children [12]. These legal guidelines must be implemented by any given software system for which the guidelines apply. Our work supports privacy threat analysis, which has to be performed in order to comply with any of these regulations.

The rest of the paper is organized as follows. Section 2 presents the problem frame approach and our support tool, and Sect. 3 presents the eHealth case study. We introduce our method in Sect. 4, and illustrate its application to the case study in Sect. 5. Section 6 contains related work, and Sect. 7 concludes.

[1] `http://www.nessos-project.eu/`

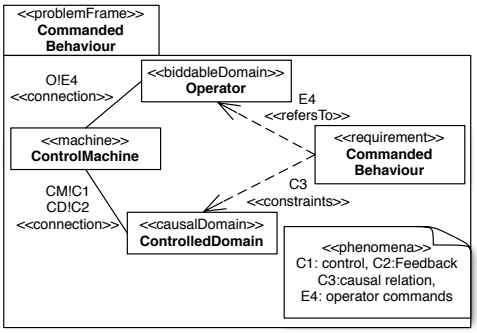

Fig. 1. *Commanded Behaviour* problem frame

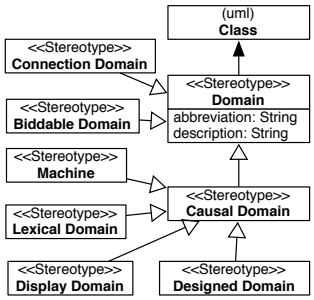

Fig. 2. Inheritance structure of different domain types

2 Background

We use the problem frames approach to build our privacy threat identification on, because problem frames are an appropriate means to analyze not only functional, but also dependability and other quality requirements [13,1].

Problem frames are a means to describe software development problems. They were proposed by Jackson [17], 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 [14]. 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 class of problems.

Figure 1 shows an example of a problem frame. The class with the stereotype ≪machine≫ represents the thing to be developed (e.g., the software). The classes with some domain stereotypes, e.g., ≪causalDomain≫ or ≪biddableDomain≫ represent *problem domains* that already exist in the application environment. Jackson distinguishes the domain types *causal domains* that comply with some physical laws, *lexical domains* that are data representations, and *biddable domains* that are usually people. We use the formal meta model [14] shown in Fig. 2 to annotate domains with their corresponding stereotype.

Domains are connected by interfaces consisting of shared phenomena. 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. For example, in Fig. 1 the notation *O!E4* means that the phenomena in the set *E4* are controlled by the domain **Operator**. These interfaces are represented as associations, and the name of the associations contain the phenomena and the domains controlling the phenomena.

In Fig. 1, the **ControlledDomain** domain is constrained and the **Operator** is referred, because the **ControlMachine** has the role to change the **ControlledDomain**

on behalf of the **Operator**'s commands for achieving the required **Commanded Behaviour**. These relationships are modeled using dependencies that are annotated with the corresponding stereotypes.

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. Other problem frames besides the commanded behavior frame shown in Fig. 1 are *required behaviour*, *simple workpieces*, *information display*, and *transformation* [17].

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. An example is given in Fig. 3. 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*. Examples are given in Fig. 4, 5, and 6.

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 [18] for details). The specification describes the machine and is the starting point for its construction.

The UML4PF framework provides tool support for this approach. A more detailed description can be found in [5].

3 Case Study

To illustrate our approach for identifying privacy threats, we use a scenario taken from the health care domain. It concerns managing **Electronic Health Records (EHR)**s and is provided by the industrial partners of the EU project NESSoS. EHRs contain any information created by health care professionals in the context of the care of a patient. Examples are laboratory reports, X-ray images, and data from monitoring equipment. The information stored in the EHR shall only be accessed with the consent of the patient. The only exception is a medical emergency, in which case the patient's physical status may prevent her from giving the consent. In addition, the information in the EHR supports clinical research.

In Fig. 3 we present a context diagram of the electronic health system (**EHS**). The **EHS** is the machine to be built and the lexical domain **EHR** is directly connected to it. The **EHS** is also connected to the **Patient** using the **Browser Patient**, a **Mobile Device**, which is further connected to **Sensor**s that are in turn attached to the **Patient**. A **Monitor** or the **Browser Care Provider**s connects the machine to the health care professionals **Nurse** and **Doctor**. The **Researcher** uses the **Browser Researcher** to access the **Research Database Application**, which is in turn connected to the **EHS**.

We identified 19 preliminary functional requirements for the EHS, which were refined into 34 functional requirements and corresponding problem diagrams. For reasons of space, we focus on the following requirements for the remainder of the paper, which define the basic functionality of a EHS and include a potential privacy threat:

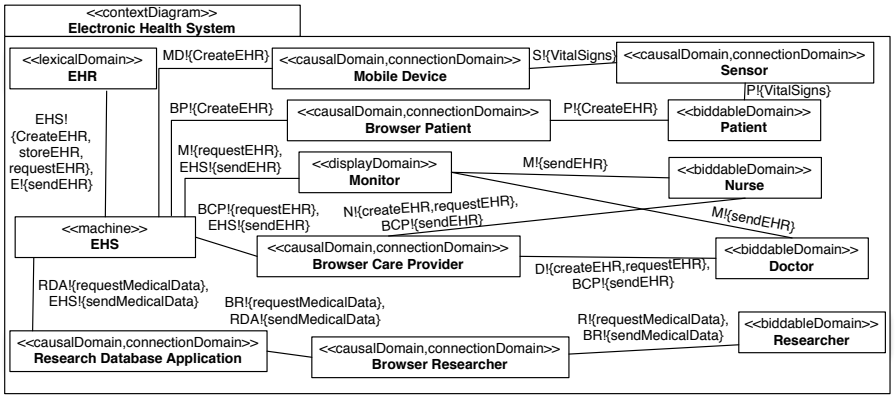

Fig. 3. Context Diagram

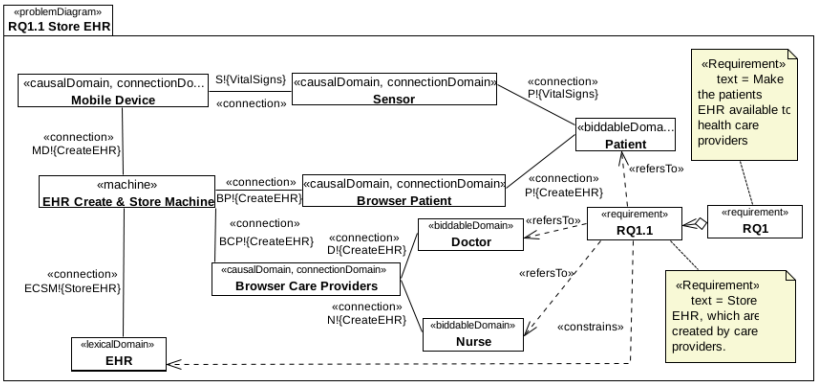

Fig. 4. Problem Diagram for Requirement RQ1.1

RQ1.1 Store **EHR**, which are created by care providers.
RQ1.2 Display **EHR** to care providers as needed
RQ15.1 Send alarms, appointments or instructions using **EHR** from **Doctor** to **Patient**
RQ15.2 Send automated alarms or instructions based on **EHR** analysis to **Patient**
RQ16 Release medical data to **Researcher**s

The problem diagram for RQ1.1 describes creating and storing of **EHR**s (depicted in Fig. 4). The **Patient** is connected to a **Sensor** that reports the **Patient**'s vital signs to the **EHR Create & Store Machine** using a **Mobile Device**. The machine stores the **EHR**. In addition, the **Patient** can use the **Browser Patient** to create an **EHR**. **Doctor**s and **Nurse**s can use the **Browser Care Providers** to command the machine to create **EHR**s.

The problem diagram for RQ1.2 shown in Fig. 5 describes how care providers can access the **EHR**. **Doctor**s and **Nurse**s can either use the **Monitor** or the **Browser Care**

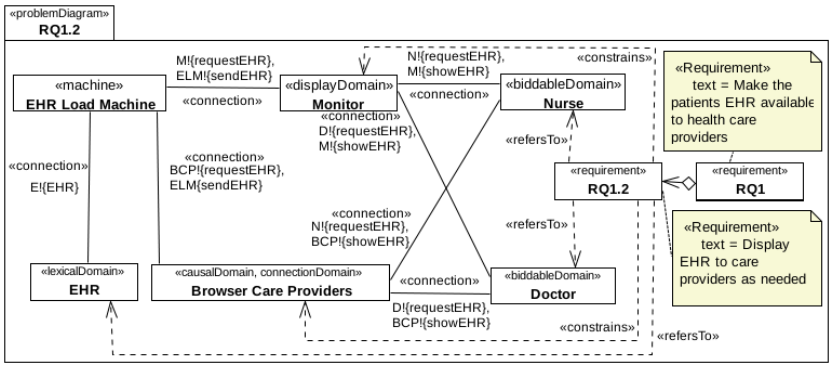

Fig. 5. Problem Diagram for Requirement RQ1.2

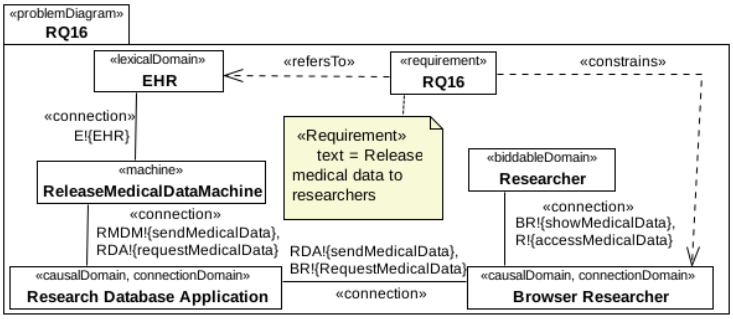

Fig. 6. problem Diagram for Requirement RQ16

Providers to request **EHR**s from the **EHR Load Machine**. The machine can access the **EHR** and display it on either the **Monitor** or the **Browser Care Providers** to the **Doctor**s and **Nurse**s.

The release of medical information to researches described in RQ16 and depicted in the problem diagram in Fig. 6. **Researcher**s can use the **Browser Researcher** to request medical data from the **Research Database Application**. This application requests the data in turn from the **ReleaseMedicalDataMachine**, which releases it the **Research Database Application**. The application sends the information to the browser, where it is shown to the **Researcher**s.

The problem diagrams for RQ15.1 and RQ15.2 are drawn in a similar manner.

4 Method

An overview of the ProPAn method is shown in Fig. 7. It consists of four steps Draw context diagram and problem diagrams, Add privacy requirements to model, Generate privacy threat graphs, and Analyze privacy threat graphs that will be explained in detail in the following.

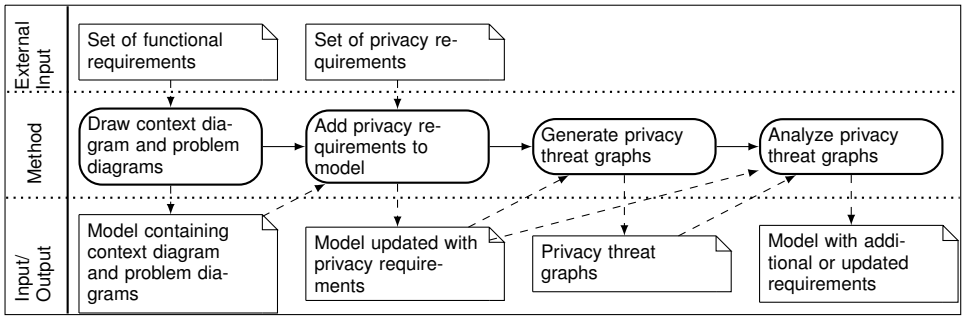

Fig. 7. Illustration of the ProPAn method

4.1 Creation of the Model

The first step of the ProPAn method is to Draw context diagram and problem diagrams for the given Set of functional requirements. For this purpose, we use the UML4PF tool. The result of this step is a Model containing context diagram and problem diagrams. This step follows the basic principles of requirements engineering using the problem frames approach as explained in Sect. 2.

4.2 Privacy Requirements

As a second step we consider the Set of privacy requirements the system-to-be shall enforce. To Add privacy requirements to model we introduce a new stereotype. In Fig. 8 on the left hand side the UML profile is shown. We derived this stereotype from the CC's privacy specification. All privacy specifications follow the pattern that they describe whose privacy shall be protected from whom. For our privacy threat identification it is not necessary to distinguish between the different privacy goals, such as anonymity, unlinkability, unobservability, and pseudonymity (see Sect. 2), but the profile offers the opportunity to easily refine our general privacy stereotype. The general privacy stereotype has three attributes. The attribute stakeholder is the biddable domain whose privacy shall be protected against the domain given in the attribute counterStakeholder. In the attribute Description a more detailed textual description of the privacy requirement can be given. Our threat identification focuses on internal counterstakeholders, i.e. domains that occur in a problem diagram. In contrast to the term *"attacker"* a *"counterstakeholder"* may obtain sensitive data about the stakeholder involuntarily.

The Model containing context diagram and problem diagrams is updated to obtain the Model updated with privacy requirements. For our example, we add the privacy requirement Preserve anonymity with the stakeholder Patient and the counterstakeholder Researcher (see Fig. 8 on the right-hand side).

4.3 Graph Generation

All graphs $(\mathcal{V}, \mathcal{E})$ that we use for our threat identification in the ProPAn method are labeled and directed. The set of vertexes is a subset of the domains occurring in the model,

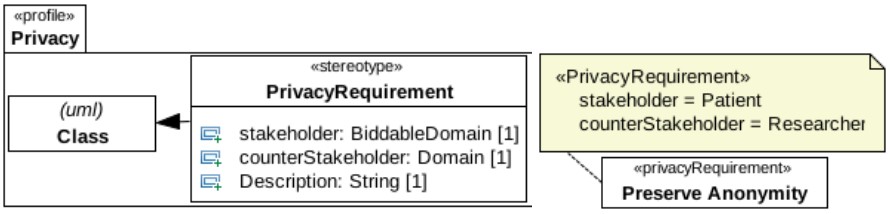

Fig. 8. Privacy Profile (left) and Privacy Requirement (right)

formally $\mathcal{V} \subseteq Domain$. The edges are annotated with problem diagrams and point from one domain to another, formally $\mathcal{E} \subseteq Domain \times ProblemDiagram \times Domain$. In the following we describe a graph $(\mathcal{V}, \mathcal{E})$ only by its edges $\mathcal{E}$.

Global Information Flow Graph To Generate privacy threat graphs we use the Model updated with privacy requirements as an input. From this static model we create the global information flow graph $\mathcal{G}$, which is an over-approximation of the information flow occurring in the system-to-be. An edge (d_1, p, d_2) is in $\mathcal{G}$, iff the domains d_1 and d_2 are not equal, are both part of the problem diagram p, and the domain d_2 is constrained in p. This is expressed using the following formula.

$$\mathcal{G} = \{(d_1, p, d_2) : Domain \times ProblemDiagram \times Domain \mid \\ d_1 \neq d_2 \wedge d_1, d_2 \in p \wedge constr(d_2, p)\}$$

Because of the annotation of the edge we keep the information which problem diagram causes the information flow. Thus, our information flow graph contains traceability links that are used in our further analysis. The semantics of an edge $(d_1, p, d_2) \in \mathcal{G}$ is that in problem diagram p there is possibly an information flow from domain d_1 to domain d_2.

Stakeholder Information Flow Graph We now want to determine where data of the stakeholder s, whose privacy shall be protected against the counterstakeholder c, is possibly processed or stored. Using the global information flow graph $\mathcal{G}$, we can compute the stakeholder information flow graph $\mathcal{S}_s \subseteq \mathcal{G}$. The algorithm for the computation of $\mathcal{S}_s$ is given in Listing 1.1.

The algorithm operates on four sets. During the algorithm the set D contains all domains from whom possibly an additional information flow can occur. D is initialized as the singleton set containing the stakeholder s. The domains, which have already been used by the algorithm, are collected in the set U, which is initially empty. The set P consists of all problem diagrams, which are not yet part of an information flow in the stakeholder information flow graph. The set P is initialized as the set of all problem diagrams. The set E is used to temporally store the edges, which will be added to the stakeholder information flow graph at the end of each iteration. The set $\mathcal{S}_s$ is the resulting stakeholder information flow graph and is initialized as the empty set of edges. Inside the repeat loop, we firstly initialize the set of new edges E as empty (line 5). We

```
1  var D, U : Set[Domain]; P : Set[ProblemDiagram];
2  var E, S_s : Set[Domain × ProblemDiagram × Domain];
3  D := {s}; U := ∅; P := {p : ProblemDiagram}; S_s := ∅;
4  repeat
5    E := ∅;
6    foreach d ∈ D do
7      foreach p ∈ P do
8        E := E ∪ {d' : Domain | (d, p, d') ∈ G • (d, p, d')}
9      endforeach
10   endforeach;
11   U := U ∪ D ;
12   D := {d : Domain | d ∉ U ∧ ∃d' : Domain; p : ProblemDiagram • (d', p, d)∈ E};
13   P := P \ {p : ProblemDiagram | ∃d, d' : Domain • (d,p,d') ∈ E};
14   S_s := S_s ∪ E
15 until D = ∅ ∨ P = ∅ endrepeat
```

Listing 1.1. Algorithm for the computation of the stakeholder information flow graph

then iterate all domains of D (line 6) and all problem diagrams of P (line 7). All edges from $\mathcal{G}$, which start from a domain in D and are annotated with a problem diagram in P, are added to E (line 8). After the iteration of D and P, all domains from D are added to the used Domains U (line 11). Then the set of domains D is updated to the set of domains, which are reachable from the new edges E, but are not in U, i.e., they have not yet been used (line 12). The set of problem diagrams P is reduced by the set of problem diagrams that occur as annotations in the set of new edges E (line 13). At last, the new edges E are added to the stakeholder information flow graph $\mathcal{S}_s$ (line 14). The algorithm terminates when one of the sets D or P is empty (line 15). This is ensured, because each domain and each problem diagram is considered for at most one execution of the repeat loop.

It is sufficient to consider each problem diagram for at most one execution of the repeat loop, because all information flows that would be added later, are redundant to the existing information flows.

Counterstakeholder Graph To determine which information the system-to-be provides to the counterstakeholder c, we generate the counterstakeholder graph $\mathcal{C}_c$. An edge (c, p, d) is in $\mathcal{C}_c$ iff the counterstakeholder c and the domain d both occur in the problem diagram p. We express this using the following formula.

$$\mathcal{C}_c = \{(d_1, p, d_2) : Domain \times ProblemDiagram \times Domain \mid d_1 = c \wedge d_1, d_2 \in p\}$$

Please note that the counterstakeholder graph $\mathcal{C}_c$ is not a subgraph of the global information flow graph $\mathcal{G}$. The semantics of an edge $(c, p, d) \in \mathcal{C}_c$ is that the counterstakeholder c may gain information from the domain d in the problem diagram p, which differs from the semantics of an edge in $\mathcal{G}$.

Privacy Threat Graph Finally, we automatically generate the privacy threat graph $\mathcal{T}_{s,c}$ for the stakeholder s and the counterstakeholder c. This graph connects the stakeholder information flow graph $\mathcal{S}_s$ with the counterstakeholder graph $\mathcal{C}_c$. From $\mathcal{C}_c$ the privacy threat graph $\mathcal{T}_{s,c}$ contains only the edges from $\mathcal{C}_c$, which point to domains, which possibly provide information about the stakeholder s. We call this counterstakeholder subgraph $\mathcal{C}_{c,s} \subseteq \mathcal{C}_c$. The privacy threat graph $\mathcal{T}_{s,c}$ contains only the edges from $\mathcal{S}_s$, which are part of a path from the stakeholder s to a domain that possibly provides information to the counterstakeholder c. We call this stakeholder information flow subgraph $\mathcal{S}_{s,c} \subseteq \mathcal{S}_s$. $\mathcal{T}_{s,c}$ is then the union of $\mathcal{S}_{s,c}$ and $\mathcal{C}_{c,s}$. Formally we have:

$$\begin{aligned}
\mathcal{S}_{s,c} =& \{(d, p, d') : \mathcal{S}_s \mid \exists (e, p', e') : \mathcal{C}_c;\ (d_1, p_1, d_2), \ldots, (d_{n-1}, p_{n-1}, d_n) : \mathcal{S}_s \bullet \\
& \qquad d' = d_1 \wedge d_n = e'\} \\
\mathcal{C}_{c,s} =& \{(e, p, e') : \mathcal{C}_c \mid \exists (d, p', d') : \mathcal{S}_s \bullet e' = d'\} \\
\mathcal{T}_{s,c} =& \mathcal{S}_{s,c} \cup \mathcal{C}_{c,s}
\end{aligned}$$

Thus we generate in the Phase Generate privacy threat graphs one Privacy Threat Graph for each privacy requirement, which is part of the Model updated with privacy requirements. All graph generations are performed automatically using OCL expressions. The details of the automatic graph generation are described in Sect. 4.5.

4.4 Analysis

For the Analyze privacy threat graphs we have to note that in the Privacy threat graphs we can distinguish two kinds of edges:

1. Edges (c, p, d) have the semantics that the counterstakeholder c may gain information from the domain d in problem diagram p.
2. All edges (d_1, p, d_2) with $d_1 \neq c$, i.e. the edges do not start from the counterstakeholder c, have the semantics that information is possibly transferred from domain d_1 to d_2 in problem diagram p.

In the analysis we have to take a closer look at all domains d for which a problem diagram p exists such that $(c, p, d) \in \mathcal{T}_{s,c}$. For each such domain d, we may have found a privacy threat against the stakeholder s from the counterstakeholder c. The threat graph $\mathcal{T}_{s,c}$ provides us two different kinds of information according to the two different kinds of edges in the graph as mentioned above.

1. We provide the information which requirements may allow the counterstakeholder c to gain information from the domain d about the stakeholder s. These requirements are contained in the problem diagrams p with $(c, p, d) \in \mathcal{T}_{s,c}$.
2. How the information about the stakeholder s got to the domain d is described through the path $(s, p_1, d_1), \ldots, (d_{n-1}, p_n, d)$ where the requirements in the problem diagrams p_i should explain which information is transferred to which domain.

Since our approach is an over-approximation of the system's actual information flow, there can be edges that actually do not represent an information flow in the system. These edges can be identified manually by studying the problem diagram and the

requirement expressed in it. Edges that after the removal of the above-mentioned edges are no longer part of a path starting from the stakeholder, can be removed. To solve the threats in the refined threat graph $\mathcal{T}'_{s,c}$ there are again two possibilities, which also can be used in combination.

1. For an edge $(c, p, d) \in \mathcal{T}'_{s,c}$, the requirement in the problem diagram p can be modified or an additional requirement can be added, expressing that the counterstakeholder c is not able to gain personal information of the stakeholder s via the domain d.
2. For an edge $(d_1, p, d_2) \in \mathcal{T}'_{s,c}$ with $d_1 \neq c$, the requirement in the problem diagram p can be modified or an additional requirement can be added, expressing that no personal information of the stakeholder s is processed or stored by the domain d_2.

The outcome of our last step Analyze privacy threat graphs is the Model with additional or updated requirements created from the Model updated with privacy requirements and the Privacy threat graphs.

4.5 Technical Realization

For the generation of the graphs explained in Sect. 4.3 we developed the ProPAn tool[2]. This tool generates the four kinds of graphs, namely the global information flow graph, the stakeholder information flow graph, the counterstakeholder access graph, and the privacy threat graph, from a UML model. This model has to contain problem diagrams and has to be annotated with the stereotypes presented in Sects. 2 and 4.2. The graphs are generated automatically without user interaction for all privacy requirements, stakeholders, and counterstakeholders, respectively using OCL expressions.

The generation of the global information flow graph $\mathcal{G}$ is done by the OCL expression shown in Listing 1.2. This OCL expression first defines the sets of all domains doms (lines 1-3) and all problem diagrams pds (lines (4-6) of the model. For the generation of $\mathcal{G}$, the expression iterates the set of all problem diagrams pds (line 8). For each domain $p \in$ pds the sets pdoms (lines 9-12) and pcdoms (lines 13-17) are generated. The set pdoms consists of all domains that are part of the problem diagram p. The set pcdoms includes all constrained domains of the problem diagram p. Then the elements of pcdoms and pdoms are iterated (lines 18-19). For all $c \in pcdoms$ and $d \in pdoms$ with $d \neq c$ an edge (d, p, c) is added to the global information flow graph $\mathcal{G}$ (line 20).

For the generation of the global information flow graph $\mathcal{G}$, we iterate all problem diagrams and for each problem diagram, we iterate all domains that are part of it and all constrained domains in it. Hence, $\mathcal{G}$ contains at most $\#ProblemDiagram \cdot \#Domain^2$ many edges. Thus, the asymptotic time complexity for the generation of all privacy threat graphs is in $\mathcal{O}(\#ProblemDiagram \cdot \#Domain^2)$, because for the generation of all other graphs, the global information flow graph is used.

For the generation the ProPAn tool uses the Eclipse Platform [7], the Acceleo model to text framework [8] and the graph layout tool GraphViz [4]. Each generator is realized as Acceleo template, which generates a dot file. An Acceleo template uses OCL to query

[2] `http://www.uni-due.de/swe/apf12.shtml`

```
let doms: Set(Class) =
  ProblemFrames::Domain.allInstances().base_Class
in
let pds: Set(Package) =
  ProblemFrames::ProblemDiagram.allInstances().base_Package
in
let G: Set(Tuple(d1 : Class, pd : Package, d2 : Class)) =
  pds→iterate(p; A: Set(Tuple(d1 : Class, pd : Package, d2 : Class)) = Set{} |
    let pdoms: Set(Class) =
      p.member→select(oclIsTypeOf(Association)).member.type→asSet()
      →intersection(doms)→asSet()
    in
    let pcdoms: Set(Class) =
      p.member→select(oclIsTypeOf(Dependency))
      →select(getAppliedStereotypes().name→includes('constrains'))
      .target→select(oclIsTypeOf(Class))→asSet()
    in
    A→union(pcdoms→iterate(c; B: Set(Tuple(d1 : Class, pd : Package, d2 : Class)) = Set{} |
      B→union(pdoms→iterate(d; C: Set(Tuple(d1 : Class, pd : Package, d2 : Class)) = Set{} |
        if d≠c then C→including(Tuple{d1=d, pd=p, d2=c}) else C endif)))))
in G
```

Listing 1.2. OCL Expression for Global Information Flow Graph

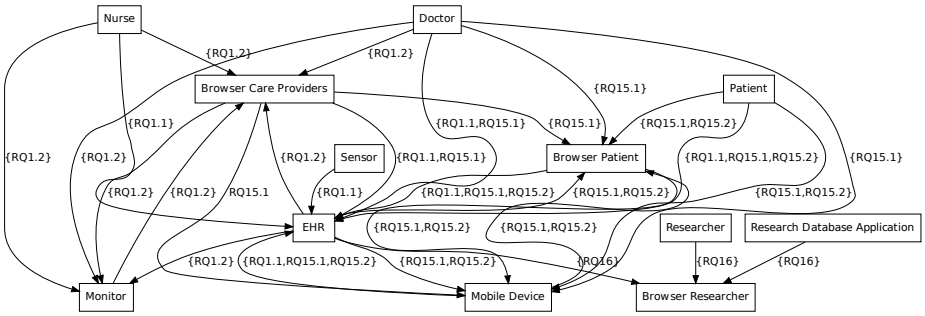

Fig. 9. Global Information Flow Graph

a model and transforms the query results in a file format defined by the template. In our case it is the dot file format which can be read by Graphviz to generate different kinds of graphical representations. These templates are exposed as plug-ins to the Eclipse Front-End. So they directly integrate with modeling tools like Papyrus [3] and UML4PF [5].

5 Application on the Case Study

In this section, we will apply our method described in Sect. 4 on the NESSoS EHS case study introduced in Sect. 3.

5.1 Graph Generation

Figure 9 shows the global information flow graph, which the ProPAn tool generated from the problem diagrams of the 5 requirements mentioned in Sect. 3. For a better readability, we joined the edges starting from and ending at the same domain and annotate

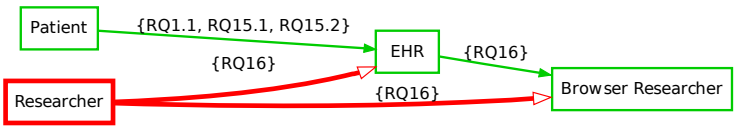

Fig. 10. Threat Graph for Stakeholder Patient and Counterstakeholder Researcher

the resulting edge with the set of problem diagrams, the joined edges were annotated with.

To analyze if researchers may gain information about patients in the EHS, we added a privacy requirement, which is shown in Fig. 8 in Sect. 4.2 on the right hand side. Figure 10 shows the threat graph for the stakeholder patient and the counterstakeholder researcher, which the ProPAn tool generated from the global information flow graph and the privacy requirement. The red part with bold edges and white arrowheads depicts the counterstakeholder graph, and the green parts depict the stakeholder graph. To improve readability, we again aggregated all edges that start from the same domain and also end at the same to domain to one edge, annotated with the set of problem diagrams of the aggregated edges.

5.2 Analysis of the Threat Graph

As mentioned in Sect. 4.4, we have two possibilities to solve the privacy threats that the threat graph identifies. We can consider the edges starting from the counterstakeholder and restrict the information the counterstakeholder can access or we consider the other edges of the threat graph and restrict the information flow between the domains.

The threat graph $\mathcal{T}_{\mathsf{Patient},\mathsf{Researcher}}$ in Fig. 10 has only two edges from the counterstakeholder Researcher to the other domains. From these red, bold edges with white arrowheads, we can see that in the problem diagram for requirement RQ16 (see Fig. 6) the researcher may gain information from the lexical domain EHR and the connection domain Browser Researcher. One possibility to resolve this threat would be to modify requirement RQ16 in such a way that the medical data released to researchers has to be anonymized or pseudonymized.

The other possibility to resolve the privacy threat would be to update all requirements that lead from the Patient to the lexical domain EHR in such a way that they forbid to write personal information about the patient into it. Which problem diagrams have to be considered for this, is shown by the annotations of the green edges that are part of a path from the Patient to the EHR. In Fig. 10, there is only one path from the Patient to the EHR, which is annotated with the requirements RQ1.1, RQ15.1, and RQ15.2. To solve the privacy threat we could update those requirements such that the medical data of the patient is stored in an anonymized or pseudonymized way in the electronic health record.

However, since the electronic health record of a patient has to contain personal information, we restrict the information the Researcher gets in requirement RQ16. We change requirement RQ16 to: “Release medical data pseudonymized to researchers.”

6 Related Work

Deng et al. [6] present 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. Hence, privacy threat identification for an existing data flow model is simplified, because for each data flow element in a model only the threats shown in the tree need to be considered. The work differs from our own, because the privacy threat identification has to be carried out manually.

The PriS method [19] elicits privacy requirements in the software design phase. Privacy requirements are modeled as organizational goals. Furthermore, privacy process patterns are used to identify system architectures, which support the privacy requirements. The PriS method starts with a conceptual model, which also considers enterprise goals, stakeholders, privacy goals, and processes. It 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. In addition, the PriS method has to be carried out manually.

Hafiz [11] describes 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. The patterns are specified in several categories. Among them are intent, motivation, context, problem and solution, as well as forces, design issues and consequences. This work focuses on privacy issues on the network layer and can complement our work in this area.

Asnar et al. [2] present a computer-aided approach to detect security threats based upon SI* models. The authors present patterns that can be used to identify specific areas in the models that present a security threat. The authors investigate access control permissions and search for roles in the models that have more permissions than they require to fulfill their goal. This analysis is carried out semi-automatically using graph patterns. The difference to our work is that we focus on privacy threat detection and we base our work upon the problem frames approach instead of SI*.

7 Conclusions

In this paper, we have presented the ProPAn method. The ProPAn is a problem-based approach for semi-automatic identification of privacy threats during the requirements analysis of software systems. The privacy threats are derived from potential access of counterstakeholders to personal information that are part of the system-to-be. We have extended the problem frames approach with the ProPAn tool[2], which consist a UML profile for privacy and a privacy threat graph generator.

The privacy threat graph helps to determine where personal information of a stakeholder may be processed and stored across the boarders of the problem diagrams, which

[2] `http://www.uni-due.de/swe/apf12.shtml`

correspond to subproblems of the overall development task. It also shows from which domains the counterstakeholder may gain information about the stakeholder. In our example the threat graph showed us that the requirement RQ16 contains a privacy threat violating the privacy requirement Preserve Anonymity.

Our graphs have formal semantics, which are strongly related to the problem frames approach. The generation of the threat graph for a privacy requirement, which is formulated with our introduced stereotype. The generation is performed automatically by the ProPAn tool from the problem diagrams, which represent the requirements the system-to-be has to fulfill. Our privacy threat identification is independent of the actual privacy goal, such as anonymity, unlinkability, unobservability, and pseudonymity, and gives guidance to detect possible privacy threats as early as possible in the software engineering process.

In summary, the ProPAn method has the following advantages:

- The privacy threat identification is re-usable for different projects.
- The privacy threat graph are generated automatically using our proposed tool chain.
- The identified privacy threats can be traced to a specific (sub-)problem of the system-to-be.
- The (sub-)problems can be enhanced with privacy requirements that constrain the functional requirements. Thus, we provide guidance where to apply privacy enhancing technologies.

Since our approach relies on the information flow inside the system-to-be, it can only detect those privacy threats that stem from an information flow starting from a stakeholders to a counterstakeholders, who both are part of system-to-be.

In the future, we plan to elaborate more on the later phases of software development. For example, we want to apply our approach to software components that were developed with the ProPAn method. We aim to identify privacy threats for existing architectures and propose solutions for these problems. The knowledge gathered during the usage of the approach might lead to the discovery of privacy threat patterns. Moreover, we want to extend the ProPAn method to support counterstakeholders that are not part of the system-to-be, but external attackers. We plan to provide extensions of the ProPAn method and tool that allows one to model the capabilities of these attackers and engineer a reasoning method for deciding if a privacy mechanism can protect a system against a certain attacker.

References

1. Azadeh Alebrahim, Denis Hatebur, and Maritta Heisel. A method to derive software architectures from quality requirements. In Tran Dan Thu and Karl Leung, editors, *Proceedings of the 18th Asia-Pacific Software Engineering Conference (APSEC)*, pages 322–330. IEEE Computer Society, 2011.
2. Yudis Asnar, Tong Li, Fabio Massacci, and Federica Paci. Computer aided threat identification. In *Proceedings of the 2011 IEEE 13th Conference on Commerce and Enterprise Computing*, CEC '11, pages 145–152. IEEE Computer Society, 2011.
3. Atos Origin. *Papyrus UML Modelling Tool*, Feb 2011. http://www.papyrusuml.org/.

4. AT&T and Bell-Labs. Graphviz - Graph Visualization Software, June 2012. http://www.graphviz.org.
5. Isabelle Côté, Denis Hatebur, Maritta Heisel, and Holger Schmidt. UML4PF – a tool for problem-oriented requirements analysis. In *Proceedings of the International Conference on Requirements Engineering (RE)*, pages 349–350. IEEE Computer Society, 2011.
6. Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requir. Eng.*, 16:3–32, March 2011.
7. Eclipse Foundation. *Eclipse - An Open Development Platform*, 2011. http://www.eclipse.org/.
8. Eclipse Foundation. *Acceleo - transforming models into code*, June 2012. http://www.eclipse.org/acceleo/.
9. Eclipse Foundation. Eclipse Modeling Framework Project (EMF), June 2012. http://www.eclipse.org/modeling/emf/.
10. 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.
11. Munawar Hafiz. A collection of privacy design patterns. In *Proceedings of the 2006 conference on Pattern languages of programs*, PLoP '06, pages 7:1–7:13. ACM, 2006.
12. M. Hansen, A. Schwartz, and A. Cooper. Privacy and Identity Management. *Security & Privacy, IEEE*, 6(2):38–45, 2008.
13. Denis Hatebur and Maritta Heisel. A foundation for requirements analysis of dependable software. In Bettina Buth, Gerd Rabe, and Till Seyfarth, editors, *Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, volume 5775 of *LNCS*, pages 311–325. Springer Berlin / Heidelberg / New York, 2009.
14. Denis Hatebur and Maritta Heisel. A UML profile for requirements analysis of dependable software. In *SAFECOMP*, pages 317–331, 2010.
15. Denis Hatebur, Maritta Heisel, and Holger Schmidt. A formal metamodel for problem frames. In *Proceedings of the International Conference on Model Driven Engineering Languages and Systems (MODELS)*, volume 5301, pages 68–82. Springer Berlin / Heidelberg, 2008.
16. 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.
17. M. Jackson. *Problem Frames. Analyzing and structuring software development problems*. Addison-Wesley, 2001.
18. M. Jackson and P. Zave. Deriving specifications from requirements: an example. In *Proceedings 17th Int. Conf. on Software Engineering, Seattle, USA*, pages 15–24. ACM Press, 1995.
19. Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. Addressing privacy requirements in system design: the PriS method. *Requir. Eng.*, 13:241–255, August 2008.
20. 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.
21. UML Revision Task Force. *Object Constraint Language Specification*. Object Management Group (OMG), May 2006.
22. UML Revision Task Force. *OMG Unified Modeling Language: Superstructure*, Feb. 2009. http://www.omg.org/docs/formal/09-02-02.pdf.
23. Alan F. Westin. *Privacy and Freedom*. Atheneum, New York, 1967.