information

MDPI

Article

Computer-Aided Identification and Validation of Privacy Requirements

Rene Meis * and Maritta Heisel

paluno—The Ruhr Institute for Software Technology, University of Duisburg-Essen, Duisburg 47057, Germany; maritta.heisel@uni-due.de

* Correspondence: rene.meis@uni-due.de; Tel.: +49-203-379-4503

Academic Editor: Eduardo B. Fernandez

Received: 15 February 2016; Accepted: 21 May 2016; Published: 26 May 2016

Abstract: Privacy is a software quality that is closely related to security. The main difference is that security properties aim at the protection of assets that are crucial for the considered system, and privacy aims at the protection of personal data that are processed by the system. The identification of privacy protection needs in complex systems is a hard and error prone task. Stakeholders whose personal data are processed might be overlooked, or the sensitivity and the need of protection of the personal data might be underestimated. The later personal data and the needs to protect them are identified during the development process, the more expensive it is to fix these issues, because the needed changes of the system-to-be often affect many functionalities. In this paper, we present a systematic method to identify the privacy needs of a software system based on a set of functional requirements by extending the problem-based privacy analysis (ProPAn) method. Our method is tool-supported and automated where possible to reduce the effort that has to be spent for the privacy analysis, which is especially important when considering complex systems. The contribution of this paper is a semi-automatic method to identify the relevant privacy requirements for a software-to-be based on its functional requirements. The considered privacy requirements address all dimensions of privacy that are relevant for software development. As our method is solely based on the functional requirements of the system to be, we enable users of our method to identify the privacy protection needs that have to be addressed by the software-to-be at an early stage of the development. As initial evaluation of our method, we show its applicability on a small electronic health system scenario.

Keywords: privacy; privacy requirements; privacy analysis; requirements engineering; computer-aided software engineering

1. Introduction

Privacy is a software quality that is closely related to security and that is gaining more and more attention in the public. Security is in general concerned with the protection of *assets* that are important in the context of the considered system against malicious attackers that want to get access to the assets, influence the content of the assets or affect the assets' availability. In contrast, privacy is concerned with the protection of *personal data*. We use the following definition of personal data from the European Commission [1]: "*'personal data' means any information relating to a data subject*". Throughout the paper, we use the terms personal information and personal data synonymously. To address privacy, personal data shall be protected against malicious and also unintended processing, disclosure, or alternation. Such events may even be caused by the end-users themselves due to insufficient privacy awareness, or due to a lack of controls. Hence, privacy includes security properties that are limited to the protection of personal data as assets, but the protection needs of these contained security properties are not limited to attackers with malicious intentions. Additionally, privacy aims at increasing the awareness of users concerning how their personal data is processed by the considered system.

The need for security protection is more obvious to companies than the need for privacy protection, especially if the personal data are not the relevant assets. Additionally, companies do not see the benefit of increasing the awareness of end-users if end-users' personal data are the central asset in the system. This is, because increasing the awareness may lead to the situation that end-users provide less personal data. This leads to the situation that companies often underestimate the value of a thorough consideration of privacy during software development. But the costs caused by data breaches increased over the last years in all countries all over the world [2,3]. This entails that end-users lose trust in service providers that handle their data, and call for more transparency on how their data is processed [4]. A systematic privacy analysis of the *system-to-be* (software to be developed) can prevent the occurrence of data breaches and can provide means to identify how and about what kinds of data processing end-users have to be informed.

The above mentioned end-user concerns already show that the software quality privacy is not limited to the protection of data to prevent data breaches. Privacy also aims at increasing the awareness of end-users about how their data is handled and to empower them to keep the control over their data. Hansen *et al.* [5] propose six protection goals for privacy engineering. These are the three classical security goals confidentiality, integrity, and availability, as well as the protection goals unlinkability, transparency, and intervenability. In comparison to the classical security goals and unlinkability, transparency and intervenability got less attention in research as we found out during a systematic literature review [6]. Transparency is concerned with increasing the awareness of end-users by informing them about how their data is processed. Intervenability is about empowering end-users by providing means to them to control how their data is processed. In this paper, we consider all six protection goals for the identification of privacy requirements.

A privacy analysis especially for complex systems is a difficult and error prone task. The privacy analysis should be integrated into the development process as proposed by Cavoukian's privacy principles [7]. This integration shall lead to a reduction of the effort that has to be put into the privacy analysis. Additionally, it shall lead to the detection of privacy issues at the earliest stage possible. The existing methods for privacy requirements engineering provide only little support for the identification of privacy requirements and lack of tool support that automates the identification of privacy requirements (*cf.* Section 10). In order to handle complex systems, more guidance and automation is needed to systematically identify the privacy requirements that have to be satisfied by the system at hand.

In this paper, we present a method to systematically derive privacy requirements from a set of functional requirements represented as problem diagrams [8]. The proposed method is an extension of the Problem-based Privacy Analysis (ProPAn) method [9]. This extension adds the possibility to identify and validate privacy requirements in a systematic way, based on a set of functional requirements. To reduce the effort to perform our method, we extended the ProPAn tool [10] to provide as much automation as possible for the identification, generation, and validation of the privacy requirements. Based on our initial evaluation, we expect that our method supports its users to perform as complete and coherent privacy analyses as possible with a reasonable effort even for complex systems.

Our paper is structured as follows. Section 2 introduces an electronic health system scenario that we use as a running example throughout the paper. Section 3 discusses the relevant parts of the problem frames approach as background of this paper. Section 4 provides an overview of our method, and Sections 5–8 provide the details of the four steps of our method. Our tool support is described in Section 9. Section 10 discusses related work, and Section 11 concludes the paper and provides future directions.

2. Running Example

We use a subsystem of an electronic health system (EHS) scenario provided by the industrial partners of the EU project *Network of Excellence (NoE) on Engineering Secure Future Internet Software*

Services and Systems (*NESSoS*) [11] to illustrate our method. This scenario is based on the German health care system which uses health insurance schemes for the accounting of treatments.

The EHS is the software to be built. It has to manage electronic health records (EHR) which are created and modified by doctors (functional requirement R1) and can also be browsed by doctors (R2). Additionally, the EHS shall support doctors to perform the accounting of treatments patients received. The accounting is based on the treatments stored in the health records. Using an insurance application it is possible to perform the accounting with the respective insurance company of the patient. If the insurance company only partially covers the treatment a patient received, the EHS shall create an invoice (R3). The billing is then handled by a financial application (R4). Furthermore, mobile devices shall be supported by the EHS to send instructions and alarms to patients (R5) and to record vital signs of patients (R6). Finally, the EHS shall provide anonymized medical data to researchers for clinical research (R7).

3. Background

Problem frames are a requirements engineering approach proposed by Jackson [8]. The system-to-be (called *machine*) and its interfaces to the environment, which consists of *domains*, are represented in a *context diagram*. Jackson distinguishes the domain types *causal domains* that comply with some physical laws, *lexical domains* that are data representations, *biddable domains* that are usually people, and *connection domains* that mediate between two domains. Additionally, Jackson distinguishes between *given domains* that already exist in the environment of the machine and *designed domains* that are part of the system-to-be and whose behavior and structure can be defined by developers of the machine. The problem to be solved by the machine is decomposed until subproblems are reached which fit to problem frames. Problem frames are patterns for frequently occurring problems. An instantiated problem frame is represented as a *problem diagram*. A problem diagram visualizes the relation of a requirement to the environment of the machine and how the machine can influence these domains. A requirement can refer to and constrain phenomena of domains. Phenomena are events, commands, states, information, and the like. Both relations are expressed by dependencies from the requirement to the respective domain annotated with the referred to or constrained phenomena. Connections (associations) between domains describe the phenomena they share. Both domains can observe the shared phenomena, but only one domain has the control over a phenomenon (denoted by a "!").

We use the UML4PF-framework [12] to create problem frame models as UML class diagrams enriched with stereotypes. All diagrams are stored in *one* global UML model. Hence, we can perform analyses and consistency checks over multiple diagrams and artifacts. The context diagram (in UML notation) for the EHS is shown in Figure 1. The context diagram shows that the machine *EHS* (class with stereotype «machine») contains the designed domains *Invoice* and *EHR* (classes with stereotype «designedDomain»). All other domains in the context diagram are given domains. As given domains, we have the four causal connection domains *FinancialApplication*, *InsuranceApplication*, *MobileDevice*, and *ResearchDatabaseApplication* that represent devices or existing applications that are used by the EHS, and the three biddable domains *Patient*, *Doctor*, and *Researcher* that represent people that (indirectly) interact with the EHS. The *EHS* is directly or indirectly (through connection domains) connected to all domains of the context diagram (indicated by associations with stereotype «connection»). The problem diagram (in UML notation) for the functional requirement *R6* is shown in Figure 2. The problem diagram is about the problem to build the submachine *Record* that records the vital signs of *Patient*s sent to it via *MobileDevice*s in the corresponding *EHR*s. The functional requirement *R6* refers to the patient from whom the vital signs are recorded and to the mobile device which forwards the vital signs (indicated by a dependencies with stereotype «refersTo»), and the requirement constrains the EHR to store the recorded vital signs in the corresponding health record of the patient (indicated by dependencies with stereotype «constrains»).

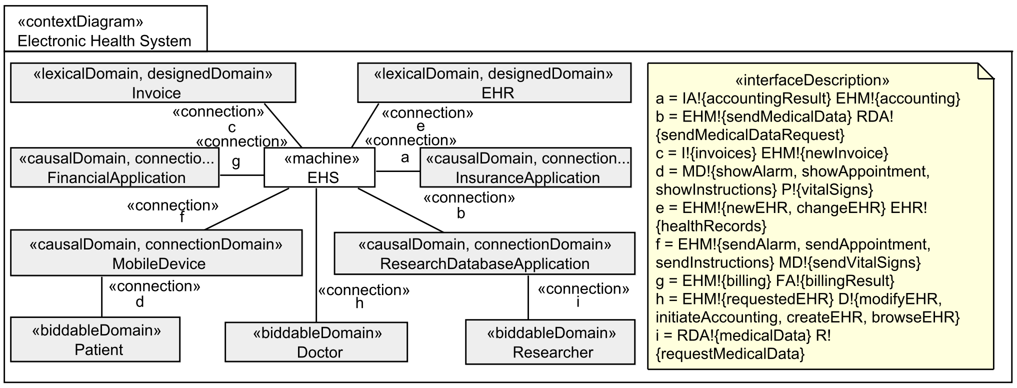

«contextDiagram»
Electronic Health System
«lexicalDomain, designedDomain» Invoice
«lexicalDomain, designedDomain» EHR
«connection» c
«connection» e
«connection»
«connection»
«causalDomain, connectio... FinancialApplication
g
«machine» EHS
a
«causalDomain, connection... InsuranceApplication
«connection» f
«connection» b
«causalDomain, connectionDomain» MobileDevice
«causalDomain, connectionDomain» ResearchDatabaseApplication
«connection» d
«connection» h
«connection» i
«biddableDomain» Patient
«biddableDomain» Doctor
«biddableDomain» Researcher
«interfaceDescription»
a = IA!{accountingResult} EHM!{accounting}
b = EHM!{sendMedicalData} RDA!{sendMedicalDataRequest}
c = I!{invoices} EHM!{newInvoice}
d = MD!{showAlarm, showAppointment, showInstructions} P!{vitalSigns}
e = EHM!{newEHR, changeEHR} EHR!{healthRecords}
f = EHM!{sendAlarm, sendAppointment, sendInstructions} MD!{sendVitalSigns}
g = EHM!{billing} FA!{billingResult}
h = EHM!{requestedEHR} D!{modifyEHR, initiateAccounting, createEHR, browseEHR}
i = RDA!{medicalData} R!{requestMedicalData}

Figure 1. Context diagram for the EHS scenario.

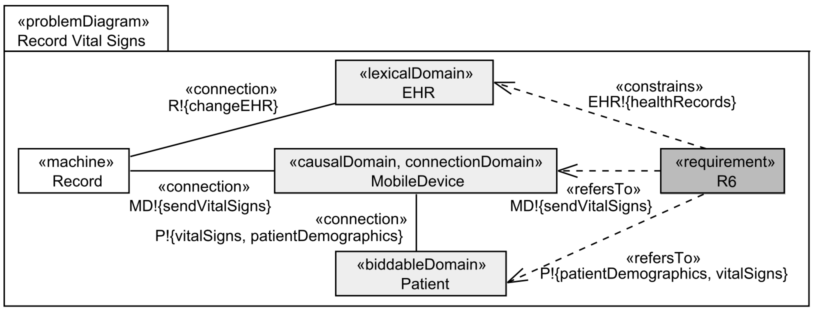

«problemDiagram»
Record Vital Signs
«lexicalDomain» EHR
«connection» R!{changeEHR}
«constrains» EHR!{healthRecords}
«machine» Record
«causalDomain, connectionDomain» MobileDevice
«requirement» R6
«connection» MD!{sendVitalSigns}
«refersTo» MD!{sendVitalSigns}
«connection» P!{vitalSigns, patientDemographics}
«biddableDomain» Patient
«refersTo» P!{patientDemographics, vitalSigns}

Figure 2. Problem diagram for functional requirement R6.

4. Overview of our Method

The UML2 activity diagram shown in Figure 3 visualizes our method to identify and validate privacy requirements based on a set of functional requirements. We will refer to the actions of the activity diagrams using the term *step*. Our method has to be carried out jointly by a requirements engineer, knowing the functional requirements, a privacy expert, knowing the privacy needs for the system under consideration, and an application domain expert, knowing the application domain of the system under consideration. In the following, we will refer to these persons using the term *user*.

Our method builds upon a central UML model, called *ProPAn Model*, which is used to provide the inputs and to store the outputs for all steps of our method. To be able to model all artifacts needed for our method, we extended the UML using profiles that define stereotypes.

In the following, we provide an overview of the four steps of our methods and briefly describe what is done in the steps and for which purpose, and how the steps are connected to each other. The details on how the steps are carried out are described in Sections 5–8.

The first step shown in Figure 3 (*Analyze Flow of Personal Data*) consists of several sub-steps that we presented in previous work. In this step, we identify privacy relevant domain knowledge as introduced in [13,14], we generate different kinds of graphs that visualize possible privacy issues implied by the functional requirements and the identified domain knowledge as introduced in [9,15], and we identify the personal data that are processed by the system under consideration, how it flows through the system and at which places (domains) the personal data are available and in which quality it is available there as introduced in [15]. These outputs produced during the first step form the foundation of the following steps.

The following steps form the main contribution of this paper.

In the second step of our method (*Generate Privacy Requirements Candidates*), we use the identified *flow of personal data* and the information about the *personal data at domains* to automatically generate the *privacy requirements* that are implied by the provided input. In this paper, we consider the generation of privacy requirements related to the protection goals for privacy engineering proposed by Hansen [5]. These protection goals include the classical security goals *confidentiality*, *integrity*, and *availability* and the privacy goals *unlinkability*, *transparency*, and *intervenability*.

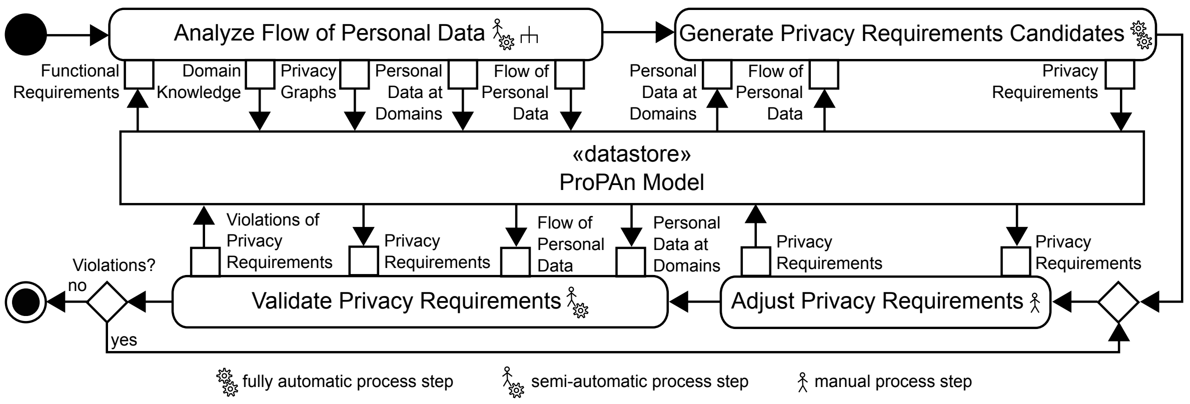


Figure 3. Overview of our proposed method.

In the third step *Adjust Privacy Requirements*, the user has to review, complete, and adjust the generated *privacy requirements*. This is needed, because the generated privacy requirements may lack of details that are not extractable from the ProPAn model or the generated requirements are considered to be incomplete, too strong, or too weak. As result of this step, we obtain a set of revised privacy requirements.

The revised privacy requirements are validated automatically in the fourth step of our method. It is, for example, checked whether the manually adjusted privacy requirements are still consistent with the flow of personal data and the availability of personal data at the different domains as documented in the ProPAn model. The tool support provides the user information about the kinds of violations of the consistency of the privacy requirements and the elicited information flows. Based on this information, the user has to decide, whether the privacy requirements have to be adjusted again, or whether the presented consistency violations are no violations or acceptable violations.

In the following, we will discuss all steps of our method in detail.

5. Analyze Flow of Personal Data

This step summarizes several sub-steps that we already published in previous work. The results of these sub-steps form the foundation of the steps that we introduce in this paper. Figure 4 shows the sub-steps of the step *Analyze Flow of Personal Data* from the overview of our method shown in Figure 3.

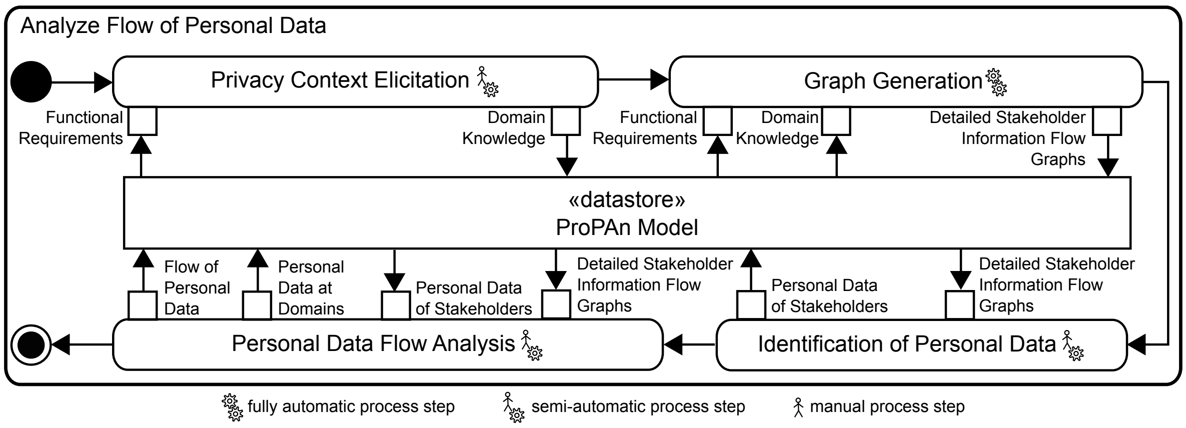

Analyze Flow of Personal Data
Privacy Context Elicitation
Graph Generation
Functional Requirements
Domain Knowledge
Functional Requirements
Domain Knowledge
Detailed Stakeholder Information Flow Graphs
«datastore» ProPAn Model
Flow of Personal Data
Personal Data at Domains
Personal Data of Stakeholders
Detailed Stakeholder Information Flow Graphs
Personal Data of Stakeholders
Detailed Stakeholder Information Flow Graphs
Personal Data Flow Analysis
Identification of Personal Data
fully automatic process step
semi-automatic process step
manual process step

Figure 4. Refinement of the step Analyze Flow of Personal Data.

5.1. Privacy Context Elicitation

First, we elicit privacy-relevant domain knowledge based on questionnaires to identify (1) indirect stakeholders, *i.e.*, people of whom personal information is processed by the system under consideration, but who are not explicitly mentioned in the functional requirements; (2) indirect counterstakeholders, *i.e.*, people who possibly are able to access personal data processed by the system under consideration with or without malicious intentions; and (3) implicit information flows, *i.e.*, information flows that possibly occur between domains of the system under consideration, but that are not explicitly captured in the functional requirements as introduced in [13]. Additionally, we investigate whether interfaces between domains should be refined to identify connection domains which may cause privacy issues as introduced in [14] for the example of clouds as connection domains that possibly introduce privacy problems.

The identified domain knowledge is represented in so-called *domain knowledge diagrams*. These are similar to Jackson's problem diagrams, but they do not contain a machine and instead of a requirement they contain *facts* and *assumptions*. Facts and assumptions are both statements about domains that are indicative. Similar to requirements, facts and assumptions also refer to and constrain domains. The difference between facts and assumptions is, that a fact is a statement that is always true and an assumption is a statement that under certain circumstances may be violated. For example, most statements about the behavior of people (biddable domains) are assumptions, because people may show a different behavior than expected. In the following, we will use the term *statement* as a term that includes requirements, facts, and assumptions.

Application to Running Example

For the EHS scenario, we did not identify any privacy-relevant connection domains, but we identified additional indirect counterstakeholders and implicit information flows.

For example, we identified for the insurance application and the mobile device the domain knowledge shown in Figure 5. Assumption A4 documents that there is an information flow between the patient and the insurance application that is not covered by our requirements. A4 says that information about the patient's insurance contracts, including contact information, is available at the insurance application. Additionally, assumption A5 documents that employees of the insurances gain information about the patient's insurance contracts and their accounting requests from the insurance application. A5 documents the existence of the indirect counterstakeholder insurance employee and also the information this counterstakeholder may have access to.

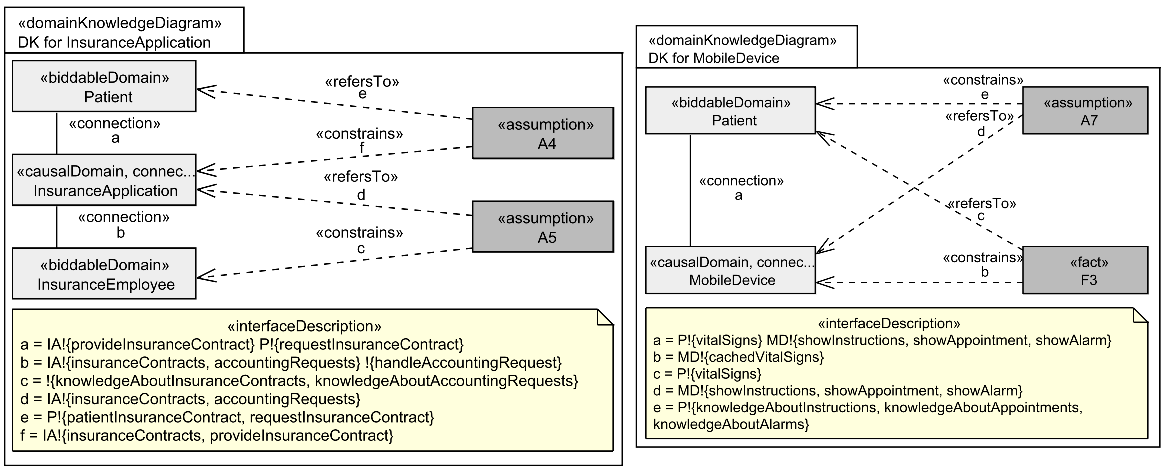

«domainKnowledgeDiagram» DK for InsuranceApplication
«biddableDomain» Patient
«connection» a
«causalDomain, connec... InsuranceApplication
«connection» b
«biddableDomain» InsuranceEmployee
«refersTo» e
«constrains» f
«assumption» A4
«refersTo» d
«constrains» c
«assumption» A5
«interfaceDescription»
a = IA!{provideInsuranceContract} P!{requestInsuranceContract}
b = IA!{insuranceContracts, accountingRequests} !{handleAccountingRequest}
c = !{knowledgeAboutInsuranceContracts, knowledgeAboutAccountingRequests}
d = IA!{insuranceContracts, accountingRequests}
e = P!{patientInsuranceContract, requestInsuranceContract}
f = IA!{insuranceContracts, provideInsuranceContract}
«domainKnowledgeDiagram» DK for MobileDevice
«biddableDomain» Patient
«connection» a
«causalDomain, connec... MobileDevice
«constrains» e
«refersTo» d
«assumption» A7
«refersTo» c
«constrains» b
«fact» F3
«interfaceDescription»
a = P!{vitalSigns} MD!{showInstructions, showAppointment, showAlarm}
b = MD!{cachedVitalSigns}
c = P!{vitalSigns}
d = MD!{showInstructions, showAppointment, showAlarm}
e = P!{knowledgeAboutInstructions, knowledgeAboutAppointments, knowledgeAboutAlarms}

Figure 5. Domain Knowledge about the Insurance Application (**left**) and the Mobile Device (**right**).

Requirement R5 states that instructions, appointments, and alarms shall be sent to patients' mobile devices. What the requirement does not prescribe is that patients recognize the event that their mobile devices received instructions, appointments, and alarms and, e.g., that patients follow the instructions sent to them. Because this is out of the scope of the part of the environment the EHS machine can influence (*cf.* Figure 1). Hence, we document with assumption A7 that we assume that patients recognize and take care about the information sent to their mobile devices. This is another example for an implicit information flow. Fact F3 also documents an implicit information flow, namely that mobile devices will cache the vital signs which are recorded due to requirement R6, in their local memory.

5.2. Graph Generation

In the second sub-step, we automatically generate from the functional requirements and domain knowledge the so-called *detailed stakeholder information flow graphs*. These graphs are used in the following two steps to (1) identify the personal data processed by the system under consideration; (2) how these personal data flow through the system; and (3) at which domains these are available.

As the detailed stakeholder information flow graphs themselves are not relevant for this paper, we omit the details of their structure and generation. The details can be found in [15].

5.3. Identification of Personal Data

In the third sub-step, it is analyzed for every biddable domain in the ProPAn model which personal data of this human being are processed by the system and how these personal data are related to and collected from the owner. This information is documented in so-called *personal information diagrams* (see Figure 6). A personal information diagram is created for each biddable domain of the ProPAn model, and it relates phenomena that represent personal data of the considered biddable domain to that domain. This is realized by a dependency connecting the phenomenon and the domain with the stereotype «relatedTo». The stereotype «relatedTo» provides the possibility to document how the information was collected (attribute collection), from which statements the relation was identified (attribute origin), whether the personal information is sensitive information of the biddable domain (attribute sensitive), and whether the information itself allows one to identify the *single* individuals it is related to, a *group* of individual, or whether it provides no possibility to narrow down the set of individuals it is related to and is hence *anonymous* (attribute linkability). The details on collecting this information and generating the personal information diagrams can be found in [15].

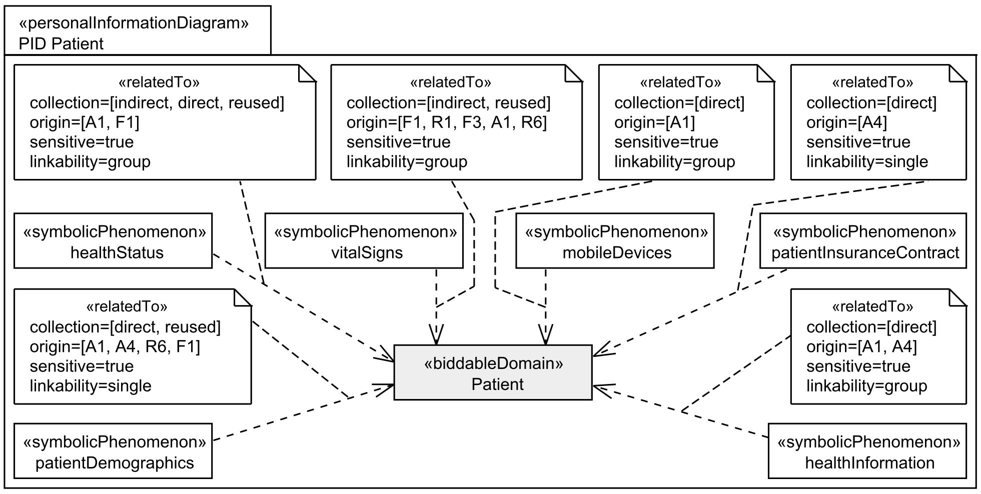


Figure 6. Personal information diagram for the stakeholder patient after the step Identification of Personal Data.

Application to Running Example

Figure 6 shows the personal information diagram for the stakeholder patient that is obtained as a result of the application of the step *Identification of Personal Data* to the EHS scenario. It shows that we identified as personal data for the *Patient* among others the following:

- The *healthStatus* that contains all data that is related to the patient's health and that is processed by the EHS. The health status is considered to be sensitive personal data and it can itself not directly be linked to the single individuals it belongs to, but due to the contained information, it can be linked to a group of individuals it possibly belongs to. The health status is collected in different ways from the patient, it is collected directly from the patient, e.g., during interviews with a doctor, it is indirectly collected by observation of his/her vital signs, and it is also reused from already existing data bases (*cf.* Figure 6).
- The *patientDemographics* summarize details of the patient such as contact information, insurance number, and billing contact. This information suffices to identify the single individual it belongs to, it is considered as sensitive information, and it is collected directly from the patient, e.g., during interviews with a doctor, and by reuse of already existing data sets (*cf.* Figure 6).

Additionally, the personal data *vitalSigns* that, e.g., represent records of patients' pulse and blood pressure, *mobileDevices* that represent information about patients' mobile devices, *patientInsuranceContract* that represent contracts that patients have with their insurances, and *healthInformation* that are used by insurance companies to select the patients' insurance contracts and tariffs are also considered as personal data of patients that are processed by the system-to-be.

5.4. Personal Data Flow Analysis

In the fourth sub-step, the user analyzes how the personal data that were identified in the previous step flow through the system due to the functional requirements and domain knowledge. The documentation of these data flows and the personal data identified in the previous sub-step form the foundation for the automatic generation and validation of privacy requirements.

To document which information is available at which domain, we use so-called *available information diagrams*. For each domain, an available information diagram (similar to a personal information diagram) is created, and for each personal information that is available at the domain, we document the relation between the corresponding symbolic phenomenon and the domain under consideration using a dependency with stereotype «availableAt». Similar to the stereotype «relatedTo», also the stereotype «availableAt» provides some attributes for documentation purposes. These attributes are set during the personal data flow analysis. For details on how the personal data flow analysis is performed and how the available information diagrams are generated see [15]. The attribute duration documents how long a specific information shall be available at the domain. The attribute origin documents from which statements it was identified that the personal data are available at the domain, and the attribute purpose documents because of which statements the personal information has to be available at the domain, *i.e.*, because of these statements the personal data flow to another domain. Figure 7 shows a view on an available information diagram that only contains the before mentioned relations.

In addition to the information which information is available at a domain, an available information diagram also contains the information whether and in which quality the available personal information is linkable to each other at the domain. To document this, we use associations with the stereotype «linkable» between the personal data in the available information diagrams. Again, this stereotype provides some attributes for documentation purposes. The attributes origin, purpose, and duration have the same meaning as for the «availableAt» relation. The additional attribute «linkability» documents with which certainty the data can be linked to each other.

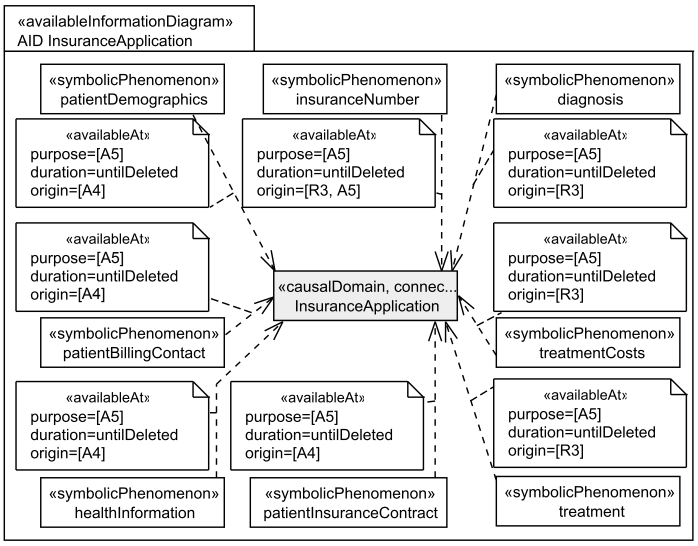

«availableInformationDiagram»
AID InsuranceApplication
«symbolicPhenomenon» patientDemographics
«symbolicPhenomenon» insuranceNumber
«symbolicPhenomenon» diagnosis
«availableAt» purpose=[A5] duration=untilDeleted origin=[A4]
«availableAt» purpose=[A5] duration=untilDeleted origin=[R3, A5]
«availableAt» purpose=[A5] duration=untilDeleted origin=[R3]
«causalDomain, connec... InsuranceApplication
«symbolicPhenomenon» patientBillingContact
«symbolicPhenomenon» treatmentCosts
«symbolicPhenomenon» healthInformation
«symbolicPhenomenon» patientInsuranceContract
«symbolicPhenomenon» treatment

Figure 7. View on the available information diagram for the insurance application showing the which personal data of patients are available at the insurance application.

During the information flow analysis, we elicit two additional relations between the identified personal data. The relation *contains* documents that one personal information contains another personal information. We document this using an aggregation with stereotype «contains». The relation *derivedFrom* documents that a personal information can be derived from one or several other pieces of personal information. We document this using a dependency with stereotype «derivedFrom». The contains and derivedFrom relations are–in contrast to the linkable relation–globally valid relations, *i.e.*, the relations are not only available at specific domains, they are valid at all domains. Figures 8 and 9 present views on a personal information diagram and an available information diagram. These diagrams show how the contains, derivedFrom, and linkable relations are modeled.

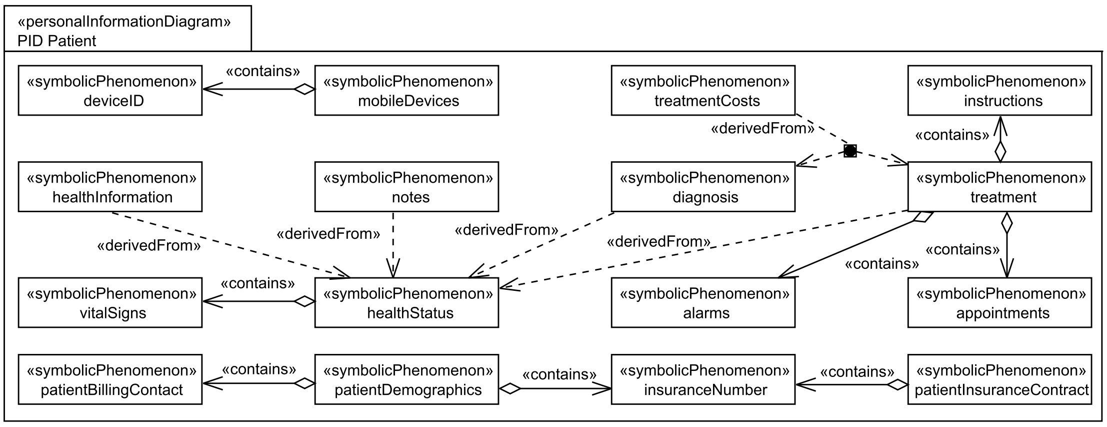

«personalInformationDiagram»
PID Patient
«symbolicPhenomenon» deviceID
«contains»
«symbolicPhenomenon» mobileDevices
«symbolicPhenomenon» treatmentCosts
«symbolicPhenomenon» instructions
«derivedFrom»
«contains»
«symbolicPhenomenon» healthInformation
«symbolicPhenomenon» notes
«symbolicPhenomenon» diagnosis
«symbolicPhenomenon» treatment
«derivedFrom»
«derivedFrom»
«derivedFrom»
«derivedFrom»
«contains»
«contains»
«symbolicPhenomenon» vitalSigns
«contains»
«symbolicPhenomenon» healthStatus
«symbolicPhenomenon» alarms
«symbolicPhenomenon» appointments
«symbolicPhenomenon» patientBillingContact
«contains»
«symbolicPhenomenon» patientDemographics
«contains»
«symbolicPhenomenon» insuranceNumber
«contains»
«symbolicPhenomenon» patientInsuranceContract

Figure 8. View on the final personal information diagram for the patient showing the personal information of the patient and the relations between this personal information.

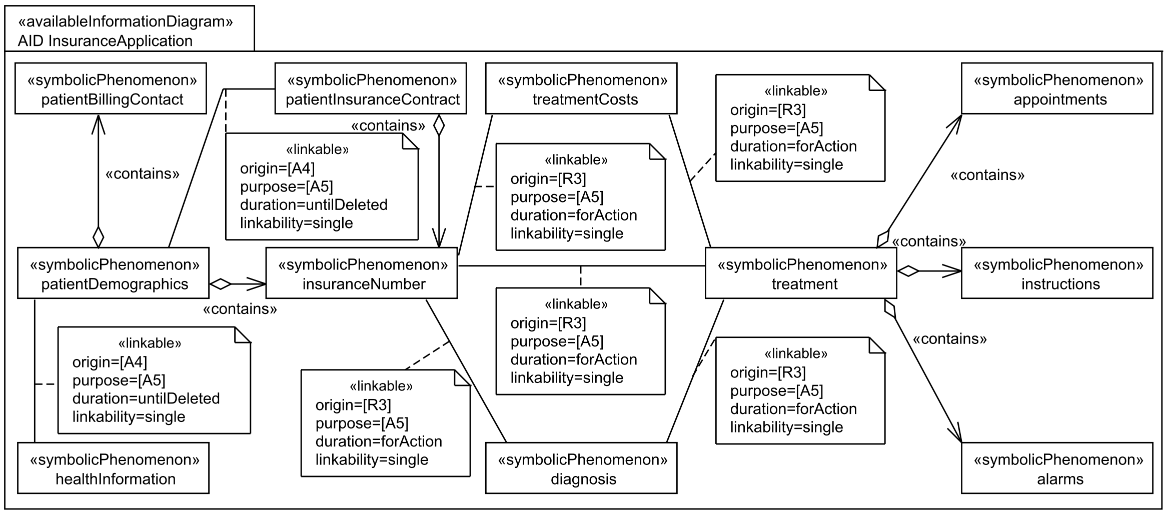

«availableInformationDiagram»
AID InsuranceApplication
«symbolicPhenomenon» patientBillingContact
«symbolicPhenomenon» patientInsuranceContract
«symbolicPhenomenon» treatmentCosts
«symbolicPhenomenon» appointments
«symbolicPhenomenon» patientDemographics
«symbolicPhenomenon» insuranceNumber
«symbolicPhenomenon» treatment
«symbolicPhenomenon» instructions
«symbolicPhenomenon» healthInformation
«symbolicPhenomenon» diagnosis
«symbolicPhenomenon» alarms
«contains»
«linkable» origin=[A4] purpose=[A5] duration=untilDeleted linkability=single
«linkable» origin=[R3] purpose=[A5] duration=forAction linkability=single

Figure 9. View on the available information diagram for the insurance application showing which links between the personal data of the patient are available at the insurance application.

Application to Running Example

Figure 8 shows all personal information that was identified for the patient. During the personal data flow analysis new personal information was identified based on the personal data that were identified during the step Identification of Personal Data (*cf.* Figure 6). All newly identified personal data can be derived from or is contained in the initially identified personal data and additionally, we identified contains and derivedFrom relations among the initially identified personal data. For example, we identified that the *healthInformation* of patients can be derived from the patient's *healthStatus*. Furthermore, it was identified that the *diagnosis*, which doctors create for patients, and the chosen *treatment* are derived from the patient's *healthStatus* by doctors. In addition, from the *diagnosis* and *treatment*, the costs for the performed treatment (*treatmentCosts*) can be derived.

Figure 7 shows which personal information of patients are available at the insurance application. For example, from requirement R3 (attribute origin) it was identified that for accounting, the patient's diagnosis, treatment, treatment costs, and insurance number are sent to the insurance application. This information is kept available there until there are no further legal obligations to keep it and it has to be deleted. It is documented that this information has to be available at the insurance application due to assumption A5 (attribute purpose) to ensure that insurance employees are able to perform the accounting.

Figure 9 shows which relations between the personal data of patients is available at the insurance application. For example, it is (globally) documented that the patient's billing contact and the patient's insurance number are contained in the patient's demographics (*cf.* Figure 8). Additionally, it is documented that from requirement R3 it was identified that for accounting the patient's diagnosis, treatment, treatment costs, and insurance number are linkable to each other at the insurance application. This linkability is obviously needed for the accounting to be able to check whether the treatment and the associated costs are covered by the patient's insurance contract.

6. Generate Privacy Requirements Candidates

To automatically generate privacy requirements candidates, we make use of the artifacts elicited in the previous step and documented in the ProPAn model. We assume that these artifacts reflect the intended information processing that is introduced by the system-to-be and that occur in its environment. This means, we generate the privacy requirements on the assumption that only the intended processing may be performed and the end-users shall be informed about this processing.

In this context, we aim at eliciting all relevant privacy requirements. The user can then decide in the next step of our method (see Section 7) to remove or change the generated privacy requirements if the assumptions made were too strong or too weak.

The ways how we identify the privacy requirements differs for the different kinds of privacy requirements. We consider the six protection goals for privacy engineering unlinkability, transparency, intervenability, confidentiality, integrity, and availability proposed by Hansen *et al.* [5]. Hansen *et al.* state that the protection goal of *unlinkability* includes further properties such as anonymity, pseudonymity, and undetectability. These properties aim at weakening or removing links between data and the persons they belong to. *Transparency* is about increasing the awareness of end-users on how their data is processed and for what purpose by providing them appropriate information about how their data is handled. By realizing the protection goal *intervenability*, end-users shall be able to have control over how and if their data is processed by the system-to-be. In this paper, we consider the security goals confidentiality, integrity, and availability from a privacy perspective and consider these as privacy requirements. This means that the assets to be protected by the security goals are limited to personal data, but the protection of these properties is not only limited to malicious attacks, but also to unintended disclosure, changes, or errors. In this sense, *confidentiality* aims at keeping personal data secret from specific entities. The protection goal *integrity* is concerned with protecting personal data from unwanted changes to keep the data consistent. Addressing the goal *availability* means to ensure that all entities can retrieve the personal data that they are allowed to access at any time. In the following, we describe for each protection goal the privacy requirements that are refinements of the protection goal, and how we automatically identify which of these privacy requirements have to be considered based on the information elicited in the previous step.

6.1. Unlinkability Requirements

To derive the privacy requirements related to the protection goal of unlinkability, we use the terminology introduced by Pfitzmann and Hansen [16]. Pfitzmann and Hansen define the privacy properties anonymity, unlinkability, undetectability, unobservability, and undetectability. Hansen *et al.* [5] summarize all these properties under the protection goal unlinkability. Based on this terminology, we created the UML profile shown in Figure 10. The top-level privacy requirement was originally introduced in [9] and specifies the core of every privacy requirement, namely the stakeholder who shall be protected, the counterstakeholders from whom the stakeholder shall be protected, and the personal data (expressed as phenomena) of the stakeholder that shall be protected. For the protection goal unlinkability, we derived the sub-requirements pseudonymity, unlinkability, and undetectability. The sub-requirement unlinkability is further refined into data unlinkability (requires that certain personal data shall not be linkable to each other) and anonymity (requires that certain personal data shall not be linkable to the corresponding individual). Note that the privacy property unobservability is not represented as a separate requirement, but it can be expressed by instantiating respective anonymity and undetectability requirements (*cf.* [16]). For the automatic generation, we only consider the following requirements that may be refined into pseudonymity requirements in the step *Adjust Privacy Requirements* (Section 7). In Section 7, also the meaning of a pseudonymity requirement is explained.

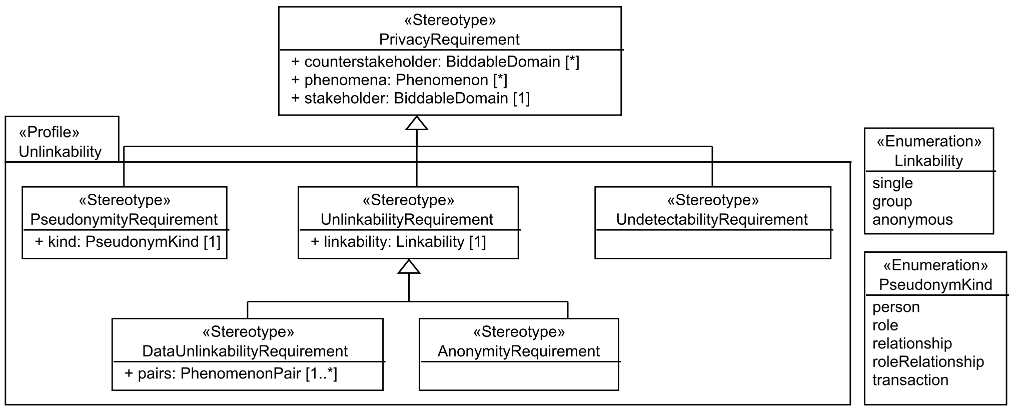


Figure 10. Used Taxonomy of Unlinkability Requirements.

UndetectabilityRequirement Pfitzmann and Hansen define undetectability as: "Undetectability of an item of interest (IOI) from an attacker's perspective means that the attacker cannot sufficiently distinguish whether it exists or not."
AnonymityRequirement Pfitzmann and Hansen define anonymity as: "Anonymity of a subject from an attacker's perspective means that the attacker cannot sufficiently identify the subject within a set of subjects, the anonymity set."
DataUnlinkabilityRequirement Pfitzmann and Hansen define unlinkability as: "Unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker cannot sufficiently distinguish whether these IOIs are related or not." Our data unlinkability requirements express the intended relations between messages, actions, and the like which a subject performs and do not concern the relations of these to the subject itself, as these are represented by anonymity requirements.

6.1.1. Undetectability

Based on the above given definition of Pfitzmann and Hansen, an undetectability requirement of our taxonomy (*cf.* Figure 10) has the following meaning:

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

If a personal information of a stakeholder is not available at a counterstakeholder and also not part of any personal information available at the counterstakeholder, then we assume that this personal information is undetectable for the counterstakeholder. Note that an undetectability requirement may be too strong for this case, because the counterstakeholder may be allowed to know that a specific personal information exists, but may not be allowed to know the content of it. Hence, the user may weaken an undetectability requirement in the next step of our method (Section 7) to a confidentiality requirement.

To keep the number of requirements that are generated small, we create for each pair of stakeholder and counterstakeholder only one undetectability requirement containing all personal information of the stakeholder that shall be undetectable for the counterstakeholder. A personal information p that shall be undetectable for the counterstakeholder c has to be related to the stakeholder s (represented by the existence of a relatedTo relation in s' personal information diagram that relates p to s) and must not be available to c (represented by the absence of an availableAt relation in c's available information diagram that relates p to c).

Application to Running Example

For the sake of simplicity, we only consider the stakeholder patient and the counterstakeholder insurance employee for the generation of the unlinkability requirements.

To the biddable domain insurance employee the same personal information of the patient is available as at the insurance application (*cf.* Figure 7). Hence, an undetectability requirement is generated for the counterstakeholder insurance employee and the stakeholder patient with all personal data of the patient (*cf.* Figure 8) that is *not* available at the insurance employee as value for the attribute phenomena. The undetectability requirement is represented in the first row in Table 1. When we instantiate the above template for the meaning of an undetectability requirement, then we get the following textual representation of it:

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

Table 1. Unlinkability requirements for the stakeholder patient and the counterstakeholder insurance employee.

UnlinkabilityRequirement	**Phenomena / Pairs**
Undetectability	healthStatus, mobileDevices, deviceID, vitalSigns, notes
Anonymity linkability=single	patientBillingContact, patientDemographics, healthInformation, insuranceNumber, patientInsuranceContact, treatmentCosts, treatment, diagnosis, appointments, instructions, alarms
Data Unlinkability linkability=single	(treatmentCosts,diagnosis), (treatmentCosts,patientDemographics), (treatmentCosts,healthInformation), (treatmentCosts,instructions), (treatmentCosts,alarms), (treatmentCosts,insuranceNumber), (treatmentCosts,patientBillingContact), (treatmentCosts,patientInsuranceContract), (treatmentCosts,treatment), (treatmentCosts,appointments),(diagnosis,patientDemographics), (diagnosis,healthInformation), (diagnosis,instructions), (diagnosis,alarms), (diagnosis,insuranceNumber), (diagnosis,patientBillingContact), (diagnosis,patientInsuranceContract), (diagnosis,treatment), (diagnosis,appointments), (patientDemographics,healthInformation), (patientDemographics,instructions), (patientDemographics,alarms), (patientDemographics,insuranceNumber), (patientDemographics,patientBillingContact), (patientDemographics,patientInsuranceContract), (patientDemographics,treatment), (patientDemographics,appointments), (healthInformation,instructions), (healthInformation,alarms), (healthInformation,insuranceNumber), (healthInformation,patientBillingContact), (healthInformation,patientInsuranceContract), (healthInformation,treatment), (healthInformation,appointments), (instructions,alarms), (instructions,insuranceNumber), (instructions,patientBillingContact), (instructions,patientInsuranceContract), (instructions,treatment), (instructions,appointments), (alarms,insuranceNumber), (alarms,patientBillingContact), (alarms,patientInsuranceContract), (alarms,treatment), (alarms,appointments), (insuranceNumber,patientBillingContact), (insuranceNumber,patientInsuranceContract), (insuranceNumber,treatment), (insuranceNumber,appointments), (treatment,appointments), (patientBillingContact,patientInsuranceContract), (patientBillingContact,treatment), (patientBillingContact,appointments), (patientInsuranceContract,treatment), (patientInsuranceContract,appointments)

6.1.2. Data Unlinkability Requirements

Based on the above given definition of Pfitzmann and Hansen, a data unlinkability requirement has the following meaning:

> For each pair of personal information <*pairs*> of the <*stakeholder*>, the <*counterstakeholder*>s shall at most be able to link instances of the two elements of the pair to each other with linkability <*linkability*>.

Note that a data unlinkability requirement with linkability *single* does not constrain the system-to-be to ensure that counterstakeholders are not able to link personal data of the stakeholder to each other, but nevertheless, we make this information explicit to document that this linkability is intended. We generate for each combination of stakeholder and counterstakeholder at most three data unlinkability requirements. Namely, one for each linkability (*single*, *group*, and *anonymous*) with which a counterstakeholder may be able to relate a personal information to the corresponding stakeholder. The information whether two pieces of personal information are linkable to each other by a counterstakeholder can be derived from his/her available information diagram using the linkable relation and the globally defined contains relation (*cf.* Section 5.4).

The linkable relation has a transitive nature. That is, if a personal information a is linkable to a personal information b and b linkable to a personal information c, then this introduces a link between a and c. These transitive relations do not have to be modeled by the user manually, they are automatically computed when needed by the provided tool. During this automatic computation, we consider all contains relations between personal information available at a domain also as linkable relations with linkability *single*. To decide which linkability a derived link between a and c has, we distinguish two cases. The cases are visualized in Figure 11. First (case (a)), if not yet a link between a and c is identified during the closure computation, then the linkability between a and c is the minimum of the existing linkability v between a and b, and the existing linkability w between b and c. Second (case (b)), if there is already a link between a and c, then we replace the existing linkability u only if the minimum of v and w has a greater linkability than u. Minimal and maximal linkability are defined by the total ordering $anonymous < group < single$. For example, if u = anonymous, v = single, and w = group, then $min(v, w)$ = group, and $max(u, min(v, w))$ = group. If no linkable relation is documented between a personal information a and a personal information b that both are available at a domain, then we consider a and b to be linkable to each other with linkability *anonymous* in the closure. That is, it is not known at the domain which personal information a is related to which personal information b.

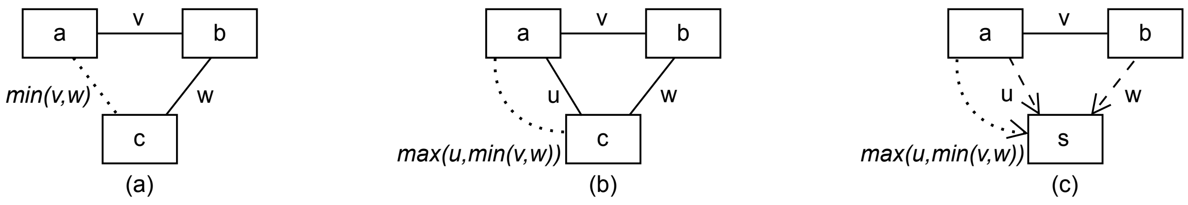


Figure 11. Cases describing how the linkability attribute of the linkable and related to relations are determined for the transitive closure of them. (**a**) Linkable relation exists between a and b, but not between a and c; (**b**) Linkable relations exist between a, b, and c; (**c**) a and b are related to S and a linkable relation between a and b exists.

Based on the computed closure of the linkable relation, the tool decides for each pair of the stakeholder s' personal information that is available at the counterstakeholder c with which linkability it is linkable to each other and to which of the three data unlinkability requirements for s and c it has to be added. In this way, we set the attribute pairs. The attribute phenomena is set to the set of all personal information that is contained in a pair of the attribute pairs of the same requirement.

Application to Running Example

To the biddable domain insurance employee not only the same personal information of the patient is available as at the insurance application, also the same personal information is linkable to each other for the insurance employee with the same linkability. Hence, we can see from Figure 7 that all personal information available to the insurance employee is connected to each other in the transitive closure of the linkable relation with linkability *single*. That is, an insurance employee is able to know which pieces of personal information that are available to him/her belong to each other. Thus, we only obtain one data unlinkability requirement, namely for the linkability *single*, because there is no pair of personal data available at the insurance employee that is not linkable with linkability *single*. This data unlinkability requirement is represented by the third row in Table 1 and contains all 55 pairs of the 11 pieces of personal information available to the insurance employee.

6.1.3. Anonymity Requirement

Based on the above given definition of Pfitzmann and Hansen, an anonymity requirement has the following meaning:

> The <*counterstakeholder*>s shall at most be able to link the personal information <*phenomena*> to the stakeholder with linkability <*linkability*>.

Note that an anonymity requirement with linkability *single* does not constrain the system-to-be to preserve the anonymity of the stakeholders personal data against counterstakeholders, but nevertheless, we make this information explicit to document that this linkability is intended. Similar to the data unlinkability requirements, we also instantiate three anonymity requirements for each pair of stakeholder and counterstakeholder, namely, one for each of the possible linkabilities.

We have now to decide which personal information a can be related to the stakeholder s by the counterstakeholder c with which linkability, to decide to which anonymity requirement's attribute phenomena a has to be added. For this, we use the relatedTo relation from s' personal information diagram and the previously discussed closure of the linkable relation from c's available information diagram. The relatedTo relation (*cf.* Section 5.3) describes for every personal information a of a stakeholder s with which linkability this personal information a can be related to s (by any domain). However, it is possible that we obtain at a domain a linkability of a to s that is greater than the linkability u documented in the relatedTo relation. This case is illustrated in Figure 11c. If a is linkable to a personal information b with linkability v at a domain and b can be related at that domain with linkability w to s, then this implies that at that domain a can be linked to s with at least the linkability $min(v, w)$. Similar as for the closure of the linkable relation, we only take into account the derived linkability $min(v, w)$ if it has a greater linkability than the existing linkability u. The described combination of the linkable and relatedTo relations introduces a new relation *linkableTo* that provides for a stakeholder, a counterstakeholder, and a personal information of the stakeholder that is available to the counterstakeholder the linkability with which the counterstakeholder can relate the personal information to the stakeholder. The linkableTo relation is generated on demand by the tool, and is used to instantiate anonymity requirements.

Application to Running Example

From the generation of the data unlinkability requirements, we know that all personal information of the patient that is available at the insurance employee is linkable to each other with linkability *single*. In addition, from Figure 6, we can see that, e.g., the personal information *patientDemographics* can be related to the single patient it belongs to. Hence, the computed linkableTo relation for the patient and the insurance employee returns for every personal information available at the insurance employee the linkability *single*. That is, for every personal information available to the insurance employee, he/she is able to know to which patient's demographics the personal information belongs. Hence, he/she is able to relate, e.g., the patient's *healthInformation*, which in isolation is only linkable to a group of possible patient's it might belong to, to the patient it belongs to. Thus, we only generate one anonymity requirement for the patient and the insurance employee. This anonymity requirement is shown in the second row of Table 1.

6.2. Transparency and Intervenability Requirements

Transparency and intervenability are protection goals that not yet got as much attention as unlinkability and the classical security goals. In [6], we developed a requirements taxonomy for the protection goal of transparency. In ongoing work, we are developing a similar taxonomy for the protection goal of intervenability. Preliminary results [17] show that we will obtain an analogous structure of intervenability requirements as we identified for transparency requirements. Hence, we assume that we will be able to apply the same strategies for the identification of intervenability requirements as we present for transparency requirements in this section.

The taxonomy of transparency requirements (shown in Figure 12) distinguishes three kinds of requirements. First, presentation requirements that are concerned about *how* the information has to be presented to the stakeholder, second, processing information requirements that inform the stakeholder how his/her data is processed by the system-to-be, and third, exceptional information requirements that state that in the case of exceptional cases, e.g., privacy breaches, the stakeholder and additional authorities have to be informed about these incidents. The first kind of requirements has to be set up manually, as we cannot derive from the output of the first step of our method how stakeholders have to be informed. Also the third kind of requirements cannot be derived from the previously elicited information, but it will be possible to derive them when threats to the unlinkability and security requirements are identified, which is part of our future work. For the second kind of transparency requirements, we show how we can identify about which collection, flow, and storage of personal data stakeholders have to be informed.

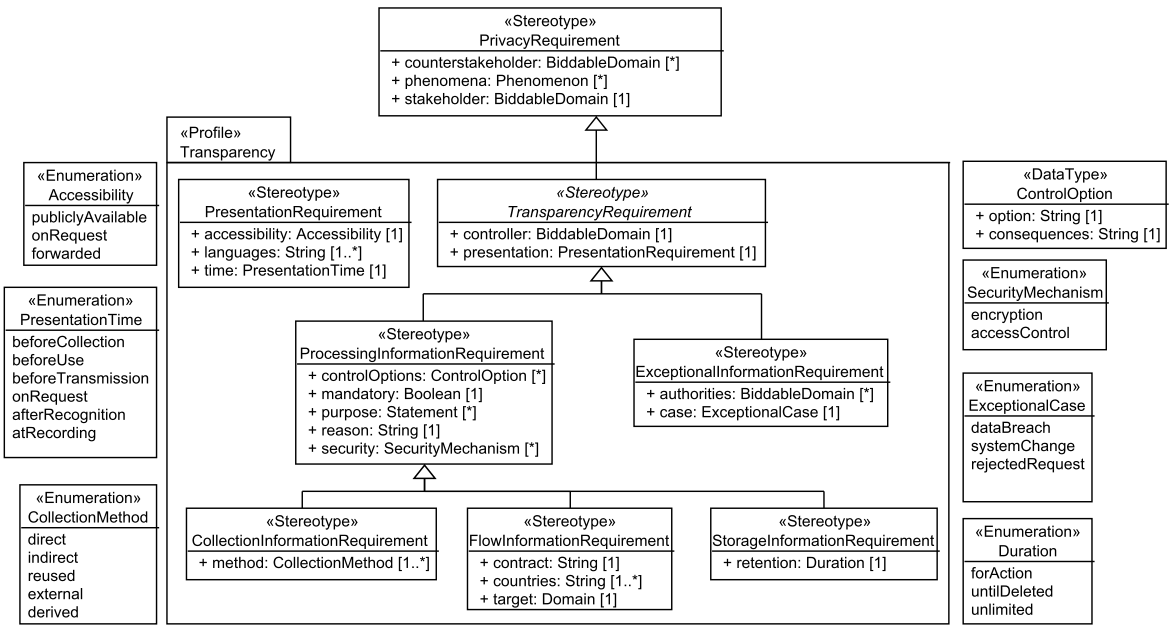

«Stereotype» PrivacyRequirement
+ counterstakeholder: BiddableDomain [*]
+ phenomena: Phenomenon [*]
+ stakeholder: BiddableDomain [1]
«Profile» Transparency
«Enumeration» Accessibility
publiclyAvailable
onRequest
forwarded
«Stereotype» PresentationRequirement
+ accessibility: Accessibility [1]
+ languages: String [1..*]
+ time: PresentationTime [1]
«Stereotype» TransparencyRequirement
+ controller: BiddableDomain [1]
+ presentation: PresentationRequirement [1]
«DataType» ControlOption
+ option: String [1]
+ consequences: String [1]
«Enumeration» PresentationTime
beforeCollection
beforeUse
beforeTransmission
onRequest
afterRecognition
atRecording
«Stereotype» ProcessingInformationRequirement
+ controlOptions: ControlOption [*]
+ mandatory: Boolean [1]
+ purpose: Statement [*]
+ reason: String [1]
+ security: SecurityMechanism [*]
«Stereotype» ExceptionalInformationRequirement
+ authorities: BiddableDomain [*]
+ case: ExceptionalCase [1]
«Enumeration» SecurityMechanism
encryption
accessControl
«Enumeration» ExceptionalCase
dataBreach
systemChange
rejectedRequest
«Enumeration» CollectionMethod
direct
indirect
reused
external
derived
«Stereotype» CollectionInformationRequirement
+ method: CollectionMethod [1..*]
«Stereotype» FlowInformationRequirement
+ contract: String [1]
+ countries: String [1..*]
+ target: Domain [1]
«Stereotype» StorageInformationRequirement
+ retention: Duration [1]
«Enumeration» Duration
forAction
untilDeleted
unlimited

Figure 12. Used Taxonomy of Transparency Requirements.

Our basis to identify these transparency requirements is the data flow implied by the available information diagrams. If a personal information p is available at domain d_2 with origin st and at a domain d_1 with purpose st, then we know that due to statement st the personal information p flows from d_1 to d_2. Figure 13 shows an excerpt of a *stakeholder data flow graph* (SDFG) that visualizes which personal information of a stakeholder flows between which domains. For the sake of readability, the SDFG does not show due to which statements the data flow, but this information is be deduced from the available information diagrams when needed. For details on the generation of this graph see [15].

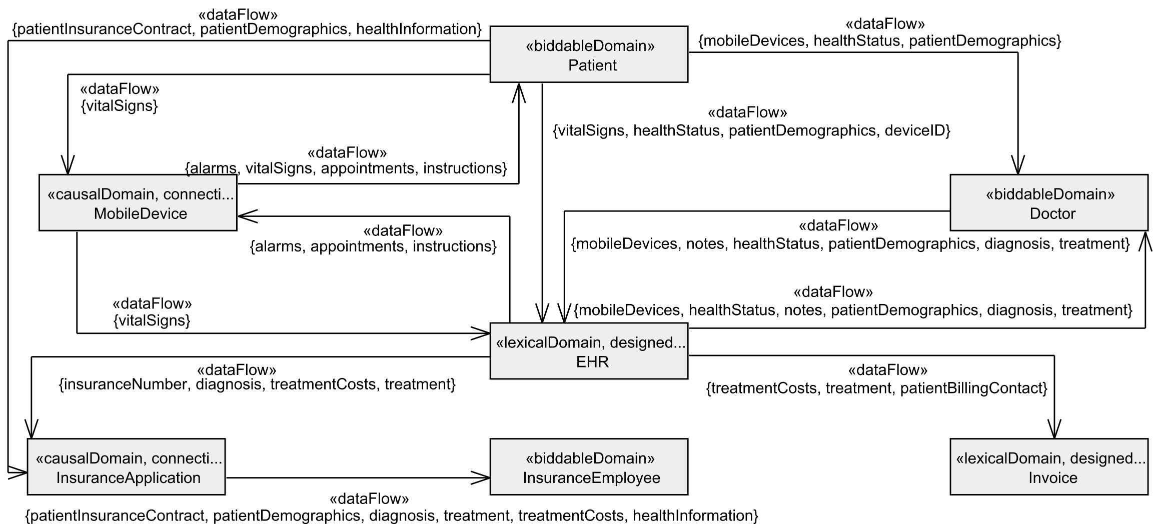

«biddableDomain» Patient
«biddableDomain» Doctor
«causalDomain, connecti... MobileDevice
«lexicalDomain, designed... EHR
«causalDomain, connecti... InsuranceApplication
«biddableDomain» InsuranceEmployee
«lexicalDomain, designed... Invoice
«dataFlow» {patientInsuranceContract, patientDemographics, healthInformation}
«dataFlow» {mobileDevices, healthStatus, patientDemographics}
«dataFlow» {vitalSigns}
«dataFlow» {vitalSigns, healthStatus, patientDemographics, deviceID}
«dataFlow» {alarms, vitalSigns, appointments, instructions}
«dataFlow» {alarms, appointments, instructions}
«dataFlow» {mobileDevices, notes, healthStatus, patientDemographics, diagnosis, treatment}
«dataFlow» {mobileDevices, healthStatus, notes, patientDemographics, diagnosis, treatment}
«dataFlow» {vitalSigns}
«dataFlow» {insuranceNumber, diagnosis, treatmentCosts, treatment}
«dataFlow» {treatmentCosts, treatment, patientBillingContact}
«dataFlow» {patientInsuranceContract, patientDemographics, diagnosis, treatment, treatmentCosts, healthInformation}

Figure 13. Excerpt of the Stakeholder Data Flow Graph for the stakeholder patient.

6.2.1. Collection Information Requirements

A collection information requirement has the following meaning:

> The <*stakeholder*> shall be informed that his/her personal data <*phenomena*> are <*mandatory*>ly collected by the system-to-be that is run by the <*controller*>. The applied collection methods to obtain <*phenomena*> from the <*stakeholder*> are <*method*>. The <*stakeholder*>'s possibilities to control the collection of his/her data are <*controlOptions*>. <*phenomena*> are collected for the purpose of <*purpose*> because <*reason*>. The <*controller*> has selected the security mechanisms <*security*> to protect the personal data <*phenomena*>. The details on how the information has to be presented to the <*stakeholder*> are defined in the presentation requirement <*presentation*>.

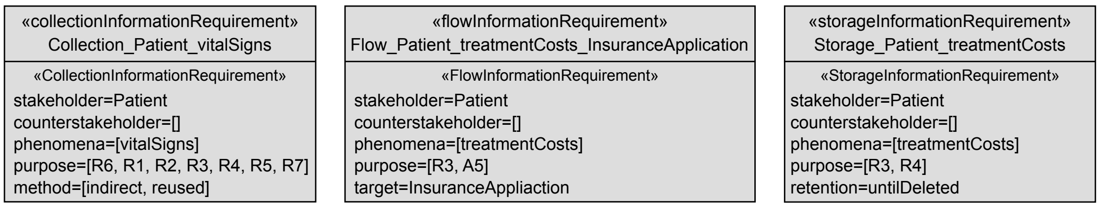
Note that for all processing information requirements the attribute counterstakeholder is not relevant and hence, kept empty.

Data collection happens at those places in the system-to-be where personal information of a stakeholder flows from a given domain to a designed domain, which represent parts of the system-to-be (*cf.* Section 3). This is, because every information that is collected by the system-to-be has to flow from outside the system-to-be (a given domain) to a part of it (a designed domain).

For each personal information p and stakeholder s for whom p represents personal information, we instantiate a collection information requirement with the attributes stakeholder = s and phenomena = $\{p\}$ if p flows from a given domain to a designed domain. Additionally, we can automatically set the attributes purpose and method. As purpose, we set the statements due to which p flows to the designed domain(s) and due to which it (or contained personal information) flows from the designed domain further to other domains. The attribute method is instantiated using the documented collection methods from the relatedTo relations in the personal information diagram of s. All other attributes have to be adjusted by the user in next step of our method.

Application to Running Example

The data that is collected from patients can be derived from the in-going edges of the designed lexical domain EHR in Figure 13. In the EHS scenario, the electronic health records (EHR) are the central point of the system-to-be where all data about patients is collected. For example, the patient's *vitalSigns* are collected by the system-to-be from mobile devices, patients (indirectly through a possible reuse of already existing health records), and doctors (contained in the personal information *healthStatus*). Figure 14 shows the generated collection information requirement for the patient and the personal information vital signs. In Figure 14, only the instantiated attributes are shown, all other attributes have later to be set by the user. The collection methods for the vital signs were already collected during the Identification of Personal Data and documented in the relatedTo relation of the patient's personal information diagram (*cf.* Figure 6). It is documented that the vital signs are indirectly collected from the mobile devices and doctors that measure these, or may originate from already existing health records (reused). The vital signs as part of the health status of the patient are an information in the EHS from which other personal information of the patient is derived, such as diagnoses and treatments (*cf.* Figure 8). Hence, it is automatically derived that the vital signs are collected for the purpose of all functional requirements. Note that we also consider the statements due to which the information is collected (*i.e.*, R1 and R6) as purpose.


«collectionInformationRequirement»
Collection_Patient_vitalSigns
«CollectionInformationRequirement»
stakeholder=Patient
counterstakeholder=[]
phenomena=[vitalSigns]
purpose=[R6, R1, R2, R3, R4, R5, R7]
method=[indirect, reused]

«flowInformationRequirement»
Flow_Patient_treatmentCosts_InsuranceApplication
«FlowInformationRequirement»
stakeholder=Patient
counterstakeholder=[]
phenomena=[treatmentCosts]
purpose=[R3, A5]
target=InsuranceAppliaction

«storageInformationRequirement»
Storage_Patient_treatmentCosts
«StorageInformationRequirement»
stakeholder=Patient
counterstakeholder=[]
phenomena=[treatmentCosts]
purpose=[R3, R4]
retention=untilDeleted

Figure 14. Examples of generated transparency requirements for the EHS scenario.

6.2.2. Flow Information Requirements

A flow information requirement has the following meaning:

> The <*stakeholder*> shall be informed that his/her personal data <*phenomena*> flow <*mandatory*>ly to the <*target*> due to the system-to-be that is run by the <*controller*>. The <*target*> is located in <*countries*> and contractual obligations <*contract*> exist between the <*target*> and the <*controller*>. The <*stakeholder*>'s possibilities to control the flow of his/her data are <*controlOptions*>. <*phenomena*> flow to the <*target*> for the purpose of <*purpose*> because <*reason*>. The <*controller*> has selected the security mechanisms <*security*> to protect the personal data <*phenomena*>. The details on how the information has to be presented to the <*stakeholder*> are defined in the presentation requirement <*presentation*>.

Hence, stakeholders have to be informed about all data flows that are introduced by the system-to-be. Stakeholders do not have to be informed about the information flows inside the system-to-be, because they only have to be informed about the behavior of the system-to-be as a whole. Additionally, stakeholders do not have to be informed about information flows that happen independently of the system-to-be, because these flows are out of the scope of the system-to-be. Hence, we only consider the documented data flows to given domains that are caused by the system-to-be. We distinguish two kinds of data flows introduced by the system-to-be.

First, the system-to-be may introduce flows of personal information from designed domains to given domains. Second, there might be information flows between given domains that originate from a functional requirement. This can be the case if a part of the system-to-be is only responsible to forward personal information p of a stakeholder s from a given domain d_1 to another given domain d_2

without collecting or storing the data itself. We instantiate a flow information requirement for each combination of given domain d_2 and stakeholder s if a piece of personal information p of s is sent to d_2 from a designed domain and/or due to a functional requirement from a given domain.

In both cases of information flows, we generate a flow information requirement with stakeholder = s, phenomena = $\{p\}$, and target = d_2. Additionally, we can automatically set the attribute purpose to the set of statements for which it was documented that the personal information p has to be available at d_2 and due to which statements p flows to d_2. The other attributes have to be set manually by the user in the next step of our method.

Application to Running Example

In the EHS scenario, we have no example for a flow of personal information between two given domains that originates from a functional requirement, but the SDFG for the patient (*cf.* Figure 13) shows two information flows from the designed lexical domain EHR to the given domains insurance application and doctor. For the stakeholder patient, the target insurance application, and the phenomenon treatmentCosts, we generate the flow information requirement shown in Figure 14. The *purpose* for which this information flows from the system-to-be to the insurance application is deduced from the corresponding available information diagram shown in Figure 7. The availableAt relations documented in the available information diagram show that the personal information flows because of the functional requirement R3 and is needed at it due to assumption A5. Hence, both statements are documented as purpose for the flow of personal information.

6.2.3. Storage Information Requirements

A storage information requirement has the following meaning:

> The <*stakeholder*> shall be informed that his/her personal data <*phenomena*> are stored <*mandatory*>ly by the system-to-be that is run by the <*controller*>. The <*phenomena*>'s retention in the system-to-be is <*retention*>. The <*stakeholder*>'s possibilities to control the storage of his/her data are <*controlOptions*>. <*phenomena*> are stored for the purpose of <*purpose*> because <*reason*>. The <*controller*> has selected the security mechanisms <*security*> to protect the personal data <*phenomenon*>. The details on how the information has to be presented to the <*stakeholder*> are defined in the presentation requirement <*presentation*>.

Every personal information that is available at a designed domain, is stored by the system-to-be for at least the time that it is necessary to be there to satisfy the functional requirements. We instantiate for each pair of stakeholder s and personal information p of s a storage information requirement if p is available at a designed domain. For a storage information requirement with stakeholder = s and phenomena = $\{p\}$, we set retention to the maximal duration with which it is available at a designed domain. The maximal duration is determined by the total ordering $forAction < untilDeleted < unlimited$. Additionally, we can automatically derive the purposes for which p is stored by the system-to-be from the available information diagrams.

Application to Running Example

In the EHS scenario, all personal information that is collected by the EHS, is also stored by it. Hence, we obtain for each collection information requirement also a storage information requirement. For example, we get a storage information requirement for the patient and his/her vital signs similar to the collection information requirement shown in Figure 14, but instead of the attribute method it has the attribute retention with value *untilDeleted*. The value *untilDeleted* is selected because the vital signs may be retained longer as needed for the purpose, but they have to be deleted due to some regulations if the patient's health record becomes outdated. Additionally, the EHS stores personal information of the patient, that is derived from the collected personal information, e.g., the *treatmentCosts* are derived from the diagnosis and treatment performed by doctors (*cf.* Figure 8) and stored at the designed lexical domain *Invoice*. For the patient and his/her treatment costs, we generate the storage information requirement shown in Figure 14. The derived purpose for storing the treatment costs are the functional requirements R3 and R4, which are concerned with the accounting and billing of patients. Analogous to the storage information requirement for the patient and the vital signs, the attribute retention is set to *untilDeleted*.

6.3. Security Requirements

In this paper, we consider the basic security requirements confidentiality, integrity, and availability, which we interpret in the context of privacy. Figure 15 shows our UML profile to represent the three security requirements.

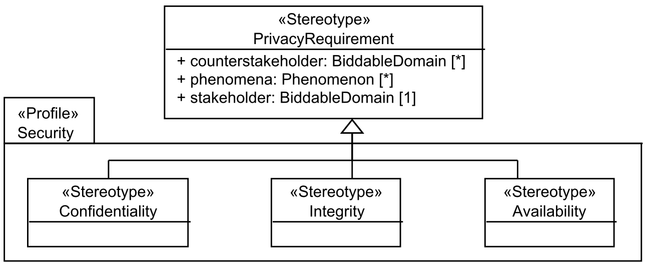


Figure 15. Used Taxonomy of Security Requirements.

A confidentiality requirement has the meaning:

> The personal data <*phenomena*> of the <*stakeholder*> shall be kept confidential from <*counterstakeholder*>.

An integrity requirement has the meaning:

> Random faults of the system and <*counterstakeholder*> shall not be able to negatively influence the consistency and correctness of the personal data <*phenomena*> of the <*stakeholder*>.

An availability requirement has the meaning:

> Random faults of the system and <*counterstakeholder*> shall not be able to negatively influence the availability of the personal data <*phenomena*> to the corresponding <*stakeholder*>.

Note that if the set of counterstakeholders is left empty for an integrity or availability requirement, then the meaning is that the integrity and availability of the stakeholder's personal data shall be preserved against all possible counterstakeholders.

Our above definition of the security requirements also includes a dimension that is classically assigned to safety requirements, namely random faults that may cause harm to the system. For privacy, these random faults are also of relevance, because privacy issues may not only arise from attacks a counterstakeholder might perform or unwanted incidents he/she incidentally causes, but also because of random faults that do not allow access to information or that delete or change the content of the personal data of the stakeholder.

We do not generate confidentiality requirements during this step, but the user may refine an undetectability requirement to a confidentiality requirement as already mentioned above in Section 6.1.1.

The generation of our general integrity and availability requirements is straightforward. We instantiate for each stakeholder whose personal information is available at a designed domain one integrity requirement and one availability requirement with the attribute phenomena set to the set of all of the stakeholder's personal information that is available to the system-to-be, *i.e.*, available at a designed domain, and with an empty set counterstakeholder. During the next step of our method, the user may decide to refine the set of counterstakeholders.

Application to Running Example

For the patient all personal data shown in Figure 8 except *healthInformation* and *patientInsuranceContract* are available at designed domains and hence, we create corresponding availability and integrity requirements.

7. Adjust Privacy Requirements

In this step, the user manually inspects the automatically generated privacy requirements and has the possibility to complete and adjust these. In the following, we discuss the possibilities the user has to complete and adjust the generated requirements based on the different kinds of privacy requirements.

7.1. Unlinkability Requirements

The generated unlinkability requirements do not have to be completed, because all attributes of them are already automatically set. The user only has to decide whether the identified requirements really reflect the privacy protection needs.

7.1.1. Undetectability Requirements

For each undetectability requirement, the user may decide that not all of the personal information of the stakeholder *s* listed in the requirement has to be undetectable for the counterstakeholder *c*. That is, a counterstakeholder *c* may be allowed to know that a specific kind of information exists, but he/she shall not be able to know the exact content of the information. Hence, a user can decide to introduce a confidentiality requirement for *s* and *c* and move the personal information of *s* that only has to be kept confidential and not undetectable from the corresponding undetectability requirement to the new confidentiality requirement. It is possible that the user decides that all personal information shall only be kept confidential. Then the undetectability requirement is completely replaced by the introduced confidentiality requirement.

Application to the Running Example

For the stakeholder patient and the counterstakeholder insurance employee, the undetectability requirement shown in the first row of Table 1 was generated. As it is not possible and needed to hide the information that the EHS processes the patient's health status (including vital signs), mobile devices (including their IDs), and notes about patient's, we decide to relax the undetectability requirement to a confidentiality requirement.

7.1.2. Anonymity and Data Unlinkability Requirements

For each anonymity and data unlinkability requirement, the user has to consider whether the contained personal information or pairs of personal information belong to the correct requirement. That is, the user has to decide whether the linkability of the personal information to the stakeholder or the linkability between the pair of personal information was correctly derived from the ProPAn model. The user can decide to weaken or strengthen the requirements by increasing or reducing the linkability with which a personal information can be linked to the stakeholder or a pair of information can be linked to each other by the counterstakeholder, respectively. The user can even decide that a personal information shall be undetectable or confidential to the counterstakeholder.

Another possibility is that a user refines an anonymity requirement or a part of it to a pseudonymity requirement (*cf.* Table 1). The meaning of a pseudonymity requirement is:

For the personal information <*phenomena*> of the <*stakeholder*> the <*counterstakeholder*> shall only be able to relate it to a <*kind*> pseudonym and not to the <*stakeholder*> himself/herself.

An anonymity requirement with linkability *single* can be translated into a pseudonymity requirement with kind *person* or *role*. The pseudonym kind *person* means that for every individual only one pseudonym exists. Hence, all personal information can be linked to the single individual it belongs to if the relation between pseudonym and individual is known. A *role* pseudonym is used for specific roles. That is, an individual has a single pseudonym for each role. As we consider biddable domains as stakeholders and biddable domains normally represent roles of individuals, a *role* pseudonym allows the same level of linkability as a *person* pseudonym in our situation. This is, because we do not distinguish further roles that a biddable domain may have.

An anonymity requirement with linkability *group* may be translated into a pseudonymity requirement with kind *relationship* or *roleRelationship*. Due to the consideration of biddable domains as stakeholders, *relationship* and *roleRelationship* pseudonyms are also equivalent for our case. The pseudonym kind *relationship* means that for every communication partner another pseudonym is used. For example, only the messages sent to the same partner can be linked to each other using a *relationship* pseudonym and not all communication.

An anonymity requirement with linkability *anonymous* may be translated into a pseudonymity requirement with kind *transaction*. A *transaction* pseudonym is only used once for one action or information that is related to an individual. Hence, the pseudonyms themselves do not provide any link to the individual they belong to.

For more details on the kinds of pseudonyms see [16].

Application to the Running Example

We decide not to change or refine the anonymity and data unlinkability requirements for the patient and the insurance employee shown in the second and third row of Table 1. All this information has to be available to the insurance employee with the possibility to link it to the single patient it belongs to. In addition, the available information has to be linkable to each other with linkability *single*. This is, because insurance employees need all this information to perform the accounting of treatments patients received.

7.2. Transparency Requirements

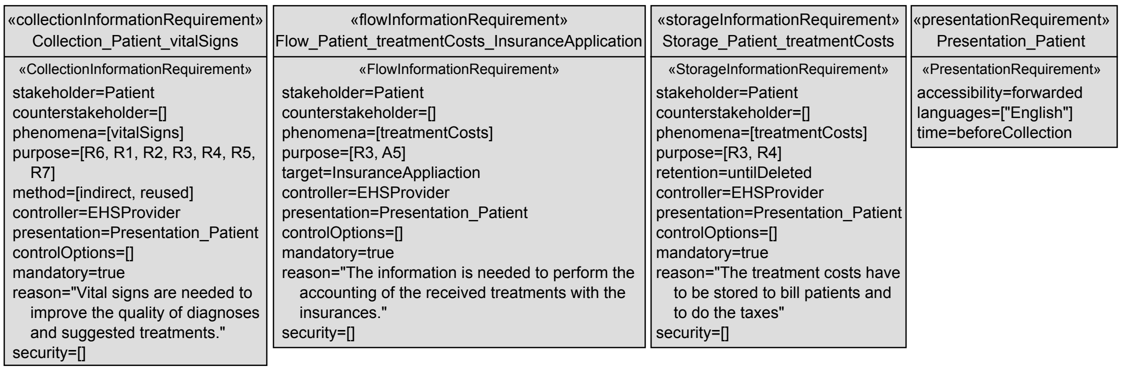
Not all attributes of the transparency requirements were automatically filled during the generation of them. Hence, the user has to complete the generated transparency requirements manually. Normally, we do not expect further modifications on the generated transparency requirements, but the user can add, delete, or merge transparency requirements if needed.

Application to the Running Example

The completed versions of the transparency requirements of Figure 14 are shown in Figure 16. We added to all transparency requirements the same presentation requirement. This presentation requirement states:

The information contained in the related transparency requirements has to be presented *before* the data is collected from the stakeholder in *English* and made accessible to stakeholders by *forwarding* the information to them.

Additionally, all transparency requirements have the same controller that is the provider of the EHS. There are no options to control or intervene into the processing of personal information for the patient, and the processing is mandatory. The attribute security is not set, because yet no concrete security mechanisms where selected, but when such mechanisms have been selected, the attribute should be set. For each transparency requirement, we also provide a reason why the personal information is collected, stored, or flows to another domain.



«collectionInformationRequirement»
Collection_Patient_vitalSigns
«CollectionInformationRequirement»
stakeholder=Patient
counterstakeholder=[]
phenomena=[vitalSigns]
purpose=[R6, R1, R2, R3, R4, R5, R7]
method=[indirect, reused]
controller=EHSProvider
presentation=Presentation_Patient
controlOptions=[]
mandatory=true
reason="Vital signs are needed to improve the quality of diagnoses and suggested treatments."
security=[]

«flowInformationRequirement»
Flow_Patient_treatmentCosts_InsuranceApplication
«FlowInformationRequirement»
stakeholder=Patient
counterstakeholder=[]
phenomena=[treatmentCosts]
purpose=[R3, A5]
target=InsuranceAppliaction
controller=EHSProvider
presentation=Presentation_Patient
controlOptions=[]
mandatory=true
reason="The information is needed to perform the accounting of the received treatments with the insurances."
security=[]

«storageInformationRequirement»
Storage_Patient_treatmentCosts
«StorageInformationRequirement»
stakeholder=Patient
counterstakeholder=[]
phenomena=[treatmentCosts]
purpose=[R3, R4]
retention=untilDeleted
controller=EHSProvider
presentation=Presentation_Patient
controlOptions=[]
mandatory=true
reason="The treatment costs have to be stored to bill patients and to do the taxes"
security=[]

«presentationRequirement»
Presentation_Patient
«PresentationRequirement»
accessibility=forwarded
languages=["English"]
time=beforeCollection

Figure 16. Examples of generated transparency requirements for the eHealth scenario.

7.3. Security Requirements

For the generated availability and integrity requirements, the user may decide for a specific personal information that its integrity does not have to be ensured by the system-to-be, or that it does not have to be made available to the corresponding stakeholder by the system-to-be. Furthermore, the user may limit the availability and integrity requirements to a number of counterstakeholders that shall not be able to negatively influence the availability or integrity of the stakeholders personal data.

Application to the Running Example

We decide not to change or refine the integrity and availability requirements for the patient.

8. Validate Privacy Requirements

In this step, we discuss how the privacy requirements adjusted by the user can be validated. We present validation conditions that indicate inconsistencies between the privacy requirements themselves, or between the privacy requirements and the elicited domain knowledge in the ProPAn model (from which the privacy requirements were originally generated). Our proposed tool is able to detect all inconsistencies discussed in the following and to inform the user about them. The user then has the possibility to correct the inconsistencies. Note that all privacy requirements that are automatically generated by our method do not cause any errors or warnings due to the validation conditions (except the validation conditions that check whether the user completed the attributes of the transparency requirements that cannot be derived from the ProPAn model). An inconsistency between the ProPAn model and the privacy requirements can only be introduced during the manual adjustment of the privacy requirements. Hence, the validation conditions check whether the adjusted privacy requirements are still consistent to each other and to the ProPAn model. We obtained these validation conditions by considering the possible inconsistencies that can be introduced by adjustments of the user between privacy requirements and the ProPAn model.

We distinguish two kinds of validation conditions. First, an adjustment of the user introduces an inconsistency between the privacy requirements or between a privacy requirement and the ProPAn model. In this case, the validation condition raises an error and the user has to remove this inconsistency. Second, an adjustment can be consistent to the ProPAn model, but represent a weaker property than the one that we derived from the ProPAn method during the step *Generate Privacy Requirements Candidates* (*cf.* Section 6). In this case, we present a warning to the user because he/she possibly incidentally weakened the privacy requirements.

8.1. Privacy Requirements

For every privacy requirement, the tool checks whether every phenomenon documented in the requirement is personal information of the privacy requirement's stakeholder. If this is not the case, the user has to correct the privacy requirement or the documented personal information of the stakeholder in the ProPAn model. We formulate this using following validation condition.

VP1 Raise an error for every privacy requirement if the following condition is not satisfied. Every *p* in *phenomena* has to be personal information of the *stakeholder*.

Application to the Running Example

In this paper, we only consider the privacy requirements for patients. Hence, it has to be checked for these privacy requirements whether the attribute phenomena only contains the personal information presented in Figure 8 for validation condition **VP1**. The privacy requirements shown in Table 1 and Figure 14 satisfy this condition and hence, **VP1** does not raise an error for the discussed privacy requirements.

8.2. Unlinkability Requirements

For each combination of stakeholder *s* and counterstakeholder *c*, the corresponding undetectability, anonymity (one for each linkability), and confidentiality requirements have to be consistent to each other. That is, a personal information *p* of the stakeholder *s* may only occur in one of the requirements. Otherwise, we have contradictory requirements. This is checked by the following validation condition.

VU1 Raise an error for every undetectability, anonymity, and confidentiality if the following condition is not satisfied. Every *p* in *phenomena* must not be contained in another undetectability, anonymity (with a different linkability), or confidentiality requirement for the same *stakeholder* and *counterstakeholder*.

In the previous step, the user may have introduced inconsistencies between the unlinkability requirements and the ProPAn model.

In the case that the user strengthened an anonymity requirement (and analogously for data unlinkability requirements), then this introduces an inconsistency to the documented linkability in the ProPAn model. The user may have strengthened an anonymity requirement in one of the following ways:

1. By deciding that the counterstakeholder shall only be able to link the personal information with a weaker linkability to the stakeholder.
2. By changing an anonymity requirement (or a part of it) to an undetectability requirement.
3. By changing an anonymity requirement (or a part of it) to a confidentiality requirement.

Based on the available information diagram of the counterstakeholder, the user has to decide whether his/her adjustments introduce too strong privacy requirements, or whether the functional requirements are not restrictive enough concerning the implied information flows. The following validation conditions check the described inconsistencies.

VU2 For every undetectability and confidentiality requirement do the following. Raise an error if a phenomenon exists in *phenomena* that is available to the *counterstakeholder*.
VU3 For every anonymity requirement do the following. Raise an error if personal information exists in *phenomena* that is linkable to the stakeholder with a greater linkability than *linkability* for the *counterstakeholder*.
VU4 For every data unlinkability requirement do the following. Raise an error if a pair of personal information exists in *pairs* that is linkable to each other with a greater linkability than *linkability* for the *counterstakeholder*.

Weakening unlinkability requirements does not introduce inconsistencies. For example, if counterstakeholder *c* is able to link personal information *p* with linkability *group* to the stakeholder *s* and the related anonymity requirement allows that *c* is able to link *p* to *s* with linkability *single*, then the system-to-be satisfies this anonymity requirement.

If the user deleted an unlinkability or confidentiality requirement that belonged to stakeholder *s* and counterstakeholder *c*, then it might be the case that a personal information *p* (or a pair of personal information) of *s* does not occur in any of the other unlinkability or confidentiality requirements that belong to *s* and *c*. In this case, we have no statement how *p* has to be protected against *c*. This might be intended by the user, but nevertheless, the tool will warn the user about this situation.

VU5 For every combination of stakeholder *s* and counterstakeholder *c*, warn the user when a personal information of *s* exists that does not occur in the *phenomena* of any of the undetectability, anonymity, or confidentiality requirements for *s* and *c*.

VU6 For every combination of stakeholder *s* and counterstakeholder *c*, warn the user when a pair of personal information of *s* exists for which both elements are available at *c*, but that does not occur in the *pairs* of any of the data unlinkability requirements for *s* and *c*.

Application to the Running Example

During the adjustment of the unlinkability requirements, we only decided to turn the undetectability requirement shown in Table 1 into a confidentiality requirement. Hence, all personal information of the patient occurs either in the confidentiality requirement, or in the anonymity requirement with linkability *single* (*cf.* Table 1). Hence, **VU1** does not raise an error for any discussed undetectability, anonymity, and confidentiality requirements. The validation condition **VU2** does not raise an error for our confidentiality requirement that was created from the undetectability requirement shown in Table 1, because the phenomena referenced by the requirement are all not available at the counterstakeholder insurance employee (*cf.* Figure 7). **VU3** and **VU4** do not raise errors, because we did not strengthen the generated anonymity and data unlinkability requirements and hence, the linkability attribute of these requirements is still consistent to the linkability documented in the ProPAn model. The conditions **VU5** and **VU6** do not cause warnings, because all personal data of patients occur in an undetectability, anonymity, or confidentiality requirement for the stakeholder patient and the counterstakeholder insurance employee, and also each pair of personal data of the patient occurs in a respective data unlinkability requirement.

8.3. Transparency Requirements

For the transparency requirements, it has to be checked whether the user completed all generated transparency requirements. The user is informed which transparency requirements have unset attributes and have to be completed.

VT1 Raise an error for every transparency requirement where an attribute is unset.

VT2 For every transparency requirement, warn the user if an attribute with a multiplicity greater than 1 (except counterstakeholder) is empty.

A user might have changed the transparency requirements inconsistently by adding phenomena to transparency requirements that are not collected or stored by the system-to-be, or do not flow to the specified domain. These inconsistencies have to be corrected by the user.

VT3 Raise an error for every collection and storage information requirement if a *p* in *phenomena* exists that is not available at a designed domain.

VT4 Raise an error for every flow information requirement if a *p* in *phenomena* exists that is not available at the *target*.

The user could have deleted transparency requirements or removed phenomena from them. This would lead to an inconsistency between the transparency requirements and the transparency needs that can be derived from the ProPAn model as described in Section 6.2. However, these inconsistencies might be intended by the user, e.g., if there is no need to inform the stakeholder about a specific processing of his/her personal information.

VT5 Warn the user if a personal information flows from a given domain to a designed domain, but no corresponding collection information requirement exists.
VT6 Warn the user if a personal information is available at a designed domain, but no corresponding storage information requirement exists.
VT7 Warn the user if a personal information flows from a designed domain to a given domain, or between two given domains due to a functional requirement, but no corresponding flow information requirement exists.

Application to the Running Example

For the completed transparency requirements shown in Figure 16, the tool notifies us due to validation condition **VT2** that the attributes controlOptions and security are empty, which might indicate that the requirements are incomplete. As we left these attributes empty on purpose, we can ignore this warning. All other validation conditions for transparency requirements are satisfied. **VT1** does not raise an error, because no attribute is unset (*null*). All phenomena contained in the considered collection and storage information requirement are available at a designed domain. Hence, **VT3** does not raise an error. The phenomena contained in the flow information requirement are all available at the insurance application (*cf.* Figure 7) and hence, **VT4** does not raise an error. The validation conditions **VT5**, **VT6**, and **VT7** do not cause any warnings, because there exist no personal information flows from given to designed domains that are not covered by a collection information requirement, there is no personal information available at a designed domain for which no corresponding storage information requirement exists, and there are no personal information flows from designed to given domains, or between given domains due to a functional requirement for which no corresponding flow information requirement exists.

8.4. Security Requirements

Users may remove or add phenomena from availability and integrity requirements during the previous step. To ensure that the user did not incidentally remove a personal information of a stakeholder that is available at a designed domain, we have the following validation condition.

VS1 Warn the user if a personal information of a stakeholder that is available at a designed domain is not contained in the corresponding availability and integrity requirements.

Furthermore, we check whether the user added personal information of a stakeholder to a corresponding availability or integrity requirement that is not available at a designed domain. The system-to-be cannot assure the integrity or availability of personal information that is not processed by it. Hence, we have the following validation condition.

VS2 Raise an error if not all phenomena contained in an availability or integrity requirement are available at a designed domain.

Application to the Running Example

As we did not modify the generated availability and integrity requirements, both validation conditions for security requirements do not cause errors or warnings. **VS1** does not cause an warning because all personal data of patients that are available at a designed domain are contained in the availability and integrity requirements. Furthermore, **VS2** does not raise an error, because only the personal data of patients that is available at a designed domain is contained in the integrity and availability requirements.

The evaluation of all proposed validation conditions can be performed efficiently by the ProPAn tool, because the needed information, e.g., personal data of stakeholder, information flows, designed domains, and given domains, is added to the ProPAn model during the first step of our method. The asymptotic complexity of the evaluation of all validation conditions is limited by $\langle\text{number of phenomena}\rangle^2 \times \langle\text{number of domains}\rangle^2$, because for each validation condition (finitely many), we have to intersect two sets of phenomena and in the worst case, this has to be done for all pairs of domains.

9. Tool Support

In this section, we discuss the tool support that we developed to realize the automatic generation and validation of privacy requirements.

9.1. Technical Realization

We integrated the tool support for our proposed method into the ProPAn tool [10]. The ProPAn tool itself is based on the [18]. The UML4PF tool provides UML profiles that allow users to model Jackson's problem frame models as UML diagrams enriched with stereotypes from the UML profiles. Examples for these diagrams are given in Figures 1, 2 and 5. The ProPAn tool provides additional profiles that make it possible to capture and document privacy-relevant knowledge in UML diagrams. Figures 6–9 and Figure 13 show diagrams that are created using the UML profiles that are already provided by the ProPAn tool. For this paper, we added additional profiles to the ProPAn tool that allow users to represent privacy requirements in UML diagrams. These profiles are shown in Figures 10, 12 and 15. For the transparency requirements, we show in Figures 14 and 16 UML diagrams that use the newly introduced UML profiles.

To edit the UML diagrams, we use Papyrus [19] in the Eclipse IDE [20], but in general every UML tool that creates a UML-conform model can be used.

For the automatic and semi-automatic steps of our method, we use the Epsilon platform [21]. Epsilon offers a variety of languages for, e.g., manipulating, transforming, and validating EMF-based [22] models. Additionally, Epsilon provides a language to specify wizards that can easily be integrated into GMF-based ([23]) editors, such as Papyrus. We use these languages for the (semi-)automatic generation of the ProPAn model and the validation of it.

Hence, the ProPAn tool consists of two parts. First, a collection of UML profiles that allow users to model the needed elements for the method in a single UML model. Second, a collection of Epsilon programs that can be called from the Papyrus user interface using the wizard and validation languages provided by Epsilon.

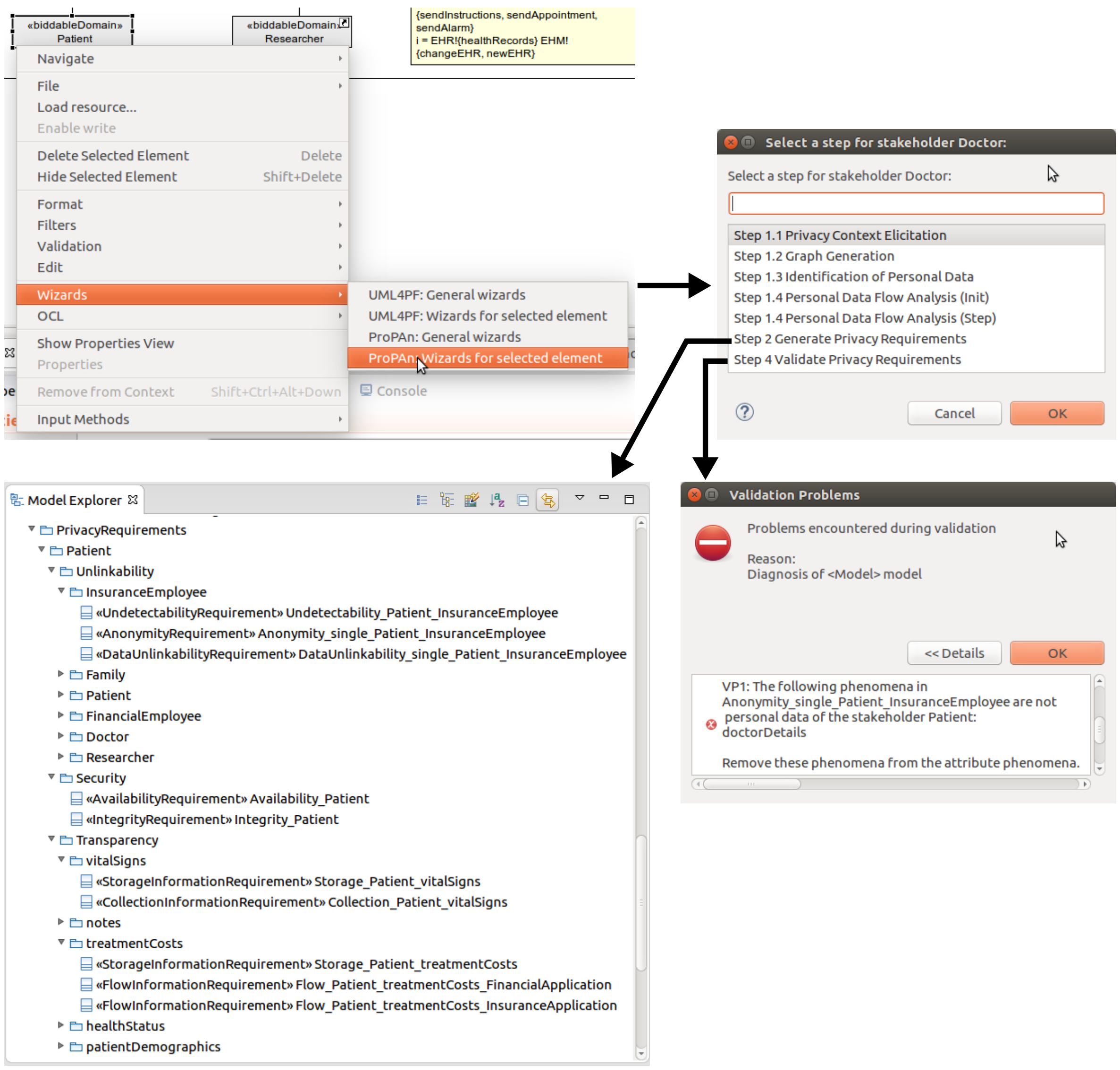
9.2. Using the ProPAn Tool

The UML profiles provided by the ProPAn tool can be used in the same way as all UML profiles using the built-in features of Papyrus. The (semi-)automatic support for the steps of the ProPAn method can be called by a right-click on the biddable domain that shall be considered as a stakeholder and selecting in the context menu the Entry "Wizards->ProPAn: Wizards for selected element" as shown in Figure 17. Then the user can select the step that he/she wants to perform for the selected stakeholder. The steps correspond to the method steps that we introduced in this paper.

From Figure 17, we can see that our tool supports all sub-steps of the first step of our method *Analyze Flow of Personal Data* that was discussed in Section 5. Furthermore, our tool supports the second step of our method, namely the generation of privacy requirements (see Section 6). The sub-steps of the first step of our method are semi-automatic steps that require further user interaction, while the second step of our method is fully automatic and requires no further user interaction.

A part of the automatically generated privacy requirements for patients is shown as a tree view on the UML model in Figure 17. This tree view also shows that we structure the generated privacy requirements using packages to avoid that users get lost in the number of generated privacy requirements. We generate for each stakeholder for whom privacy requirements are generated a package. In this package we add sub-packages for the protection goals unlinkability, security, and transparency. The sub-package for unlinkability contains for each biddable domain that is considered as a counterstakeholder in a privacy requirement belonging to the stakeholder a sub-package that contains all unlinkability related privacy requirements. The sub-package for security contains the stakeholder's confidentiality, integrity, and availability requirements. The transparency sub-package includes for each personal information of the stakeholder for which a transparency requirement exists a sub-package. These sub-packages contain the transparency requirements that are concerned about the respective personal information.

The third step of our method is the adjustment of the automatically generated privacy requirements (see Section 7). The adjustments of the privacy requirements can be performed manually using the standard user interface of Papyrus. Only for turning an undetectability requirement into a confidentiality requirement, we provide a wizard that can be called using a right-click on an undetectability requirement (similar to Figure 17). We added this wizard because attribute values cannot easily be copied from one stereotype instance to another.



{sendInstructions, sendAppointment, sendAlarm}
i = EHR!{healthRecords} EHM!
{changeEHR, newEHR}

Select a step for stakeholder Doctor:

Step 1.1 Privacy Context Elicitation
Step 1.2 Graph Generation
Step 1.3 Identification of Personal Data
Step 1.4 Personal Data Flow Analysis (Init)
Step 1.4 Personal Data Flow Analysis (Step)
Step 2 Generate Privacy Requirements
Step 4 Validate Privacy Requirements

```
▼ PrivacyRequirements
  ▼ Patient
    ▼ Unlinkability
      ▼ InsuranceEmployee
          «UndetectabilityRequirement» Undetectability_Patient_InsuranceEmployee
          «AnonymityRequirement» Anonymity_single_Patient_InsuranceEmployee
          «DataUnlinkabilityRequirement» DataUnlinkability_single_Patient_InsuranceEmployee
      ▸ Family
      ▸ Patient
      ▸ FinancialEmployee
      ▸ Doctor
      ▸ Researcher
    ▼ Security
        «AvailabilityRequirement» Availability_Patient
        «IntegrityRequirement» Integrity_Patient
    ▼ Transparency
      ▼ vitalSigns
          «StorageInformationRequirement» Storage_Patient_vitalSigns
          «CollectionInformationRequirement» Collection_Patient_vitalSigns
      ▸ notes
      ▼ treatmentCosts
          «StorageInformationRequirement» Storage_Patient_treatmentCosts
          «FlowInformationRequirement» Flow_Patient_treatmentCosts_FinancialApplication
          «FlowInformationRequirement» Flow_Patient_treatmentCosts_InsuranceApplication
      ▸ healthStatus
      ▸ patientDemographics
```

Problems encountered during validation

Reason:
Diagnosis of <Model> model

VP1: The following phenomena in Anonymity_single_Patient_InsuranceEmployee are not personal data of the stakeholder Patient: doctorDetails

Remove these phenomena from the attribute phenomena.

Figure 17. Generation and validation of privacy requirements with the ProPAn tool.

The validation of the adjustments of the privacy requirements, which is the fourth and last step of our method (see Section 8), is also supported by the tool support (see Figure 17). After the validation an overview of all errors and warnings detected during it is presented to the user (*cf.* Figure 17). Additionally, the corresponding privacy requirement is annotated with the symbol ✖ in the case that an error was raised for it during the validation or with the symbol ⚠ if a was warning produced for it. In both cases a tooltip provides the information which validation condition was violated and why.

10. Related Work

The LINDDUN-framework proposed by Deng *et al.* [24] is an extension of Microsoft's security analysis framework STRIDE [25]. The basis for the privacy analysis is a data flow diagram (DFD) which is then analyzed on the basis of the high-level threats Linkability, Identifiabilitiy, Non-repudiation, Detectability, information Disclosure, content Unawareness, and policy/consent Noncompliance.

The PriS method introduced by Kalloniatis *et al.* [26] considers privacy requirements as organizational goals. The impact of the privacy requirements on the other organizational goals and their related business processes is analyzed. The authors use privacy process patterns to suggest a set of privacy enhancing technologies (PETs) to realize the privacy requirements.

Liu *et al.* [27] propose a security and privacy requirements analysis based on the goal and agent-based requirements engineering approach i^* [28]. The authors integrate the security and privacy analysis into the elicitation process of i^*. Already elicited actors from i^* are considered as attackers. Additional skills and malicious intent of the attackers are combined with the capabilities and interests of the actors. Then the vulnerabilities implied by the identified attackers and their malicious intentions are investigated in the i^* model.

The above mentioned methods all support the identification of high-level privacy threats or vulnerabilities and the selection of privacy enhancing technologies (PETs) to address the privacy threats or vulnerabilities. These steps are not yet supported by the ProPAn method. But in contrast to a problem frame model, DFDs, goal models, and business processes, as they are used by the above methods, contain not as detailed information as problem frame models. These details help to identify personal data that is processed by the system and how the personal data flow through the system in a systematic way. Hence, our method provides more support for the elicitation of the information that is essential for a valuable privacy analysis than the methods proposed by Deng *et al.*, Kalloniatis *et al.*, and Liu *et al.* In addition, the above mentioned methods do not consider all six protection goals for privacy engineering. Hence, our method provides a more complete consideration of privacy. Additionally, we provide a tool-supported method to systematically identify flows of personal data and the implied privacy requirements. Thus, there exist approaches to model the refinement of privacy and security goals to requirements in the literature, but to the best of our knowledge there is a lack of systematic and automated methods that assist the refinement process. Our proposed method aims at closing this gap.

Omoronyia *et al.* [29] present an adaptive privacy framework. Formal models are used to describe the behavioral and context models, and users' privacy requirements regarding the system. The behavioral and context model are then checked against the privacy requirements using model checking techniques. This approach is complementary to ours, because the knowledge and privacy requirements identified by our method can be used to set up adequate models, which is crucial to obtain valuable results.

Oetzel and Spiekermann [30] describe a methodology to support a complete process for a Privacy Impact Assessment (PIA). Their methodology describes which steps have to be performed in which order to perform a PIA. The results of our work can be used to concretize the steps and to generate the artifacts proposed by Oetzel and Spiekermann.

Antón and Earp [31] derived from around 50 website privacy policies in the e-commerce and health domain a privacy goal and a vulnerability taxonomy. These taxonomies shall help consumers to compare different website privacy policies and to decide which website provides the privacy protection fitting to their needs. All privacy goals considered by Antón and Earp are also reflected by the privacy requirements that we consider. For a more detailed comparison of our work to Antón and Earps's concerning transparency requirements see [6].

Hoepman [32] proposes eight so called privacy design strategies. These privacy design strategies describe fundamental approaches to achieve privacy goals. The considered privacy goals are purpose limitation, data minimisation, data quality, transparency, data subject rights, the right to be forgotten, adequate protection, data portability, data breach notifications, and accountability. The goal data quality is covered in our work by the requirement integrity. The protection goal transparency covers Hoepman's goals transparency and data breach notification. Hoepman's goals data subjects rights, the right to be forgotten, and data portability are covered by the protection goal intervenability. The goal adequate protection is covered by the security goals and the goal unlinkability. The goals purpose limitation, data minimisation, and accountability are directed towards the implementation of specific mechanisms to realize or contribute to the other privacy goals. At this point, we are not interested in selecting possible solutions for the identified privacy goals, but we plan to consider the goals purpose limitation, data minimization, and accountability in a further extension of our method.

11. Conclusions

In this paper, we presented a method to systematically derive high-level privacy requirements for a system-to-be based on its functional requirements. Our proposed method is an extension of the Problem-based Privacy Analysis (ProPAn) method and is integrated into the [10]. We showed how the existing elements of the ProPAn method can be used to elicit (1) the personal data that the system-to-be processes; and (2) the flows of these personal data that the system-to-be introduces. We described how the developed tool generates privacy requirement candidates from this information. For the generation of privacy requirements, we presented refinements of the protection goals for privacy engineering unlinkability, transparency, intervenability, confidentiality, integrity, and availability [5] by means of UML profiles and textual descriptions of each refined requirement. We discussed how a user of our method may adjust and complete the generated privacy requirements. Finally, we presented how our tool is able to validate the consistency of the resulting privacy requirements.

Due to the systematic nature of our method and the high degree of automation, we expect that our method empowers requirements engineers to identify an as complete and coherent set of privacy requirements as possible, especially for complex systems. The results of the application of our method on the electronic health system (EHS) scenario are promising, but we have to further validate our method based on industrial-size case studies and empirical experiments as part of our future work.

The main limitation of our approach to generate privacy requirements is that the completeness and correctness of the generated privacy requirements highly depends on how complete and correct the first step of our method was performed by the user. Otherwise the assumptions on which the generation and validation of the privacy requirements are based (*cf.* Sections 6 and 8) are not valid. That is, if personal data or data flows were not correctly identified and documented, then also incorrect privacy requirements are generated. Hence, the first step of our method should be carried out jointly by a requirements engineer, knowing the functional requirements, a privacy expert, knowing the privacy needs for the system under consideration, and an application domain expert, knowing the application domain of the system under consideration. Another limitation that we have to cope with is that the user of the methods has to be familiar with the problem frames approach and is willing to create problem diagrams for the functional requirements of the system-to-be.

As future work, we want to further extend the ProPAn method to identify privacy threats that could lead to violations of the defined privacy requirements. The elicited information about the flows of personal data will be useful to identify the parts of the system were the identified threats may manifest. Then we want to suggest privacy enhancing technologies (PETs) that can be used to mitigate the identified privacy threats. Finally, we want to elaborate how these PETs have to be integrated into the requirements model in order to mitigate the privacy threats and to satisfy the privacy requirements.

Acknowledgments: This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media".

Author Contributions: Rene Meis wrote the paper and Maritta Heisel provided substantial feedback that improved the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

EHR	Electronic Health Record
EHS	Electronic Health System
PET	Privacy Enhancing Technology
ProPAn	Problem-based Privacy Analysis
SDFG	Stakeholder Data Flow Graph

References

1. European Commission. Proposal for a Regulation of the European Parliament and of the Council on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data (General Data Protection Regulation), 2012. Available online: http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf (accessed on 24 May 2016).
2. Verizon. 2016 Data Breach Investigations Report. Available online: http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/ (accessed on 24 May 2016).
3. Ponemon Institute. 2015 Cost of Data Breach Study: Global Analysis. Available online: http://www-03.ibm.com/security/data-breach/ (accessed on 24 May 2016).
4. GSMA. MOBILE PRIVACY: Consumer Research Insights and Considerations for Policymakers, 2014. Available online: http://www.gsma.com/publicpolicy/wp-content/uploads/2014/02/MOBILE_PRIVACY_Consumer_research_insights_and_considerations_for_policymakers-Final.pdf (accessed on 24 May 2016).
5. Hansen, M.; Jensen, M.; Rost, M. Protection Goals for Privacy Engineering. In Proceedings of the 2015 IEEE Symposium on Security and Privacy Workshops, SPW 2015, San Jose, CA, USA, 21–22 May 2015.
6. Meis, R.; Heisel, M.; Wirtz, R. A Taxonomy of Requirements for the Privacy Goal Transparency. In *Trust, Privacy, and Security in Digital Business*; Springer: Cham, Switzerland, 2015; pp. 195–209.
7. Cavoukian, A. The 7 Foundational Principles, 2011. Available online: https://www.ipc.on.ca/images/resources/7foundationalprinciples.pdf (accessed on 24 May 2016).
8. Jackson, M. *Problem Frames: Analyzing and Structuring Software Development Problems*; Addison-Wesley: Boston, MA, USA, 2001.
9. Beckers, K.; Faßbender, S.; Heisel, M.; Meis, R. *A Problem-based Approach for Computer Aided Privacy Threat Identification*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 1–16.
10. ProPAn: A UML4PF Extension. Available online: http://www.uml4pf.org/ext-propan/index.html (accessed on 24 May 2016).
11. Network of Excellence (NoE) on Engineering Secure Future Internet Software Services and Systems. Available online: http://www.nessos-project.eu (accessed on 24 May 2016).
12. Côté, I.; Hatebur, D.; Heisel, M.; Schmidt, H. UML4PF—A Tool for Problem-Oriented Requirements Analysis. In Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference, Trento, Italy, 29 August–2 September 2011; pp. 349–350.
13. Meis, R. Problem-Based Consideration of Privacy-Relevant Domain Knowledge. In *Privacy and Identity Management for Emerging Services and Technologies*; Springer: Berlin/Heidelberg, Germany, 2014.
14. Beckers, K.; Faßbender, S.; Gritzalis, S.; Heisel, M.; Kalloniatis, C.; Meis, R. Privacy-Aware Cloud Deployment Scenario Selection. In *Trust, Privacy, and Security in Digital Business*; Springer: Cham, Switzerland, 2014; pp. 94–105.

15. Meis, R.; Heisel, M. Supporting Privacy Impact Assessments Using Problem-Based Privacy Analysis. In *Software Technologies*, Proceedings of the 10th International Joint Conference, ICSOFT 2015, Colmar, France, 20–22 July 2015; Lorenz, P., Cardoso, J., Maciaszek, L.A., van Sinderen, M., Eds.; Communications in Computer and Information Science; Springer: Cham, Switzerland, 2016; pp. 79–98.
16. Pfitzmann, A.; Hansen, M. A Terminology for Talking About Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, 2010. Available online: http://www.maroki.de/pub/dphistory/2010_Anon_Terminology_v0.34.pdf (accessed on 23 May 2016).
17. Sabit, S. Consideration of Intervenability Requirements in Software Development. Master's Thesis, University of Duisburg-Essen, Duisburg, Germany, 2015.
18. UML4PF Tool. Available online: http://www.uml4pf.org (accessed on 24 May 2016).
19. Papyrus. Available online: https://eclipse.org/papyrus/ (accessed on 24 May 2016).
20. Eclipse IDE. Available online: http://www.eclipse.org (accessed on 24 May 2016).
21. Epsilon Platform. Available online: http://www.eclipse.org/epsilon (accessed on 24 May 2016).
22. EMF-Based Models. Available online: http://www.eclipse.org/modeling/emf/ (accessed on 24 May 2016).
23. GMF-Based Models. Available online: http://www.eclipse.org/modeling/gmp/ (accessed on 24 May 2016).
24. 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.* **2011**, *16*, 3–32.
25. Howard, M.; Lipner, S. *The Security Development Lifecycle*; Microsoft Press: Redmond, WA, USA, 2006.
26. Kalloniatis, C.; Kavakli, E.; Gritzalis, S. Addressing privacy requirements in system design: The PriS method. *Requir. Eng.* **2008**, *13*, 241–255.
27. Liu, L.; Yu, E.; Mylopoulos, J. Security and Privacy Requirements Analysis within a Social Setting. In Proceedings of the 11th IEEE International Conference on Requirements Engineering, Monterey Bay, CA, USA, 8–12 September 2003; pp. 151–161.
28. Yu, E. Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, Annapolis, MD, USA, 6–10 January 1997.
29. Omoronyia, I.; Cavallaro, L.; Salehie, M.; Pasquale, L.; Nuseibeh, B. Engineering Adaptive Privacy: On the Role of Privacy Awareness Requirements. In Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, 18–26 May 2013.
30. Oetzel, M.; Spiekermann, S. A systematic methodology for privacy impact assessments: A design science approach. *Eur. J. Inf. Syst.* **2014**, *23*, 126–150.
31. Antón, A.I.; Earp, J.B. A requirements taxonomy for reducing Web site privacy vulnerabilities. *Requir. Eng.* **2004**, *9*, 169–185.
32. Hoepman, J. Privacy Design Strategies (Extended Abstract). In Proceedings of the 29th IFIP TC 11 International Conference on ICT Systems Security and Privacy Protection, Marrakech, Morocco, 2–4 June 2014.