Pattern-based Representation of Privacy Enhancing Technologies as Early Aspects

Rene Meis (http://orcid.org/0000-0001-5274-0324) and Maritta Heisel

paluno - The Ruhr Institute for Software Technology,
University of Duisburg-Essen, Duisburg, Germany
{rene.meis, maritta.heisel}@paluno.uni-due.de

Abstract. Several regulations and standards emphasize that privacy shall already be considered from the very beginning in software development. A crucial point during the development of a privacy-friendly software is the selection and integration of measures that implement specific privacy requirements or mitigate threats to these. These measures are called privacy enhancing technologies (PETs). PETs have a cross-cutting nature. That is, a PET needs often to be integrated into several base functionalities of the software-to-be. For example, anonymization techniques need to be integrated into functionalities that shall reveal originally identifiable information in an anonymized form to others. One possibility to handle cross-cutting concerns already on the requirements level is aspect-oriented requirements engineering. In this paper, we show how PETs can be represented as early aspects and how these can be integrated into a given requirements model in problem frames notation. Furthermore, we show how PETs can be represented as patterns to help requirements engineers to identify and select appropriate PETs that address the privacy requirements they have to satisfy. We use the PET Privacy-ABCs (Attribute-Based Credentials) to illustrate our approach.

1 Introduction

Regulations, such as the EU General Data Protection Regulation [1], and industrial standards, such as ISO 29100 [2], emphasize that privacy shall already be considered from the very beginning in software development. To realize the privacy requirements of the software-to-be, privacy enhancing technologies (PETs) may be used at different stages. First, the selection of a PET can emerge from the given requirements. For example, it could be an initial requirement that an anonymous authentication scheme shall be used to ensure the authenticity and correctness of personal information (PI) provided by data subjects (DS), e.g., end-users, without revealing too much information to the software-to-be and consequently its controller. Second, during a privacy risk analysis it can become apparent that the integration of PETs is necessary to mitigate unacceptable privacy risks. For example, it could be identified that specific data needs first to be anonymized before it is transmitted, or that a mechanism needs to be integrated to inform end-users about the controller's privacy policy.

In both situations, requirements engineers face the following questions. (1) How to find out whether and which PETs exist with the needed properties? (2) How to select from a set of PETs addressing a privacy requirement the most appropriate for the system-to-be? (3) How to align the selected PET with the other requirements? That is, the selected PET needs to be integrated into one or more other functional requirements to satisfy the desired privacy requirement.

In this work, we propose a pattern-based representation of PETs that aims at assisting in answering questions (1) and (2) by providing a common structure to describe PETs. This structure shall help requirements engineers to assess whether a PET can be integrated into their software system (question (1)) and to compare the benefits and liabilities of different PETs to select the best-fitting PET (question (2)). To support question (3), we propose the consideration of PETs as early aspects. PETs (or parts of it) describe *cross-cutting functionality* that is integrated into the *base functionality* of the software-to-be to ensure certain privacy properties, e.g., anonymity and transparency, or to mitigate specific privacy threats, e.g., eavesdropping and unawareness. The cross-cutting functionalities are also called *aspects* in aspect-oriented requirements engineering. Aspects are described independently from the base functionality they shall be integrated into. Additionally, it needs to be described how an aspect is integrated into the base functionality. This integration is also called *weaving*.

The rest of the paper is structured as follows. Section 2 provides the background of this paper. The pattern format to represent PETs is introduced in Section 3. An example instantiation of the format for the PET Privacy-ABCs follows in Section 4. We discuss the contribution of this paper in Section 5. Related work is presented in Section 6 and Section 7 concludes the paper.

2 Background

In this paper, we use context and problem diagrams to illustrate the software systems a PET shall be integrated into and the PET itself. Jackson [3] introduced these diagrams as part of his problem frames notation. A context diagram shows the software system consisting of the software-to-be, called *machine* (represented by the symbol ◫), in its environment, which consists of *domains*. A domain can either be biddable $\boxed{B}$ (a human), causal $\boxed{C}$ (a technical device with a predictable behavior), or lexical $\boxed{X}$ (a physical representation of data). The domains and machines are connected by *interfaces* that consist of *phenomena* (e.g., events, actions, operations, and data) the domains and machines share with each other. A phenomenon is always controlled by exactly one of the connected domains and machines and observed by the others. A problem diagram shows a part of the context diagram that is responsible to address a certain *functional requirement*. Hence, it consists of the machine that is responsible to satisfy the requirement, the domains relevant for the requirement, the interfaces between them, and the requirement itself with its references to the domains. These references can either just *refer to* phenomena of a domain, or *constrain* a domain to cause specific phenomena or a modification of its phenomena's states.

Figure 2 shows a simple problem diagram. It consists of the machine Base Machine, the biddable domain User, and the domain Resource depicted as rectangles. Note that the type of the domain Resource is left open, i.e., it can be of an arbitrary type. The problem diagram also contains two interfaces. One connects the User with the Base Machine and one connects the Base Machine with the Resource. At the interface between the User and the Base Machine, the User is able to issue the phenomenon requestResource. This is expressed using the notation U!, where U is the abbreviation for the domain User. Furthermore, the problem diagram contains the requirement Provide Service depicted as dashed oval. This requirement refers to the request of the User (dashed line without arrowhead) and constrains the Resource to provide its service (dashed line with arrowhead).

We extended Jackson's notation in previous work [4] to model cross-cutting functional requirements (called *aspects*). For this purpose, we introduce the concept of *join points*. A join point is a placeholder for a domain or machine of a *base problem* (i.e., a not cross-cutting functional requirement) the aspect shall be integrated into. This allows to describe the aspect independently from a concrete base problem it shall be integrated into. For the integration of the aspect, the join points are instantiated. In this paper, we represent join points as rectangles with gray background and "normal" domains as rectangles with white background. Figure 5 shows an aspect diagram with the Verifier Machine, Presentation Policy, and User Agent as "normal" domains, the Base Machine, Resource, and User as join points, and the cross-cutting requirement Request Policy which states that if a User requests access to the Resource, the User Agent first requests the Presentation Policy. In consequence, the Base Machine shall request the Presentation Policy from the Verification Machine and provide it to the User Agent.

In addition to the structural view provided by the above-mentioned diagrams, we use UML sequence diagrams to provide a behavioral view on the base problems, aspects, and their integration, which is also called *weaving*. In the sequence diagrams, we also highlight the elements related to join points with gray background (cf. Figure 6). How context, problem, aspect, and sequence diagrams are created is out of the scope of this paper. Details on this can be found in [3,4].

3 Pattern Format for PET Patterns

Table 1 shows the pattern format that we propose to represent PETs. The audience of the PET patterns are requirements engineers that want to identify, select, and integrate PETs that address certain privacy requirement they have to consider. The patterns themselves can be created by anyone who is familiar with the respective PET and the aspect-oriented problem frames notation (see Section 2). The pattern format is based on the suggestions of Harrison [5].

The pattern sections **Motivation**, **Context**, **Problem**, **Privacy Forces**, and **General Forces** are concerned with the kind of software system the PET can be integrated into. These pattern sections can be used by requirements engineers to assess whether they can use the PET or not. We propose to describe the **Context** using a high-level context diagram that contains the machines,

Table 1. The pattern format for PET patterns

1 Name	All known names of the PET
2 Motivation	Example scenarios that show the necessity of the PET
3 Context	Description of the software systems the PET can be integrated into.
4 Problem	Description of the software system's base problems with privacy requirements the PET shall address.
5 Privacy Forces	Privacy requirements in the Problem the PET addresses
5.1 Confidentiality	PI shall be kept secret
5.2 Integrity	PI shall be correct and up-to-date
5.3 Availability	PI shall be accessable
5.4 Anonymity	PI shall not be linkable to the data subject (DS)
5.5 Data Unlinkability	PI shall not be linkable to each other
5.6 Undetectability	Existence or occurrence of PI shall not be recognizable
5.7 Pseudonymity	Only pseudonyms shall be linkable to the PI
5.8 Collection Information	DS shall be informed about data collection
5.9 Storage Information	DS shall be informed about storage procedures
5.10 Flow Information	DS shall be informed about flows of PI to others
5.11 Exceptional information	DS and authorities shall be informed about breaches
5.12 Data subject intervention	DS shall be able to intervene into the processing
5.13 Authority Intervention	Authorities shall be able to intervene into the processing
6 General Forces	Other issues making it difficult to address the Problem
6.1 End-user friendliness	The PET's influence on the user experience
6.2 Performance	The PET's impact on the system's performance
6.3 Costs	Costs and effort to be spent emerging from the PET
6.4 Impact on functionality	Potential effects of the PET on the software system
6.5 Abuse of PET	Unintended usage of the PET
6.6 Revocation	Possibilities to abolish privacy properties of the PET
7 Solution	Description of the PET and how it can be integrated into base problems fitting to the Context and Problem
7.1 General Overview	Overview of the domains involved in the PET
7.2 Assumptions	Assumptions on which the PET relies
7.3 Aspects	Description of the PET's cross-cutting functionality
7.4 Weaving	Explains how Aspects are integrated into the Problem
7.5 Base Problems	Not cross-cutting functionality introduced by the PET
8 Design Issues	Discussion of specific design and implementation details
9 Privacy Benefits	The PET's positive consequences on the Privacy Forces
10 General Benefits	The PET's positive consequences on the General Forces
11 Privacy Liabilities	The PET's negative consequences on the Privacy Forces
12 General Liabilities	The PET's negative consequences on the General Forces
13 Examples	Applications of the PET (e.g., on the Motivation)
14 Related Patterns	A list of patterns describing related PETs

domains, and interfaces that such a software system typically has as join points. The base problems in section **Problem** shall be presented as high-level problem diagrams using the machines, domains, and interfaces of the context diagram provided in the **Context**. Additionally, a behavioral view on the base problems shall be specified for the later description of the *Weaving*.

We suggest to describe the *forces* that make it difficult to address the existing privacy requirements of the software system in two pattern sections. First, the **Privacy Forces** shall be documented, i.e., the privacy requirements that the PET shall address. We propose to consider the privacy requirements that we also use in the Problem-based Privacy Analysis (ProPAn) method [6,7] (cf. Table 1), but the list can also be extended with further privacy requirements if appropriate. Second, **General Forces** shall be discussed. We identified six generic general forces that should be considered (cf. Table 1), but others may be added.

The PET itself and its consequences are considered in the pattern sections **Solution**, **Design Issues**, **Privacy Benefits**, **General Benefits**, **Privacy Liabilities**, **General Liabilities**, and **Examples**. The **Solution** is structured into five subsections. First, it contains a *General Overview* of the domains involved in the PET (including the join points of the **Context**) and the interfaces between them in the form of a context diagram. It is also important to document the *Assumptions* on which the functionality of the PET and its proposed privacy-enhancing properties rely. The PET's cross-cutting functionality shall be described as *Aspects* providing both a structural and a behavioral view. The *Weaving* explains how the aspects can be integrated into the base problems that are described in the pattern section **Problem**. The weaving shall combine the behavioral views of the base problems and the PET's aspects. Finally, a PET can introduce additional *Base Problems*, i.e., functionality that is not cross-cutting. These base problems are mostly concerned with the configuration of the PET.

To describe the consequences of a PET, we distinguish, as usual, between positive consequences, called *benefits*, and negative consequences, called *liabilities*. We further differentiate between **Privacy Benefits/Liabilities** and **General Benefits/Liabilities**, as we also did for the forces. The documented consequences shall help requirements engineers to compare different PETs that fit to their software system with each other and to finally select a PET.

4 A PET Pattern for Privacy-ABCs

In this section, we show how the pattern format described in the previous section can be used to present the PET Privacy-ABCs (Attribute-Based Credentials). As source for this PET pattern, we took the description [8] from the ABC4Trust project. The rest of this section shows the PET Pattern for Privacy-ABCs.

1 Name Privacy-ABCs, Attribute-Based Credentials

2 Motivation A cigarette vending machine shall only provide cigarettes to adults. Hence, the machine has to check whether a customer is adult before cigarettes are provided to him or her. The cigarette vending machine shall not

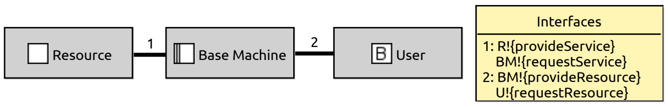

Fig. 1. Context of Privacy-ABCs

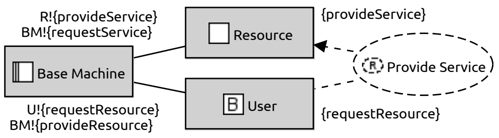

Fig. 2. Basic structure of base problems addressed by Privacy-ABCs

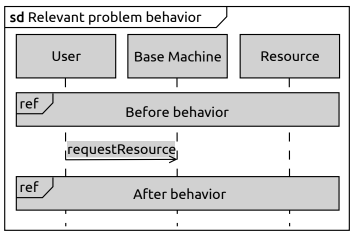

Fig. 3. Behavior of base problems addressed by Privacy-ABCs

be able to gain more information about the customer or to learn that a certain customer already purchased cigarettes from it.

3 Context A software shall be developed that processes personal information (PI) of its users in order to provide a service using an additional resource. It shall be ensured that certain PI provided by the user is correct and authentic. That is, users shall not be able to input incorrect data about them. Figure 1 shows a context diagram that consists of the core elements of systems the PET shall be integrated into. The context diagram shows the User that can request the service of the Resource from the Base Machine that manages this Resource.

4 Problem A mechanism is needed to prove that a user's PI has a certain property or to provide parts of the PI while as little PI as necessary is revealed to the software. Figure 2 shows the kind of base problems Privacy-ABCs might be integrated into. The problems have in common that a User requests a Resource's service from a Base Machine. The Base Machine processes the request and shall only provide under specific circumstances the requested service of the Resource to the User. Figure 3 shows the relevant behavior of the base problems. The behavior consists of an arbitrary Before behavior, the request of the User and an arbitrary After behavior. The request is the relevant behavior because the Base Machine shall only execute the After behavior if the information provided with the request is authentic, correct, and satisfies certain properties, e.g., it contains a proof that the user's age is above 18.

5 Privacy Forces

5.1 Confidentiality: Only partial PI or the proof that the PI satisfies a certain property is needed. The actual PI shall not be disclosed at all.

5.2 Integrity: The provided information shall be authentic and correct.

5.4 Anonymity: The service provider shall not be able to link the data collected during an interaction with the user to him or her.

5.5 Data unlinkability: The service provider shall not be able to link the data

collected during an interaction with the user to the data collected during other interactions of him or her.
5.7 Pseudonymity: Transaction pseudonyms are needed. That is, for each interaction with a user, a new pseudonym is created that is neither linkable to the user nor to other actions of him or her (for details see [9]).
5.8 Collection information: Users shall be informed about the PI to be collected.

6 General Forces
6.1 End-user friendliness: The mechanism to check the authenticity and correctness of the user's request and the provided data shall not introduce much inappropriate effort that needs to be spent by users in comparison to the sensitivity of the PI that is needed to provide the requested service.
6.2 Performance: The mechanism to check the authenticity and correctness of the user's request and his or her data shall not unnecessarily reduce the response time of the software-to-be or slow down the overall software system.
6.3 Costs: The costs, also in the sense of effort, to implement, integrate, deploy, and maintain the PET shall be appropriate in comparison to its benefits.
6.4 Impact on software system's functionalities: The integration of a solution into the base problems shall not negatively influence other system functionality.
6.5 Abuse of PET: It shall not be possible to get access to the service by providing incorrect data.
6.6 Revocation: In certain situations, e.g., abuse of the service, it may be wished to be able to re-identify the individual user that performed certain actions that led to that certain situation.

7 Solution
7.1 General Overview: Figure 4 shows the context diagram for a basic Privacy-ABCs system derived from [8]. The gray domains originate from the base problem Privacy-ABCs shall be integrated into and the white domains are introduced by Privacy-ABCs. The machine that needs to be built is the Verifier Machine. This machine is operated by the Verifier (service provider) who is able to manage the Presentation Policy. The Presentation Policy describes which information a User has to disclose in order to get access to the Resource. To create a Presentation Policy, the Credential Specification and Issuer Parameters provided by an Issuer are used. The Issuer's task is to provide Credentials to Users and to ensure that these Credentials contain only valid information about the respective User. Which information can be stored in a Credential is defined in the Credential Specification. The Issuer Parameters specify how Presentation Tokens generated from User's Credentials can be verified to satisfy or to not satisfy certain properties. To generate Credentials, the Issuer uses his or her Issuance Key. The Verifier Machine represents the software part of Privacy-ABCs that needs to be integrated into the software-to-be. It receives requests from the Base Machine to provide the Presentation Policy and to check whether a User is allowed to access the Resource by verifying a provided Presentation Token using the Presentation Policy and the Credential Specification. The Verifier Machine may store these Used Presentation Tokens. Instead of receiving the requests directly from the User, the Base Machine receives the User's request from his or her User Agent. The User Agent manages

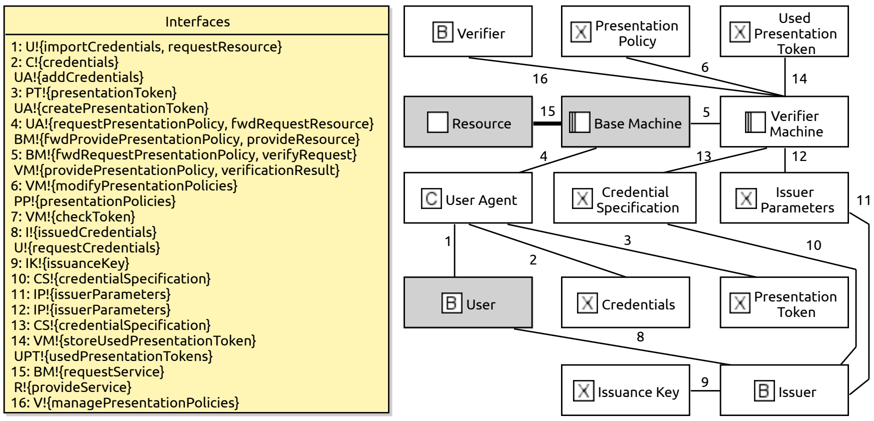
Interfaces
1: U!{importCredentials, requestResource}
2: C!{credentials}
UA!{addCredentials}
3: PT!{presentationToken}
UA!{createPresentationToken}
4: UA!{requestPresentationPolicy, fwdRequestResource}
BM!{fwdProvidePresentationPolicy, provideResource}
5: BM!{fwdRequestPresentationPolicy, verifyRequest}
VM!{providePresentationPolicy, verificationResult}
6: VM!{modifyPresentationPolicies}
PP!{presentationPolicies}
7: VM!{checkToken}
8: I!{issuedCredentials}
U!{requestCredentials}
9: IK!{issuanceKey}
10: CS!{credentialSpecification}
11: IP!{issuerParameters}
12: IP!{issuerParameters}
13: CS!{credentialSpecification}
14: VM!{storeUsedPresentationToken}
UPT!{usedPresentationTokens}
15: BM!{requestService}
R!{provideService}
16: V!{managePresentationPolicies}
B Verifier
X Presentation Policy
X Used Presentation Token
Resource
Base Machine
Verifier Machine
C User Agent
X Credential Specification
X Issuer Parameters
B User
X Credentials
X Presentation Token
X Issuance Key
B Issuer
1
2
3
4
5
6
8
9
10
11
12
13
14
15
16

Fig. 4. Context diagram of Privacy-ABCs

the User's Credentials and generates on demand Presentation Tokens based on a Presentation Policy and the Credentials to request access to a Resource. A User can request Credentials from an Issuer and import these to his or her User Agent.
7.2 Assumptions: We have to consider some assumptions for the issuing of credentials. The Issuer shall create only authentic and correct Credentials for Users using the Credential Specification and Issuance Key. The Issuer needs to be trusted by the User and the Verifier. Furthermore, we assume that the User will add the Credentials provided by the Issuer to his or her User Agent and is not able to modify them. For the generation of Presentation Tokens, we have to assume that a User's User Agent is able to properly generate Presentation Tokens for the Resource the User requests based on the User's Credentials and the Verifier's Presentation Policy. Furthermore, we have to assume that User Agent and Base Machine use an anonymous communication channel, e.g., using Tor[1]. Otherwise, it could be possible for the Base Machine to use meta-data, e.g., the User's IP address, to link Presentation Tokens to each other.
7.3 Aspects: Privacy-ABCs contain three aspects that need to be integrated into base problems which are concerned with requests of a User to a Resource that shall be protected. (1) The Presentation Policy that specifies the information a Presentation Token shall contain to get access to the requested Resource needs to be provided to the User Agent. The aspect diagram for this cross-cutting concern is shown in Figure 5. The behavioral view to address the aspect is shown in Figure 6. The sequence diagram shows that if the User requests a resource via his or her User Agent, the User Agent first requests the Presentation Policy from the Base Machine. The Base Machine forwards the request to the Verifier Machine that retrieves the Presentation Policy and provides it to the Base Machine. Finally, the Base Machine provides the Presentation Policy to the User Agent, which then consequently received the presentation policy. (2) The Presentation Token

[1]`https://www.torproject.org/` Accessed 21 Mar 2017

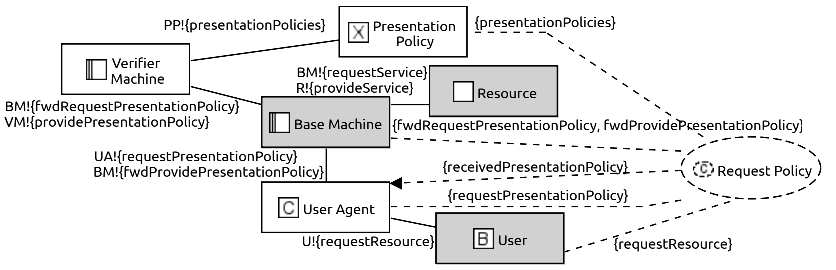
PP!{presentationPolicies}
Presentation Policy
{presentationPolicies}
Verifier Machine
BM!{requestService}
R!{provideService}
Resource
BM!{fwdRequestPresentationPolicy}
VM!{providePresentationPolicy}
Base Machine
{fwdRequestPresentationPolicy, fwdProvidePresentationPolicy}
UA!{requestPresentationPolicy}
BM!{fwdProvidePresentationPolicy}
{receivedPresentationPolicy}
Request Policy
User Agent
{requestPresentationPolicy}
U!{requestResource}
User
{requestResource}

Fig. 5. Aspect diagram for providing the presentation policy

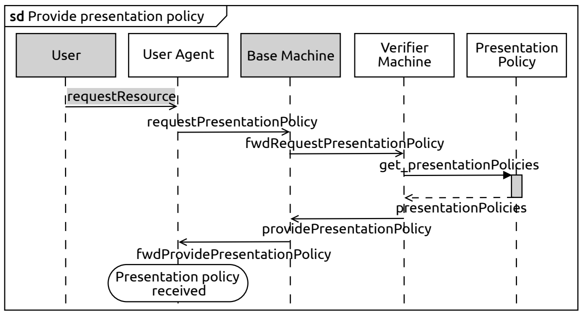
sd Provide presentation policy
User
User Agent
Base Machine
Verifier Machine
Presentation Policy
requestResource
requestPresentationPolicy
fwdRequestPresentationPolicy
get_presentationPolicies
presentationPolicies
providePresentationPolicy
fwdProvidePresentationPolicy
Presentation policy received

Fig. 6. Sequence diagram for providing the presentation policy

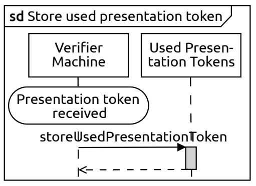
sd Store used presentation token
Verifier Machine
Used Presentation Tokens
Presentation token received
storeUsedPresentationToken

Fig. 7. Sequence diagram for the storage of used presentation tokens

provided by the User Agent needs to be verified to check whether the User is allowed to access the requested Resource using the Presentation Policy, Credential Specification, and Issuer Parameters. The result of the verification is sent to the Base Machine that then denies or provides access to the Resource for the User. For the sake of simplicity, we omit the corresponding aspect diagram. The necessary interaction between the domains to achieve the aspect is shown in Figure 8. The interaction can be started if the User Agent received the Presentation Policy. The User Agent then generates a respective presentation token (see Assumptions) for the user's request of a resource and requests the resource from the Base Machine using the generated Presentation Token. The Base Machine asks then the Verifier Machine to verify the received request. To do this, the Verifier Machine needs to retrieve the Presentation Policy, Credential Specification, and Issuer Parameters. The result of the verification is finally returned to the Base Machine. (3) The Presentation Tokens used by Users to request access to a Resource may be stored, e.g., for statistical or maintenance reasons. We left out the corresponding aspect diagram for the sake of simplicity. Figure 7 provides the behavioral view on this aspect. It specifies that if the Verifier Machine received a presentation token, then the machine can store that token in the lexical domain Used Presentation Tokens.
7.4 Weaving: The sequence diagram shown in Figure 9 shows how the three aspects are weaved into the base problem (see Figure 3). First, the arbitrary Before behavior of the base problem takes place and the User requests a Resource via his

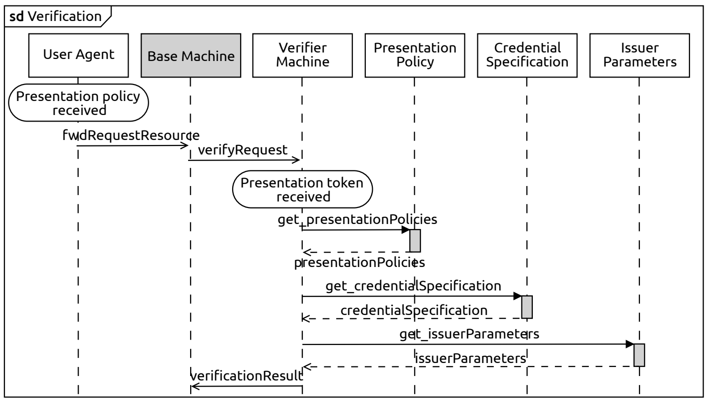

Fig. 8. Sequence diagram for the verification of presentation tokens

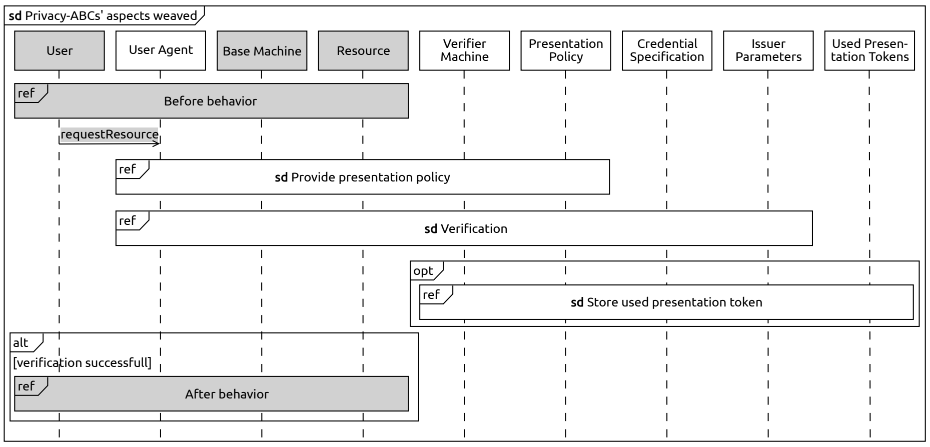

Fig. 9. Weaving of Privacy-ABCs' aspects into base problems

or her User Agent. The User Agent then requests the Presentation Policy (see Figure 6) to be able to generate the presentation token. Thereafter, the User Agent sends the generated presentation token that the Verifier Machine shall verify (see Figure 8). Optionally, the used presentation token, can be stored by the Verifier Machine (see Figure 7). Iff the Verifier Machine reports a successful verification, the Base Machine executes the After behavior, i.e., the User gets access to the requested Resource.

7.5 Base Problems: Privacy-ABCs introduce an additional requirement that does not cross-cut functionalities of the software-to-be. This is, the Verifier shall be able to specify his or her Presentation Policy that specifies which properties a User's Presentation Token must have to get access to a specific Resource. The Presentation Policy is based on the Credential Specification and the Issuer Parameters. The problem diagram for this additional base problem is shown in Figure 10.

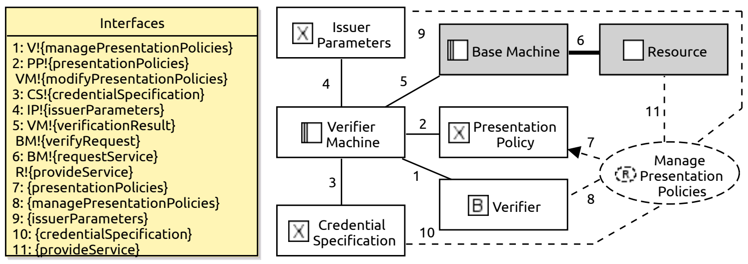
Interfaces
1: V!{managePresentationPolicies}
2: PP!{presentationPolicies}
VM!{modifyPresentationPolicies}
3: CS!{credentialSpecification}
4: IP!{issuerParameters}
5: VM!{verificationResult}
BM!{verifyRequest}
6: BM!{requestService}
R!{provideService}
7: {presentationPolicies}
8: {managePresentationPolicies}
9: {issuerParameters}
10: {credentialSpecification}
11: {provideService}
Issuer Parameters
Base Machine
Resource
Verifier Machine
Presentation Policy
Manage Presentation Policies
Verifier
Credential Specification
X
X
X
B
R
1
2
3
4
5
6
7
8
9
10
11

Fig. 10. Problem diagram for the management of presentation policies

If it is expected that further Issuer Parameters and Credential Specification from other Issuers need to be added or that the Issuer changes these in the future, then similar base problems need to be introduced that are concerned with the management of the lexical domains Issuer Parameters and Credential Specification. For the sake of simplicity, we omit the behavioral views for these base problems.

8 Design Issues If an existing Privacy-ABCs' infrastructure is used, there are not many design issues because most algorithms, protocols, and formats are prescribed. Only the presentation policy needs to be specified properly and the interface between the User and the Base Machine has to be refined with the User Agent (cf. Figure 4). If an own infrastructure shall be developed, several design decisions concerning algorithms, protocols, and formats have to be made. For the sake of simplicity, we omit the details on these issues.

9 Privacy Benefits

9.1 Confidentiality: ABCs can be used to reveal PI that shall be kept confidential only partially or to prove that the PI satisfy a certain condition without revealing it. For example, it could be proved that a user is older than 18 without revealing his or her exact age or date of birth.

9.2 Integrity: Issuers guarantee that the credentials they issue contain only authentic and correct data (with respect to the date these where issued). It is cryptographically ensured that (1) no entities except the issuers can create credentials, (2) the credentials cannot be modified to contain other data, and (3) the presentation tokens created from a credential can contain only information from this credential or proofs about its properties.

9.4 Anonymity: Presentation tokens are not linkable to their user (unless attribute values or other data outside the scope of Privacy-ABCs allow linking).

9.5 Data unlinkability: Presentation tokens are unlinkable to each other (unless attribute values or other data outside the scope of Privacy-ABCs allow linking and if it has not been explicitly specified that pseudonyms are used to be able to link specific presentation tokens to each other).

9.7 Pseudonymity: Privacy-ABCs can be used to implement transaction pseudonyms for presentation tokens, i.e., a new pseudonym is created for each presentation token. The presentation policy can also specify that specific presentation tokens are linkable to each other if the issuer parameters allow that. Hence, it is possible to implement other kinds of pseudonyms, e.g., role pseudonyms [9].

9.8 Collection information: The service provider has to specify a presentation policy that is used to generate the user's presentation tokens. This policy specifies which information needs to be encoded into the presentation tokens. The presentation policy can be assessed by the user via his or her user agent before a respective token is created. Note that if the revealed attributes do not allow the verifier to link them back to the individual they belong to, then the elicited information is not considered as PI and needs no further protection according to the EU General Data Protection Regulation [1].
9.12 Data subject intervention: If the revealed attributes do not allow the verifier to link them back to the individual they belong to, then the verifier does not need to provide specific intervention options to users.

10 General Benefits
10.1 End-user friendliness: If an existing Privacy-ABCs infrastructure can be used and the potential users already have appropriate credentials, then users do not need to explicitly register to use the software and they do not have to input their PI explicitly again. Users have to register only once at the issuer.
10.3 Costs: A Privacy-ABCs' infrastructure can be shared among several controllers that need to process the same or similar PI, or an existing infrastructure provided by an identity provider may be used. For example, the German eID card can be used by authorized and certified controllers to check whether a user's age is below or above a specified value [10].
10.4 Impact on software system's functionalities: It is possible that Privacy-ABCs replace another planned authentication mechanism and hence, make it unnecessary to manage user accounts and the like.
10.5 Abuse of PET: It is cryptographically ensured that corrupted tokens can be detected. Furthermore, the issuer guarantees that the data contained in the issued credentials are correct and belong to the user.

11 Privacy Liabilities
11.1 Confidentiality: The presentation policy specifies which information can be accessed by the verifier. It has to be specified in a way that only those PI is revealed that is necessary to carry out the verifier's duties.
11.2 Integrity: Some PI may change overtime, e.g., contact address. Hence, it may be necessary for users to request new credentials from an issuer and to invalidate the old credential. This issue is addressed by Privacy-ABCs with Revocation Authority [8].
11.4 Anonymity: The presentation policy specifies which information can be accessed by the verifier. If some provided information or other data outside the scope of Privacy-ABCs allow for linking, then anonymity may be broken.
11.5 Data unlinkability: The presentation policy specifies which information can be accessed by the verifier and whether the verifier is able to link presentation tokens to each other. The policy has to be specified in a way that presentation tokens can be linked to each other only if this is necessary.
11.7 Pseudonymity: The needed kind of pseudonym has to be specified in the verifier's presentation policy.

11.8 Collection information: The presentation policy specifies which PI is collected, however, verifiers may still need to inform users about the purpose for which the revealed information is used.
11.12 Data subject intervention: Under specific circumstances, it may be necessary to integrate a mechanism that allows users to order the deletion of presentation tokens or to restrict the processing of them (cf. Article 11 of the EU General Data Protection Regulation [1]).

12 General Liabilities
12.1 End-user friendliness: The usage of Privacy-ABCs has some issues concerning the end-user friendliness. First, users need to get credentials from an issuer that they need to trust. Second, users have to use a user agent for managing their credentials and generating presentation tokens. Hence, the perceived user-friendliness strongly depends on the properties of this user agent.
12.2 Performance: Depending on the complexity of the properties that need to be proved, the response time for the user could be higher than with a classical authentication mechanism.
12.3 Costs: The creation of an own Privacy-ABCs infrastructure, including issuing credentials and the development of user agents that generate presentations tokens, will be too expensive in most cases. If an existing infrastructure is used instead, it is possible that certain parts of the software need to be certified. Such a certification also raises costs.
12.4 Impact on software system's functionalities: It has to be ensured that the PI necessary to provide the requested services is collected and that (if necessary) users' interactions can be linked to each other.
12.5 Abuse of PET: If the software-to-be can be misused, e.g., to commit a crime or to damage the service provider, it is hardly possible to identify the malicious user (cf. **Privacy Benefits**). This threat can be mitigated by the Privacy-ABCs variant with Inspector.
12.6 Revocation: The basic Privacy-ABCs implementation provides no revocation options, but there are two extensions that provide different revocation options. The first extension allows revocation of credentials. That is, once issued credentials can be made invalid by a revocation authority. The second extension introduces the role of an inspector. The inspector is able to reveal the exact PI contained in a credential from a given presentation token or to uncover the individual to whom the presentation token belongs. The verifier shall only be allowed to request this inspection under specified circumstances that are also part of the verifier's presentation policy.

13 Examples If we apply Privacy-ABCs to the cigarette vending machine example, then the join point Base Machine (cf. Figure 2) would be instantiated with the vending machine, the Resource with the cigarettes, and the User with the customer who wants to buy cigarettes. In Germany, the existing Privacy-ABCs infrastructure of the German eID card [10] can be used. In this case the Issuer (cf. Figure 4) is the German state and the User Agent is the eID card. The Credential contains information such as the customer's name, address, date and place of birth. The Presentation Policy of the vending machine specifies that the generated Presentation Token only needs to contain a proof that the customer is older than 18.

14 Related PET Patterns Privacy-ABCs with Revocation Authority, Privacy-ABCs with Inspector

5 Discussion

Harrison states in [5]: *"The patterns community has been justly criticized for rehashing previously published material."* With this work, we want to emphasize that *"rehashing previously published material"* is necessary if the audience of the material is changed from researchers to practical requirements engineers, and beneficial if the rehashing leads to a homogeneous representation of PETs which makes it easier to identify and compare different solutions for the same privacy requirements with each other. However, our work yet lacks evidence that the proposed presentation of PETs as patterns using an aspect-oriented notion really helps requirements engineers to address questions (1)-(3) introduced in Section 1. In future work, we plan to empirically evaluate how much requirements engineers benefit from a catalog of PET patterns. We also expect that we get valuable feedback from the participants of the experiments to further improve the pattern format, e.g., by adding further pattern (sub)sections to it, and to improve the presentation of PETs as early aspects.

In addition to the question whether requirements engineers are willing to use a catalog of PET patterns, the question arises who will provide the PET patterns and maintain such a catalog. We would prefer an open platform, similar to existing platforms for privacy patterns (cf. Section 6), to which people who have experience with a certain PET can add a respective PET pattern.

In this paper, we have shown how Privacy-ABCs can be presented as PET pattern supported by an aspect-oriented notation. We additionally created PET patterns for K-Anonymity [11], P3P[2], Privacy-ABCs with Revocation Authority and Inspector, and the cryptosystem RSA [12]. These initial results have shown that both simple and more complex PETs can be represented as early aspects in the proposed pattern format.

Actually, the proposed pattern format is independent of the problem frames notation and aspect-orientation. That is, any other notation or only plain text may be used to describe the **Context**, **Problem**, and **Solution**. But we believe that the provided context, problem, aspect, and sequence diagrams help to illustrate the **Context**, **Problem**, and **Solution** of a PET. Especially, the *Weaving* shall support requirements engineers to understand into which base problems of the software-to-be a PET needs to be integrated and how.

Our proposed representation of PETs is dedicated to the requirements engineering phase, to support an early consideration of PETs and an integration of these into the other requirements of the software-to-be. Hence, our proposed pattern for Privacy-ABCs lacks information concerning concrete algorithms and

[2]`http://w3c.p3p.com` Accessed 21 Mar 2017

other implementation details, e.g., in which format presentation tokens or policies are stored. This is intended, because the focus in the requirements engineering phase should be on understanding the problem of building the software-to-be rather than on implementation details [3]. Hence, we want to understand which additional entities (domains) have to be considered if a PET is selected, how these entities are related to each other and the software-to-be, how the software-to-be needs to be adapted to integrate the PET, under which assumptions does the PET function, and with which benefits and liabilities does the PET come.

6 Related Work

Hafiz [13] presents a pattern language for developing PETs consisting of 12 patterns. Each pattern describes a solution to achieve a certain privacy property. The goal of the pattern language is to assist developers of PETs. Lobato et al. [14] propose patterns that support the presentation of privacy policies to users. Schumacher [15] presents two privacy patterns that describe best-practices for end-users to protect their privacy. Romanosky et al. [16] identified three privacy patterns for online interactions. These contain patterns for best-practices for end-users and best-practices for the design of privacy-friendly software. Porekar et al. [17] propose organizational privacy patterns. These privacy patterns shall help to address privacy issues already on the organizational level by providing corresponding solutions. The solution description is enhanced with Secure Tropos diagrams. In addition to the previously mentioned works, there are two websites[3] that provide catalogs of privacy patterns similar to the mentioned works. In contrast to these works, we propose to express PETs themselves as patterns to support requirements engineers to select appropriate PETs and to integrate them into the software-to-be to address identified privacy requirements. The consideration of PETs as early aspects is a novel contribution of this paper.

7 Conclusions

In this paper, we have proposed a pattern format to represent PETs as early aspects. We have illustrated how a PET pattern shall look like using the rather complex PET Privacy-ABCs. Initial results have shown that a variety of PETs can be expressed as PET patterns. The PET patterns shall help requirements engineers to identify the PETs that address the privacy requirements that they need to integrate into the software-to-be, then to select the PET that best fits to the needs without having too much impact on the software-to-be, and finally to integrate the PET's requirements into the requirements of the software-to-be.

In future work, we want to set up a larger catalog of PET patterns. Using this catalog, we want to empirically evaluate how much requirements engineers benefit from PET patterns. This is, we want to assess whether the catalog helps them to identify, select, and integrate PETs into a software system.

[3] `https://privacypatterns.org` and `https://privacypatterns.eu`

References

1. European Commission: Regulation (eu) 2016/679 of the european parliament and of the council (general data protection regulation) (April 2016)
2. ISO/IEC: ISO/IEC 29100:2011 Information technology – Security techniques – Privacy Framework. Technical report (2011)
3. Jackson, M.: Problem Frames. Analyzing and structuring software development problems. Addison-Wesley (2001)
4. Faßbender, S., Heisel, M., Meis, R.: A problem-, quality-, and aspect-oriented requirements engineering method. In: Software Technologies - 9th International Joint Conference, ICSOFT 2014, Revised Selected Papers. Volume 555 of CCIS., Springer (2015) 291–310
5. Harrison, N.B.: Advanced pattern writing – patterns for experienced pattern authors. In Manolescu, D., Voelter, M., Noble, J., eds.: Pattern Languages of Program Design 5. Addison-Wesley (2006)
6. Meis, R., Heisel, M.: Computer-aided identification and validation of privacy requirements. Information **7**(28) (2016)
7. Meis, R., Heisel, M.: Computer-aided identification and validation of intervenability requirements. Information **8**(30) (2017)
8. Camenisch, J., Krontiris, I., Lehmann, A., Neven, G., Paquin, C., Rannenberg, K., Zwingelberg, H.: D2.1 architecture for attribute-based credential technologies – version 1. Technical report, ABC4Trust (2011)
9. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management (August 2010) v0.34.
10. Deutscher Bundestag: Gesetz über Personalausweise und den elektronischen Identitätsnachweis sowie zur Änderung weiterer Vorschriften. Bundesgesetzblatt **I**(33) (2009)
11. Sweeney, L.: K-anonymity: A model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **10**(5) (October 2002) 557–570
12. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2) (February 1978) 120–126
13. Hafiz, M.: A pattern language for developing privacy enhancing technologies. Software: Practice and Experience **43**(7) (2013) 769–787
14. Lobato, L.L., Fernandez, E.B., Zorzo, S.D.: Patterns to support the development of privacy policies. In: Procs. of the 1st Int. Workshop on Organizational Security Aspects (OSA). (2009)
15. Schumacher, M.: Security patterns and security standards - with selected security patterns for anonymity and privacy. In: European Conference on Pattern Languages of Programs (EuroPLoP). (2003)
16. Romanosky, S., Acquisti, A., Hong, J., Cranor, L.F., Friedman, B.: Privacy patterns for online interactions. In: Proceedings of the 2006 Conference on Pattern Languages of Programs. PLoP '06, New York, NY, USA, ACM (2006) 12:1–12:9
17. Porekar, J., Jerman-Blazic, A., Klobucar, T.: Towards organizational privacy patterns. In: Second International Conference on the Digital Society. (Feb 2008) 15–19