Journal of Universal Computer Science, vol. 16, no. 3 (2010), 341-371
submitted: 7/5/09, accepted: 29/1/10, appeared: 1/2/10 © *J.UCS*

Enabling Personal Privacy for Pervasive Computing Environments

Susana Alcalde Bagüés, Andreas Zeidler, Cornel Klein
(Siemens AG, Corporate Technology, Munich, Germany
{susana.alcalde, a.zeidler,cornel.klein}@siemens.com)

Carlos Fernandez Valdivielso, Ignacio R. Matias
(Public University of Navarra
Department of Electrical and Electronic Engineering
Navarra, Spain
{carlos.fernandez, natxo}@unavarra.es)

Abstract: Protection of personal data in the Internet is already a challenge today. Users have to actively look up privacy policies of websites and decide whether they can live with the terms of use. Once discovered, they are forced to make a "'take or leave"' decision. In future living and working environments, where sensors and context-aware services are pervasive, this becomes an even greater challenge and annoyance. The environment is much more personalized and users cannot just "'leave"'. They require measures to prevent, avoid and detect misuse of sensitive data, as well as to be able to negotiate the purpose of use of data. We present a novel model of privacy protection, complementing the notion of enterprise privacy with the incorporation of personal privacy towards a holistic privacy management system. Our approach allows non-expert users not only to negotiate the desired level of privacy in a rather automated and simple way, but also to track and monitor the whole life-cycle of data.

Key Words: personal privacy, pervasive computing, privacy architecture, user manageability
Category: D.2.11, K.4.2, K.6.5, L.7

1 Introduction

Pervasive computing has the potential of deeply affecting many aspects of our daily lives. In the same way that cell phones and mobile services are changing our awareness of and the interaction with our friends and family today, tomorrow, the advent of the pervasive computing age will foster the deployment of a huge number of context-aware mobile services (CAMS). Our everyday routines may change drastically and new issues may emerge. One of the fundamental problems is that pervasive computing raises serious privacy concerns. The collection of data related to the context of an individual increases hugely. It is easy to process and share personal information with a large number of smart services. We believe that pervasive computing will change our perception of privacy in an even more substantial manner than the Internet did.

Privacy is already a prime concern in today's information society. The challenge now is to design pervasive computing systems including effective privacy protection mechanisms. Obviously, there will not be a single "magic" solution to safeguard individuals' privacy. Pervasive privacy protection will require the integration of a significant number of heterogeneous mechanisms and the cooperation of different entities. Entities involved in the process of gathering and compiling personal data. Including those entities that not always can be trusted with respect to the enforcement of privacy, since certain interests (especially commercial ones) might prevail over enforcing someone's privacy preferences.

Our work is based on the claim that it is not sufficient to rely only on enterprise's willingness to support individual's privacy requirements, as it is typically deemed sufficient today. Protecting individuals' privacy in pervasive computing requires an additional level of privacy protection, which we named personal privacy enforcement: Together with the enforcement of enterprise's privacy statements, government policies and privacy laws, it is necessary to deploy mechanisms specifically for defining, communicating and enforcing people's privacy preferences. This is in accordance to what Marc Langheinrich stated in [Lan01]: *"Protecting people's privacy is a very personal affair. Something that cannot be solved without taking people's habits, preferences, and moral views into account"*. Thus, we need means to empower users to decide by themselves on the exchange of personal data on a much simpler, automatic, and finer-grained level than it is possible today to prevent them from losing their privacy to enterprises and data sellers in the near future [OAB07].

Our approach is to make privacy protection a cooperative task of all parties involved. We developed a novel privacy enforcement model based on a *Trusted Privacy Manager*, as a central abstraction. It orchestrates disclosure of personal data and manages the collaboration with third parties. We designed and developed the *User-centric Privacy Framework* as reference implementation of such a Trusted Privacy Manager. When we addressed its realization, we had to keep in mind three conflicting user wishes. Users want to keep their sensitive data private. On the contrary, they also want to get the most value from the services they use. And they want to understand what is going on. These are the three principles that we have to balance, namely: *Privacy Protection*, *Service Usability* and *User Manageability*. We are confident that it is possible to make great progress in pervasive privacy protection by building a solid privacy model, like the one introduced in this paper, which favors the combination and integration of personal and enterprise privacy.

2 Privacy Protection in Pervasive Computing

Laws and enterprise privacy alone are not sufficient to prevent an unwanted intrusion into the private sphere of an individual. Privacy is a very subjective concept: What is acceptable to one person may be unacceptable to another. Users of pervasive computing systems need mechanisms to control and manage their own privacy concerns. For instance, by limiting who has access to what data and under which circumstances, which typically may be achieved by using privacy policies.

However, purely technological solutions, such as policy systems for access control, by themselves, can achieve privacy goals only in certain situations. They are necessary

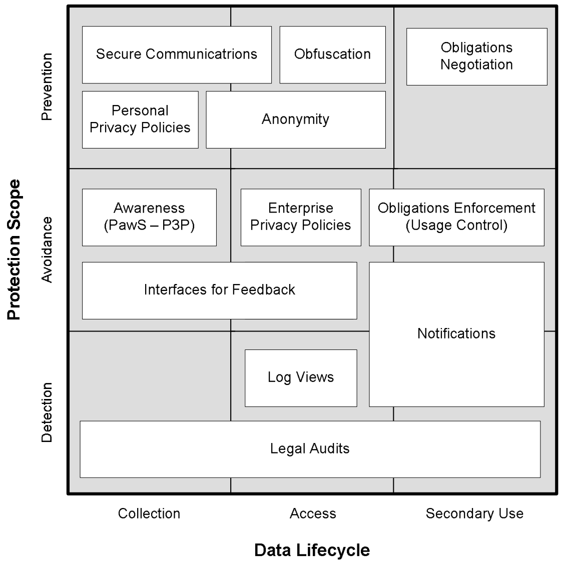

Figure 1: Privacy Design Space for Pervasive Computing

but not sufficient. Privacy control in pervasive computing requires the integration of several complementary mechanisms following a new design space.

We have classified technical solutions into measures for: *privacy aware access control*, *anonymity*, *obfuscation*, *usage control*, *awareness*, *feedback* and *detection*, and represented them within the privacy design space defined by Jiang et al. [JHL02]. This privacy design space, shown in Figure 1, is taken as the reference frame for the solutions presented in this article. We consider it to represent and cover quite accurately the new privacy needs of users of pervasive computing applications. It categorizes privacy protection technologies into measures for prevention, avoidance, and detection at the different stages of the data lifecycle: collection, access, and secondary use.

Privacy in pervasive computing environments requires the deployment of measures for: i) *prevention*, to ensure that undesirable use of personal data does not occur. Within the group of preventative measures we include privacy aware access control, obfuscation and anonymity; ii) measures for *avoidance*, to minimize the risks and maximize the benefits associated with data exchanges. By using mechanisms for awareness, feedback and usage control, the flow of information from data consumers back to data owners can be increased. Users need to be aware of personal privacy issues to properly apply privacy control and avoid risky situations during the three phases of the data life cycle; iii) finally, measures for *detection*, to detect unwanted misuses of disclosed data and act accordingly.

Protection during *collection* refers to the event in which personal data is initially collected. Important decisions should be made at this phase including who can collect what data, under which circumstances, in which format, and how trusted is the data collector. *Access* refers to the point at which data is accessed, for a particular purpose, by the ultimate recipient. Important decisions concerning this phase are how accurate and how confident the data should be, who should be allowed to access it, for what purpose, how accesses should be logged or how data is supposed to be stored. *Secondary use* refers to the use of data after initial access has been made. Secondary use may also include passing data from one party, who might have been authorized, to another party, who might be not. Important decisions made in this phase should include who else should be able to access the data, secondary use purposes, and whether it is allowed to share the data with others or how long the data can be stored.

Summing up, within this privacy space we classify personal privacy policies as a preventative mechanism used at collection time. The data owners can specify the list of constraints that need to be fulfilled before granting a collection request. Enterprise policies are enforced afterwards by the receiving service, to limit the access to authorized service's users and only for authorized purposes, acting as an avoidance and preventative measure at access. On the other hand, technical solutions for awareness are considered as an avoidance measure during collection, since their deployment allow users to be aware of collection practices beforehand of actually revealing information. Obfuscation and anonymity can be used to prevent that data, when accessed, gives away sensitive information, e.g. by not revealing the identity of the data owner or disclosing his location with granularity "city" instead of "building". For secondary use, we consider the specification and negotiation of obligations as a preventative mechanism, and their enforcement, by the enterprise recipient, an avoidance measure. By ensuring that the service, recipient of the data, is able to enforce a binding agreement on a set of obligations, misuses after initial collection can be avoided. Furthermore, it is also important to implement mechanisms for providing feedback and notify users of issues related to previous exchanges of data. With respect to detection measures, common mechanisms are data logs and legal audits. Additionally, in our work we introduce the use of notifications to actively monitor the ongoing state of disclosed data.

2.1 Related Work

There exist several known approaches to privacy but so far none of them have addressed privacy as we do. Our goal is to enable a normal, common user to control privacy by himself, in terms of managing functionalities for access control, usage control, obfuscation, awareness, feedback as well as detection. We exclude in our model only anonymity, due to the fact that it is not applicable to the kind of personalized services considered.

P3P is probably the best-known approach to privacy policies [P3P02]. It has been developed at the World Wide Web Consortium (W3C), which standardized the Platform for Privacy Preferences in 2002. P3P enables web sites to encode its data collection and data use practices in a machine-readable XML format, known as a P3P policy. In other words P3P provides a declaration format to announce data collection practices and make a user aware of the service privacy policy. Nevertheless, the current P3P standard only provides a mechanism to publish service side privacy policies but the enforcement of such policy and thus, ensure that the web site acts according to its stated policy is beyond the scope of P3P [Dek07]. Thus, P3P falls into the category of awareness measures instead of measures for access control. The Platform for Enterprise Privacy Practices (E-P3P), introduced in 2002 by Karjoth [KS02b], addressed exactly this problem. E-P3P defines an enterprise privacy enforcement system for the enforcement of enterprise-internal privacy policies. This enables for instance, to internally enforce the E-P3P privacy policies while promising a P3P statement to its customers.

However, as stated in [Cue02], P3P has not been tailored to the specific requirements of pervasive applications. PawS, a Privacy Awareness System for ubiquitous computing developed by Langheinrich [Lan02], extends P3P to cover aspects of pervasive applications. In PawS when a user enters an environment in which services are collecting data, a privacy beacon announces privacy policies of each service. A user's privacy proxy then compares these policies against the user's own privacy preferences, which can be written using APPEL [APP02]. If the policies match, services are allowed to collect information and users can utilize the services in return. If the policies do not match, the system notifies the user, who then may choose not to use the service in question or, in some cases, simply physically can leave the area in which the collection of information occurs. This approach provides users with avoidance and preventative measures at data collection but it cannot be used with the type of services, e.g. "Buddy Finder", which may not gather context data directly from the environment but depend on external context providers.

Furthermore, while APPEL provides a good starting point for expressing user privacy preferences, it cannot support the richness of expressions needed in pervasive computing scenarios. In [MFD03] user requirements for a privacy policy system are detailed with emphasis in the granularity of constraints users might want to apply to control the distribution of their location information, e.g *dynamic constraints* on the location and activity of the user and recipient. Here, rules are implemented as system components called *validators* without defining a concrete implementation language, though.

EPAL [EPA03] and XACML [XACML] are two other platform-independent languages that support the definition and enforcement of privacy policies for access control, mainly oriented towards the specification of enterprises statements. In [And05] a comparison of both languages shows that EPAL offers only a subset of the functionalities of XACML. XACML has been developed for some time and has reached a high level of standardization, but it has only started recently to take possible privacy constraints on information management into account. It may be enough for enterprise privacy control but it still lacks of some important features to specify personal privacy policies.

The mentioned works are focused mainly on providing enterprises with the means to enforce and publish their privacy policies. A different approach, oriented to the end-user is the Confab toolkit, where information is captured, stored, and processed on an end-user's device [HoL04]. This gives end-users a great level of control and choice over what personal information is disclosed but falls short in its flexibility for sharing information in pervasive computing, since it requires the information being captured and exchanged only between devices which hold the Confab system. The authors introduce a privacy sensitive architecture divided into three orthogonal layers: the physical/sensor layer, which is responsible for initially capturing personal information; the infrastructure layer, which is responsible for storing and processing personal information; and the presentation layer, which is responsible for providing user interfaces to give end-users control and feedback over their information. In the Confab architecture an initial step was given in the direction of integrating different privacy measures. It provides access control, feedback and a basic implementation of usage control. Confab introduces the *privacy tag* in an initiative to integrate usage control; it provides hints on how the data disclosed should be used outside of the user's direct control.

In the context of usage control obligations has been introduced as requirements that must be agreed on by obligation subjects (recipient of the data), before authorizing the access to the data. Obligations can be seen as a binding statement to take some course of action in the future by the obligation subject [KFJ03]. They are crucial to restrict the flow and use of personal data in the highly dynamic distributed environments assumed in pervasive computing. Nevertheless, in many policy systems obligations have been defined tightly coupled to the enforcement of access control policies and in general they cannot be used for usage control. This is the approach followed by EPAL [EPA03] and XACML [XACML]. Within these access control languages obligations are a set of operations associated with a policy that must be executed by the *Policy Enforcement Point* together with an authorization decision before allowing the access to the data. In the work of Park and Sandhu [PaS04] obligations are requirements that have to be fulfilled by a subject at the time of the request for allowing access to a resource. E.g., a user must give his name and email address to download a company's white paper.

As an extension to XACML, the work presented in [MCC07], introduce a new approach to archive digital signed commitments on obligations between distributed parties. They present the *Obligation of Trust* (OoT) protocol, which executes two consecutive steps: *Notification of Obligation* and *Signed Acceptance of Obligation*. The OoT is built upon the XACML standard following its Web Services Profile (WS-XACML) to address the integration of usage control measures. The disadvantages of this work are that it does not cater for the enforcement and monitoring of such obligations, on the one hand, and that it seems to be rather complicated for a common user to manage obligations following this protocol, on the other hand.

In the proposal, released by the Internet Engineering Task Force Working Group on Geoprivacy (Geopriv) [STM07], an authorization policy language for controlling

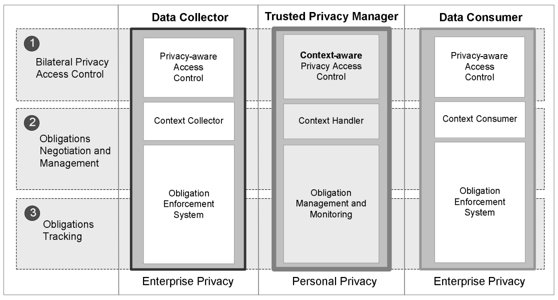

Figure 2: Main characteristics of our Privacy Enforcement Model

access to location information is detailed. Although this schema is not designed to cover individual privacy preferences, it has the advantage that it supports the obfuscation of a user location by defining the accuracy with which the data should be revealed.

Summing up, users always need to actively apply prevention throughout the whole data lifecycle. They require also mechanisms for detection and privacy enforcement by enterprise privacy frameworks for avoidance. When comparing existing related work with the privacy design space we use in our work, no privacy system is able to integrate all preventative mechanisms during collection, access and secondary use. Some policy languages tackle the idea of obfuscation, e.g., Geopriv; others include hints on how the data should be used, e.g. Confab. Related work on the side of the enterprise, such as E-P3P, XACML or EPAL, provide part but not all of the desired functionality. Thus, none of today's approaches is able to address the integration of personal privacy and enterprise privacy as two different but complementary aspects of privacy protection.

3 Our Privacy Enforcement Model

For addressing the integration of personal privacy and enterprise privacy enforcement, we developed a three-tier privacy enforcement model, shown in Figure 2. It serves as basis for the specification of the involved entities and their relationships. Its main feature is that it integrates measures for addressing prevention, avoidance and detection during the data lifecycle. Thus, it covers the privacy design space described in Section 2. The privacy enforcement model consists of the following three pillars:

1. **Data Collector**: This pillar provides abstractions for representing any data collector as typical source of context information. We classify them into two groups: i) the group of data collectors under direct control of a user, for example, a GPS tracking device. In general, they are sources of personal context that do not represent a privacy risk, *per se*; ii) data collectors that are operated by enterprises, e.g. mobile network operators. Langheinrich addressed in his work the privacy implications of

those kinds of data collectors and provided a Privacy Awareness System (PawS) to help individuals deal with them. In our model we assume that a similar framework to PawS [Lan02] is in place to handle them.

2. **Trusted Privacy Manager**: This central pillar provides abstraction for a personal and trusted privacy enforcement system. The role of the Trusted Privacy Manager is to orchestrate any disclosure of personal data and to manage the collaboration between entities with the goal to protect an individual's privacy. Thus, it must provide functionality for the evaluation of service-side privacy policies (data collectors and data consumers), the enforcement of its users' privacy policies, for guiding the negotiation of agreements on secondary use, and facilitating the monitoring of disclosed data.

3. **Data Consumer**: The last pillar represents abstractions for data consumers, in particular consumers of context information that depend on external data collectors to implement their respective services, e.g a "Restaurant Finder" or a "Buddy Finder" service. Obviously, these are the main types of emerging services in pervasive computing. However, context providers and service providers are typically decoupled and operate independently from each other and, thus, must be orchestrated.

Our privacy enforcement model is built around direct, one-to-one binding agreements between the Trusted Privacy Manager and the other two pillars. The Trusted Privacy Manager acts as mediator between data collectors and data consumers. It reduces and simplifies potentially complex relationships between them to contractual two-party agreements. It avoids direct exchange of data between data collectors and data consumers. The idea behind the use of agreements is twofold: to specify authorized actions and usage purposes and to avoid uncontrolled disclosures among data collectors and consumers.

We have defined three levels of collaboration to enable the integration of personal and enterprise privacy enforcement, as it is shown in Figure 2, namely: *Bilateral Privacy Access Control*, *Obligations Negotiation and Management*, and *Obligations Tracking*. They are complementary and depend on each other.

The first level of collaboration (*Bilateral Privacy Access Control*) demands from each party to include privacy-aware access control mechanisms. Note that, only in the case of the Trusted Privacy Manager access control should be context-aware. Since, it is the only tier that is allowed to have direct access to users' context data. Bilateral access control means, for us, that both, the service side privacy policy and the user's privacy policy (data owner) are evaluated beforehand of any disclosure. How this is achieved may differ from case to case.

The second level, the *Obligations Negotiation and Management*, requires from all three pillars to adopt a common model for the specification of obligations and trust that data collectors and data consumers will handle personal data according to such obligations. The third level *Obligations Tracking* was introduced as an initial measure for trust management and to allow lifecycle awareness of the data. Our approach to establish a trusted relationship between an enterprise service and the Trusted Privacy Manager is based on the possibility to subscribe to notifications about the use of disclosed data.

In the collaboration of the Data Collector pillar with other parts of the system, as it was mentioned already, we rely on the idea elaborated by Langheinrich. There, PawS allows data collectors to describe their collection policies in a machine-readable format and communicate them to their data subjects. If the user agrees the services can collect information and users can utilize the services. Thus, we can assume that a user is notified, via a Trusted Privacy Manager, before a new data collection is started, (e.g. video surveillance or indoor security tracking). The user then can check the service's privacy policy and either agrees to or denies the collection. Due to the implicit nature of this type of data collectors, a user in general can only accept or reject the collection of the data in a "take or (physically) leave" fashion.

For the Data Consumer, we assume that any consumer service first has to register with an instance of the Trusted Privacy Manager. Upon registration the service discloses its privacy policy. The policy (e.g. P3P) shall have at least the following elements: data consumer description, data elements the service requests, and purpose of the request. If the registration is successful, the Trusted Privacy Manager agrees on providing context broker functionality for the subscribed service. From this moment on and for each service request, a Trusted Privacy Manager entity evaluates its users' privacy policies to decide whether or not the request is granted. Thereby, *bilateral privacy access control* is enforced. The data is transmitted only if all the restrictions and conditions defined by the user are fulfilled. This avoids unwanted "take or leave situations" in interactions with Data Consumers.

However, bilateral privacy access control only addresses situations where information is disclosed for the present use. It only provides preventative and avoidance measures during collection and access. Thus, it does not cover cases where information may be stored for future use or even be sold in a potentially different situation. Here is where the second and third collaboration levels come into play. They introduce additional mechanisms for detection and secondary use control. As a result, our privacy model demands from enterprise services (data collector and data consumers) not only to apply access control, but also to actively accept and enforce obligations on behalf of the user. This notion of obligations enforcement is in accordance with the work of Casassa et al. [CAT06], where obligations are first class entities enforced by enterprises to comply with current legislation and enterprise guidelines. Obligations are complemented in our model by an agreement negotiation protocol plus notification mechanism to enhance a user's confidence in privacy enforcement by enterprises.

Our model assumes that the enterprise service agrees on taking over the task of protecting users' data by truthfully fulfilling the negotiated obligations. Once the data is disclosed the Trusted Privacy Manager can be configured to remain in a stand-by monitoring mode. Of course, trustworthiness forms an integral part of this model for

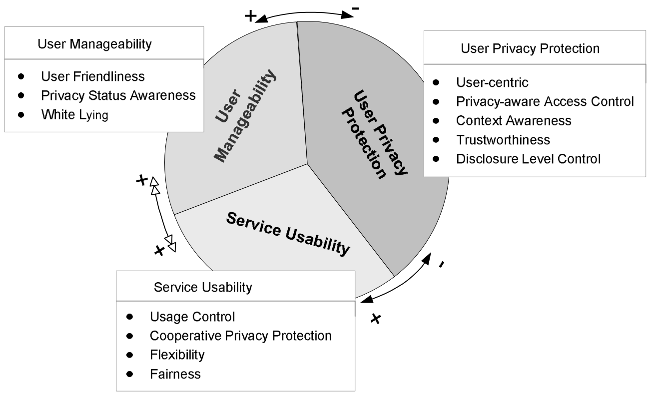

Figure 3: Requirements for Personal Privacy Protection

the reason that releasing personal information to a third party requires the trust that the receiving party will treat the information only in the agreed manner. However, the same happens in our daily life, where we trust our peers to keep confidential information to themselves. The application of known social patters, such as individual's trust on collaborations of a contractual nature (agreements) and on enterprises' reputation, will make it easier for users to understand privacy control and generates more willingness to manage their privacy actively.

4 A Trusted Privacy Manager

In this section we present the principles and requirements that a Trusted Privacy Manager should realize. As research on personal privacy protection in pervasive computing is a rather novel topic, an important result of our work was the identification of the requirements that a Trusted Privacy Manager need to address. Figure 3 shows those requirements grouped together into three subsets. During their definition our goal was to keep the balance between *Privacy Protection* and the two other key success factors *Service Usability* and *User Manageability*. We want to provide non-expert users with the means to control privacy but without forgetting the main reason for all this; users need privacy protection because they want to use the offered services.

Our first design principle, Privacy Protection, addresses the definition and enforcement of users' privacy preferences for the whole data lifecycle. The second design principle, Service Usability, is focused on providing "fair" interactions. The idea behind is to avoid that once a user starts using a service the interaction flow is terminated because the user denies the access to his data. One of our main contributions here was, to make the enforcement of a user privacy policy directly dependent on the context in which a service request, as data collector, was made and whether or not a user has previously requested the service.

The third key principle is related with how to enable common users to manage privacy on their own. User Manageability requires, among others issues, to limit the number of parameters that a user needs to configure. A mistake would be to overwhelm users with the burdensome task of creating and managing privacy (further information can be found in Section 5.3). In general, simplicity and user-friendliness lead to the loss of flexibility and expressiveness, e.g. in the number and type of restrictions the user can understand, but also to an increase in manageability.

The tradeoffs are that the more privacy protection is desired, the less service usability and user manageability can be achieved, and vice versa. On the other hand, service usability and user manageability may influence each other positively; by improving service usability, we may enhance user manageability in the context of privacy awareness. Users want to receive feedback from the services with respect to the use of disclosed data. That requires from services to collaborate with respect to the enforcement of obligations and exchange of notifications, as described in Section 5.2.

Before detailing our reference implementation of a Trusted Privacy Manager, next we outline the mentioned requirements, which have guided the overall design of our privacy framework for keeping the mentioned three principles balanced.

4.1 Principles and Requirements

i) *Privacy Protection*: the realization of this principle involves five design requirements to be implemented.

1. **User-centric**: The first principle of the OECD guidelines for Data Collection [OECD], called "Collection Limitation", states that sensitive data should be obtained by lawful and fair means and with the consent of the data subject. Therefore, individuals need mechanisms to define under which circumstances data should be disclosed. In order to offer a controlled distribution of sensitive personal data and spare individuals from spending time on setting their privacy preferences for each encountered service separately, the collection and distribution of users' personal data should be centralized.

2. **Privacy-aware Access Control**: As a corollary to the previous one, this requirement states that the system shall provide appropriate access control mechanisms for allowing the specification and enforcement of user privacy policies during collection and thereby restricting the "when, what, how, and who" of accessing personal data.

3. **Context Awareness**: Recent studies on the perception of privacy [AHK07] by individuals concluded that user preferences vary depending on place and social context. Thus, privacy policies should be made context-aware. In addition to the typical restrictions on the recipient or on the purpose of the data collection, *dynamic constraints* related to a user's environmental context should be

possible. Most attributes that describe an individual and the environment are dynamic context, e.g. location, time, temperature, blood pressure, activity, etc.

4. **Trustworthiness**: Trust was defined by e.g. Luhmann in 1968 as *"a mechanism to reduce social complexity"* [LUH79]. Privacy enforcement needs to trust the involved parties in that they will fulfill their duties with respect to privacy protection. Furthermore, the enforcement of context-aware policies entails the disclosure of sensitive information during the policy evaluation (e.g. location, activity, calendar), which raises the requirement of a new actor, a so-called Trusted Privacy Manager, in charge of evaluating personal privacy policies and coordinating the duties that untrusted parties shall adhere to.

5. **Disclosure Level Control**: As it was mentioned in Section 2 there is a need for preventative mechanisms during data access. Data obfuscation measures should be provided to control the precision of the information to be disclosed, allowing fine-grained control over the quality of data transmitted. This requirement was already introduced by Hong et al. in [HoL04] and mentioned in "Data Quality Principle" of the OECD guidelines.

ii) *Service Usability*: the realization of this principle added four extra requirements to our design.

6. **Usage Control**: Usage control extends access control and encompasses all those mechanisms that deal with future uses of the data disclosed and with the detection of privacy violations. This requirement has two aspects: to ensure privacy protection and at the same time service usability. For a user it is important to delimit what a recipient of his data is allowed to do, also stipulated by the OECD as "Use Limitation Principle". From the point of view of an enterprise it is mandatory to enforce user preferences in order to follow existing privacy legislation. The alignment of both will increase the confidence into and promote the further deployment of CAMS [SSA06].

7. **Cooperative Privacy Protection**: Pervasive privacy is only feasible if all entities involved in the exchange of personal information collaborate with each other on the enforcement of the user's privacy preferences. As the OECD "Accountability Principle" states, any data consumer should be accountable for complying with measures for the protection of such data. The enforcement of personal privacy and enterprise privacy both requires a new privacy enforcement model to formalize such collaboration as described in Section 3.

8. **Flexibility**: Obviously, the interaction with different data collectors and data consumers requires a flexible and distributed architecture, as described in Section 5.2.2, and some awareness of the underlying semantics. This recommends the use of a semantic representation model for privacy policies and context

data, which provide a common reference for the collaboration of the different entities involved.

9. **Fairness**: The system should enable "fair" communication following the "Fair Information Trade" principle. The realization of this principle gives another perspective on privacy control, where information is disclosed as "payment" for the service. [EHG07].

iii) *User Manageability*: the realization of this principle will enable non-expert users to manage privacy.

10. **User Friendliness**: There is a strong requirement for a common, easy-to-use interface for giving a user the possibility of managing his privacy. This is a common requirement often mentioned, although at present no implementation is available and also reference work such as The Faces Metaphor [LMD03] was discontinued.

11. **Awareness of Privacy Status**: "Openness", a principle of the OECD guidelines, tries to guaranty that users are aware of all the issues related to personal data. The increase of information flow back to the data owners is related with measures for avoidance and detection during data access and secondary use. People should be provided with means to access information related with the disclosure and use of his data.

12. **White Lying**: By definition, pervasive computing environments are supposed to be largely automated and "always on". In a certain sense it follows that people do not have the possibility to "switch-off" the system. We introduce the requirement of adapting the concept of white lies as a way to "disconnect" individuals temporarily from a pervasive computing environment in a plausible way and maintain standard social interaction patterns, our approach to the integration of white lies is described in [WLI07].

5 User-Centric Privacy Framework

Our reference implementation of a Trusted Privacy Manager entity is called "User-centric Privacy Framework" (UCPF). The remainder of this section is dedicated to the description of the key elements of the UCPF and its functional decomposition into services as being installable on a residential gateway for our Smart Home [AZF07] lab. The Smart Home in general provides an ideal target environment to deploy and test a privacy enforcement system for a controlled and small number of users. It naturally meets the requirements of being *User-centric* and *Trustworthy*. The residential gateway enables access to home-based services from inside and outside the home environment. The incorporation of the UCPF adds privacy control and context brokering as separate functionalities and allows inhabitants to interact with external CAMS. Part of our work,

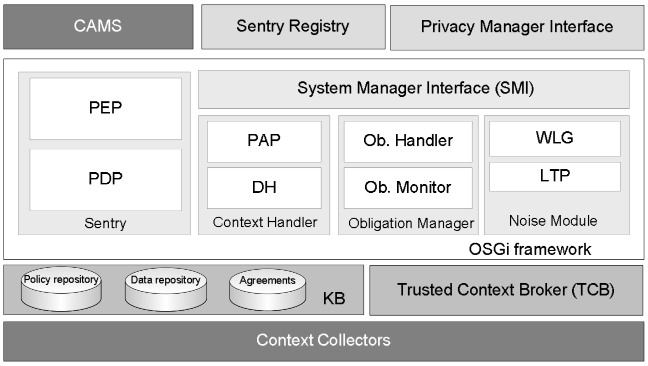

Figure 4: User-centric Privacy Framework Architecture Overview

on its design, was also used in the privacy context manager system of the IST project CONNECT [AME07].

The UCPF architecture has been designed according to the requirements identified in the previous section. Figure 4 shows a high-level view on the global architecture. A Service Oriented Architecture (SOA) framework hosts all UCPF components, except for the Sentry Registry. For our implementation we used the Knopflerfish OSGi Service Platform [Knopf]. OSGi defines a standardized component-oriented environment for networked services and handles the typical service management tasks.

The UCPF consists of five core services implemented as bundles within the OSGi framework namely, the Sentry, the System Manager Interface (SMI), the Context Handler (CH), the Obligation Manager (OM), and the Noise Module (NM). Apart from them, the UCPF incorporates a web registry (Sentry Registry) and a graphical interface for the user, named the Privacy Manager Interface.

In the following sections, we describe how the three core design principles of our model are mapped to the functional specification of each of the UCPF components.

5.1 Providing Privacy Protection

The most fundamental goal of a Trusted Privacy Manager is to enable personal privacy protection. The realization of this principle involves, as was detailed in Section 4.1, five requirements to be implemented, namely: *User-centric*, *Privacy-aware Access Control*, *Context Awareness*, *Trustworthiness*, and *Disclosure Level Control*. In other words, we need a privacy proxy responsible of taking care of privacy-related data, when being accessed from any data consumer. This privacy proxy should control the dissemination of data within the interaction chain as shown in Figure 6, based on a set of context-aware privacy policies defined by the user.

For illustrating the functionalities of such a privacy proxy we first consider a typical interaction chain between a user and a context consumer (e.g. a CAMS, c.f. Figure 5).

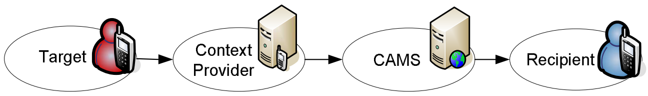

Figure 5: Typical Interaction Chain

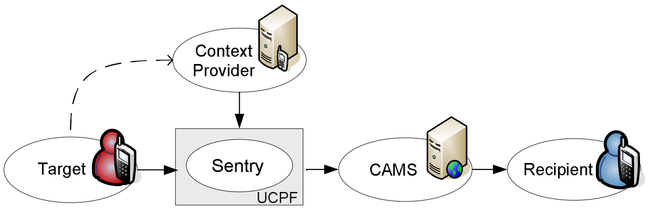

Figure 6: Protected Interaction Chain

The data flow along that interaction chain involves different autonomous entities like people, companies or organizations who typically adopt one specific role at a time. We distinguish the following general roles: i) A “Target” is the tracked individual; ii) The “Service Provider” (data consumer) compiles context information for its users; iii) The “Recipient” (one or many) is the user of the service; iv) The “Context Provider” (data collector) acts as an intermediate entity, responsible for collecting, caching and managing of context information and for disseminating it, accordingly.

It is obvious that potentially malicious third parties can take advantage of this completely open interaction setting. Except for the target itself, all other entities along the interaction chain are classified as untrusted, since they are not under the direct control of the user. Even if a contractual relationship exists between the entities (e.g. between a user and a mobile operator) we consider this not to be enough for completely trusting such third-party services with the enforcement of context-aware privacy policies. Therefore, we introduced the Sentry, as our privacy proxy within the UCPF, for taking-over a new role in the interaction chain (see Figure 6). It constitutes a trusted personal privacy enforcement point that controls all accesses to privacy-relevant data of its users and meets our requirements of being *User-centric* and providing *Trustworthiness*. A Sentry’s main purpose is to free a user from manually consenting or dissenting to privacy statements as they commonly need to do in today’s Internet services.

Other UCPF functional components needed to implement Privacy Protection a part from the Sentry are the Context Handler (CH) and the Noise Module (NM). Figure 7 shows a typical interaction sequence between those components triggered by a CAMS request. For the sake of simplicity, in this first example, we left out the negotiation of agreements, which is detailed in the following section. It is important to note that the Sentry as specified by the UCPF is acting as a façade implementation for privacy protection.

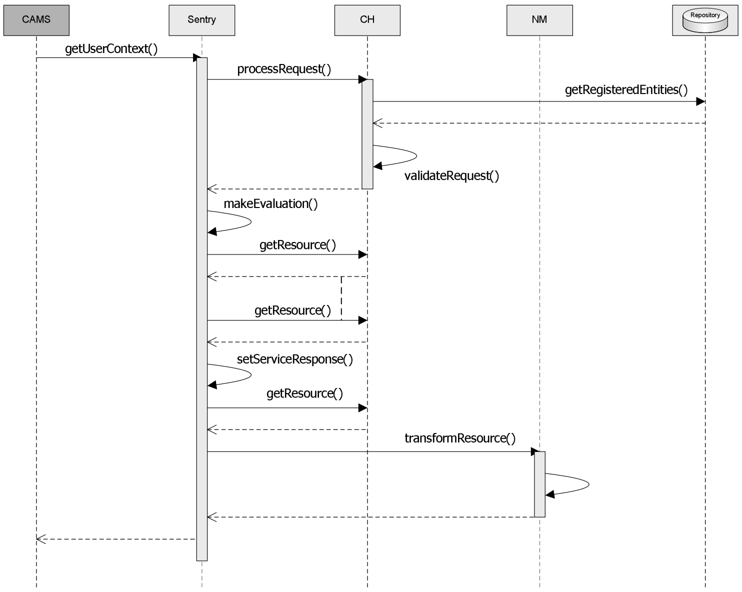

Figure 7: Interaction Sequence in the UCPF

5.1.1 Sentry

Sentry is the major architectural building block of the UCPF. It is build around the core elements of a policy distribution architecture as defined in the RFC2753 acting as Policy Decision Point (PDP), and Policy Enforcement Point (PET). It meets our requirement of providing *Privacy-aware Access Control*. The communication between a CAMS and the UCPF services is realized by the Sentry's PEP component implementing an Interceptor pattern on messages passing through the system. Figure 8 shows the template used to create the request- response-messages exchanged between a CAMS and a Sentry instance. Once the PEP gets a request message and after validating it, the message is forwarded to the PDP. When the PDP gets an evaluation message from the PEP, it imports its user policies into the rule engine and triggers its execution. The policy language and the policy evaluation algorithm used are presented in [AZV07] . Once the process finalizes the PEP compiles the message to be returned to the service.

5.1.2 Context Handler

The Context Handler (CH) provides access control to the system repositories and acts as a mediator between the Sentry and external context collectors. It also manages the semantic models used to represent policies and context data in the system. The CH has two main elements: the Policy Administration Point (PAP), which basically is used to

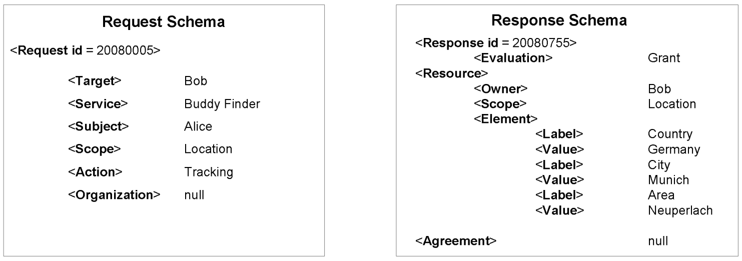

Figure 8: Request and Response Message Template

add, edit and delete policies from the repository, and the Data Handler (DH) responsible for the identification of external sources of context and make the UCPF *Context Awareness*. The DH also handles the access to the data repositories, e.g. to validate a request.

In our work, we mostly focus on interactions between the UCPF, as a Trusted Privacy Manager, and data consumer entities. In our model specification we just outlined how interactions with data collectors should take place to maintain the required level of personal privacy. However, we need to assume that there exists a context broker, e.g. as the one described in [AME07], able to gather context from existing data collectors and from which the DH can obtain context data. How that context broker is implemented and how it gathers data from different data collectors (as context providers) is considered to be out of the scope of this article.

The ideal situation is to extend the implementation of the UCPF to include a *Trusted Context Broker* (TCB) as part of its service architecture, though. It only should manage context data of the users of a single Sentry instance and be only accessible by the CH. The TCB is supposed to collect information from context providers, as a mobile operator, a GPS device or from a calendar application, and guarantee that the flow of information is unidirectional: Context Provider → TCB → CH . This assumption is based on the OMA Privacy Requirements [OMA07], which state that mobile operators must provide tools for specifying to which parties personal information may be forwarded. We suppose that all data collectors can be configured to only forward information to a TCB.

5.1.3 Noise Module

The Noise Module (NM) is a modular functional component that injects additional information into the policy matching mechanism for altering the data. Currently are implemented different levels of obfuscation, named Transformations, within the Local Transformation Point (LTP) and a Virtual Context generator within the White Lying Generator (WLG).

We define Transformations as any process that the user, as target, may define on a specific piece of context information to decrease its accuracy or granularity. The LTP is basically a set of different processes (Transformations), which are invoked by the Sentry before releasing the data. They are used to implement the requirement of allowing *Disclosure Level Control*. For instance, location information disclosed with accuracy equal to "area level", what means that information about the street and building, should be hidden from the recipient.

Also, we have developed a novel mechanism for privacy protection, which addresses the question: How can people temporarily "disconnect" from a "always-on" pervasive computing environment in a plausible way? Users can disclose a Virtual Context (as "white lie") instead of the real data. A Virtual Context is generated by the WLG based on a set of parameters and a visibility algorithm. Consequentially, we allow users to hide their location, their activity, etc., from others, without simply denying the access. More details are given in [WLI07].

5.2 Providing Service Usability

Service Usability is the second principle that guided the design of our framework. It is related with how interactions, within the protected interaction chain of Figure 6, are modeled. Service Usability involves the implementation of four of our requirements for personal privacy enforcement, namely: *Usage Control*, *Cooperative Privacy Protection*, *Fairness* and *Flexibility*. The purpose of implementing this principle is fourfold: i) to specify how interactions between the UCPF and a CAMS take place; ii) to define how collaborations are established between parties; iii) to enable the enforcement of user privacy preferences on secondary use by enterprises; iv) to make communications "fair".

In this section we first introduce the Sentry Registry component, key element to enable interactions between Sentry instances and CAMS. After that, we illustrate how functionalities are distributed by introducing our service deployment model.

5.2.1 Sentry Registry

The Sentry Registry is the only component that is not co-located with the rest of the elements in the gateway. This component is shared among Sentries and is hosted in the Internet. It tracks the availability of people's context and provides the pointer to the appropriate Sentry service instance. Services, organizations and Sentries need to be registered with the UCPF before starting any context request, which is done in the Sentry Registry. The Sentry Registry has an important role in making interactions possible between CAMS and Sentries, and also between different Sentry instances, providing *Flexibility* to deal with distributed and changing entities. Therefore, the main functionalities of the Sentry Registry component are:

Figure 9: Template Registration Request and Response Message

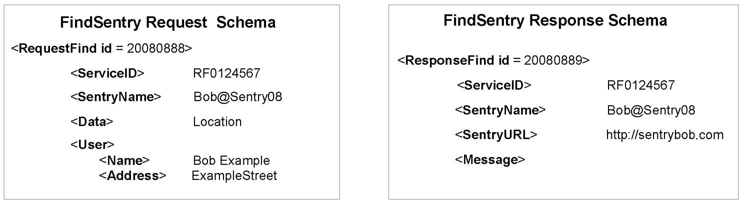

Figure 10: Template Find Sentry Request and Response Message

– Sentry Publisher: In order to make context data of the UCPF's users available to third-parties (CAMS and external Sentries), the Sentry Registry provides an interface to publish instances of Sentries detailing its users, context available and web service URL.

– Service Consumer Register: The Sentry Registry exposes an interface, which is used by external services to register themselves. Upon registration, an enterprise should disclose its privacy policy (e.g. with P3P). The registration is successful if the purpose, the data collected and the type of service are in the range of allowed values and the enterprise policy system is able to enforce obligations.

– Organization Register: We consider organizations as entities with a contractual relationship with one or more users of the UCPF. An organization may require, based on a contract agreed-on by a user, to constraint the disclosure of its clients' context data (e.g. employees). The Sentry Registry provides a web service interface to allow enterprises to define Organization Policies for UCPF's users.

– Sentry Discovery Service: The Sentry Register offers the possibility of discovering the availability of people's context. In response to a find request message, it returns the URL of the appropriate Sentry service instance. The Sentry Registry supports two types of queries, either by Sentry Name (`Bob@Sentry08`) which identifies the Sentry instance and its user, or by User, detailing his name and address, as is shown in the find-request message of Figure 10.

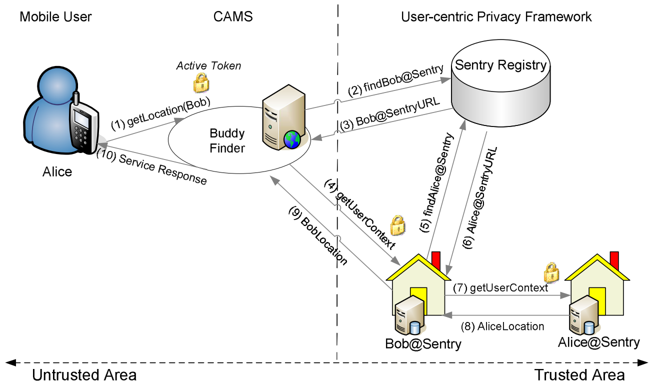

Figure 11: Deployment Model

Once an entity is registered, the messages exchange between Sentry Registry and Sentry are encrypted, for message confidentiality and to avoid identity theft. During the registration process each entity receives a UUID, used to identify them uniquely within the UCPF. The exchange of keys, used to encrypt and decrypt messages, must take place during registration, see Figure 9. The request-message includes the public key used by the Sentry Registry to encrypt the response-message. It contains the secret key to be used to encrypt and decrypt subsequent messages. We use asymmetric-key cryptography only to facilitate the distribution of secret keys. The use of symmetric cryptography in the rest of interactions provides better performance during the process of writing and reading messages.

5.2.2 Service Deployment Model

The service deployment model of Figure 11 shows how the different parts are distributed. In the "trusted area" are the Sentry Registry, located on the Internet and the instances of Sentries, `Bob@Sentry08` and `Alice@Sentry02`, which we assume are deployed on a gateway, e.g. in a smart home. A Sentry is deployed together with at least the following services: Context Handler, Noise Module, and Trusted Context Broker, as being described in the previous section. In the "untrusted area" are CAMS and Mobile Users.

In order to better explain the interactions taking place, steps 1 to 10 in Figure 11, we use the example of a "Buddy Finder" service. Alice first starts the "Buddy Finder" application on her mobile device. The "Buddy Finder" service then gets a request from Alice to compile the location of all the members on her contact list, among them also Bob (step 1). We suppose that Bob has set up his Sentry with a rule to allow the access from his group Friends to his location if they are located in the same city. When the "Buddy Finder" gets a request to obtain the location of Bob from Alice (one of the members of the group Friends) it needs to know where to find Bob's Location. Thus, it sends a query to the Sentry Registry to get `Bob@Sentry08` URL (steps 2 and 3), and subsequently is able to request Bob's location from the matching Sentry instance (step 4). As mentioned, Bob's rule includes a restriction on the location of Alice, what means that `Bob@Sentry08` needs to request its CH for Alice's location. It could happen that the CH does not know `Alice@Sentry02` and needs to get its URL from the Sentry Registry (steps 5 and 6). Once `Bob@Sentry08` gets Alice location, steps 7 and 8, the evaluation process of the "Buddy Finder" request finishes and the "Buddy Finder" finally can access the location of Bob (step 9), assuming that all the constraints are fulfilled, and sends the response with Bob location to Alice (step 10).

The addition of the UCPF in the protected interaction chain modifies the way that Mobile Users interact with CAMS. One of the implications of our model is that services cannot expect to directly get access to context data within a user request. For instance, in case of a "Restaurant Finder" service it is common to find that the *getRestaurantList()* call contains already the location of the requester (Bob), in our schema additional interactions are needed in order to get Bob's location directly from his Sentry. The advantage of this model is that it allows adding, e.g, a Transformation to reduce the granularity of the information to "area level" and also to negotiate an agreement on secondary use of the disclosed location information; or to make Bob aware of the number of disclosures made per service. On the other hand, services need to be registered with the Sentry Registry. The registration process consists of the evaluation of the service privacy policy, which should be publicly available.

5.2.3 Active vs. Passive Interactions with the UCPF

For Service Usability reasons and in order to make communication "fair", active and passive users of a service (acting as data consumer) must be distinguishable. In the active case, the user is actively using the service and some action from the service is requested, e.g. where to find the closest-by Italian restaurant, or where a friend is located. Here personal data is disclosed as "payment" for the service. On the other hand, if the user is passive, as is the case of Bob in the example of the "Buddy Finder", the user gets a data request without using the service.

An important feature of the UCPF is its ability to distinguish between three different classes of interactions (active, passive and binding) and to select the rules to be evaluated, accordingly, thereby meeting our *Fairness requirement*. Making communication "fair" involves to be aware of the context in which a service request was made. We want to avoid denying the access to user data when he, in the active role, has previously requested the service, or when he, as a "Target", has a binding agreement with the "Recipient", in the role of an organization, that obliges him to disclose his data.

We classify interactions from the point of view of the "Target" if the "Target", owner of the data requested, is an active user the interaction is called an *active interaction*. In those interactions, where the "Target" is a passive user, interactions are classified as *passive interactions*. An interaction is passive if a user receives a request from a CAMS or an external Sentry without previously requesting the service, e.g. when a colleague asks the "Friend Finder" application where a user currently is located, the disclosure of his location does not necessarily involve that the user gets any benefit in return.

Within passive interactions there exists a case that requires special consideration. When interactions are regulated with a binding contract between a user and an organization that requires from the user to disclose his data under certain circumstances. For instance, a home-care nurse might be required to disclose her location to her employer if her activity state is "working" for better organizing her daily calendar. We named these interactions *binding interaction*.

The UCPF supports three different types of interactions: active-, passive- and binding interactions. For each of those the system provides different functionality. In active interactions the UCPF discloses always the information requested, allowing users only to adjust the quality level (accuracy) of the data with a Transformation. In passive interactions the default configuration of the system is to deny all accesses to all the requesters. If a user wants to give a positive permission to one of the subscribed services or subjects, he needs to add a new rule including conditions under which such access should be granted. For binding interactions, the system allows the integration of rules created by an organization with the user policies. Thus, only in case of passive interactions a service could get a denied access to the data.

5.2.4 Active Token

The nature of a user request to any CAMS is independent of the privacy framework. Thus, the UCPF cannot distinguish whether a user is active or a passive one. We introduced the concept of an *active token* to indicate that a user has an active role in the ongoing interaction. An active token is a set of data used to proof that the user has originally requested the service. This token includes always the following fields: the user and service identifier (generated by the Sentry Registry), a timeout to allow tracking actions for a limited period of time, and the resource type.

An active token is generated and included in each request sent by a user of the UCPF to a CAMS. This means to assume that the service interface is modified to add an encrypted security token (e.g.WS-Security) and that it is forwarded in a `getUserContextRequest()` message to the appropriate Sentry. There, and before starting the actual evaluation of the request the token is decrypted.

When a Sentry gets an active token, the token can be addressed to that Sentry itself or a different Sentry. The addressee of the token is identified with a plain text ID. Only if a Sentry is the final receptor, it can decrypt it. We used asymmetric cryptography to authenticate the user@sentry and avoid that the token could be modified or forged.

In Figure 11 an active token is received by the "Friend-Finder" service and forwarded to `Bob@Sentry08`. But, as it was generated by Alice, it cannot be decrypted in

Bob@Sentry08. The request is evaluated in the context of a passive interaction them. The active token is forwarded to Alice@Sentry02 by the Context Handler whiting a new getUserContextRequest(), send to get Alice's location. In Alice@-Sentry02 it is finally decrypted and the location of Alice is revealed to Bob@-Sentry08. Here the request is evaluated in the context of an active interaction.

5.2.5 Usage Control and Collaboration

We introduced obligations in the UCPF to address two of our requirements, *Usage Control* and *Cooperative Privacy Protection*. Obligations are used to create automatic bindings with a CAMS and ensure that data protection requirements are adhered to. Their main functionalities are: i) to specify actions that should be performed by a service, acting as the recipient of a user data; ii) obligations are used to automatically exchange users' preferences on secondary use with enterprises; iii) to enable the exchange of notifications between Sentry instances and CAMS services; iii) to address prevention and avoidance at secondary use and to enable detection during collection, access and secondary use.

We created a set of predefined obligations and classified into system obligations and negotiable obligations. Obligations play a key role in the establishment of the privacy enforcement model introduced in Section 3, thus not all obligations are negotiable. For instance, the obligation "Data MUST NOT be disclosed to any third-party entity", is used to avoid direct disclosures between data collectors and data consumers. System obligations are agreed on during the registration process, with the Sentry Registry, without negotiation. They control disclosures to third-party services, monitor changes on the service privacy policy and enable the access to collected data and data logs. They reflect current privacy legislation and must be established beforehand of any commercial transaction with a service.

Negotiable obligations represent those constraints that a user may impose on an enterprise service at disclosure time. In our definition, they have two aspects: first, it is a second-class entity subject to the enforcement of a rule by the Sentry. And second, when an rule evaluation reaches the Sentry's PEP and it contains obligations, these obligations are not enforced but activate a negotiation protocol. Then, obligations are first-class entities used to exchange personal privacy preferences with an enterprise. Negotiable obligations are used to control the information disclosed among users of a particular CAMS, authorized purposes, number of accesses and retention time of user's data.

In the representation of obligations basically we follow the schema adopted by the Obligation Management System of Casassa Mont to facilitate collaboration with the enterprise privacy system. Obligations in the UCPF are XML documents with *Event*, *Action*, *Notification* and *Metadata* elements. Figure 12 shows the template followed together with an XML instance. An obligation is activated at the moment that its Event occurs. Once an obligation is activated, and its event parameters are true, its execution has

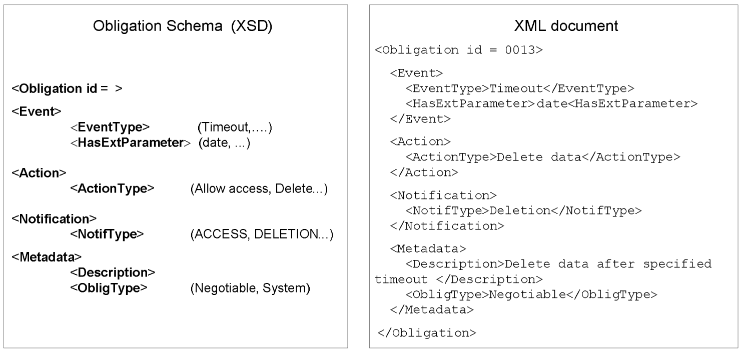

Figure 12: Obligation Template and XML Example

two effects: i) an action must be performed, specified with the property `ActionType`, ii) a notification must be sent to a Sentry instance. For that, the notification tag was added in the definition of the UCPF obligations. The type of notification that needs to be compiled is identified by the `NotificationType` property. Thus, the enforcement of an obligation in our model follows the sequence Event-Action-Notification, instead of an Event-Action used by previous approaches.

We assume in our model that the set of predefined obligations are common and known for all data collectors and data consumers; they do not need to be exchanged in each negotiation. We wrapped obligations in an agreement document, shown in Figure 14, where only the parameters of such obligations are exchanged. The agreement negotiation protocol defined in our model starts after the policy evaluation of a service request within a Sentry instance concludes. If the rule effect compiled contains obligations, the PEP queries the OM for an agreement over the pending obligations, step 1 in Figure 13. Then, if there is not a previous agreement, the OM returns the agreement to be negotiated, step 2. The Sentry launches the negotiation, steps 3 to 14, which is repeated at most for three rounds.

However, in many cases such agreements cannot be checked and thus to confirm the compliance with the obligation is almost impossible. The concept of non-observable obligations is described in [HBP05]. Hilty et al. suggest that a possible solution is the use of non-technical means, such as audits or legislation. We additionally propose the idea of employing observable bindings. This is realized by introducing a notification mechanism together with an agreement negotiation protocol, based on the exchange of messages about the whereabouts and use of data. The enforcement of obligations and the exchange of notifications provide the UCPF with tools to meet the requirements of *Usage Control* and *Cooperative Privacy Protection*.

The Obligation Manager is the only component that deals with obligations, agreements and notifications. It consists of the Obligation Handler and the Obligation Moni-

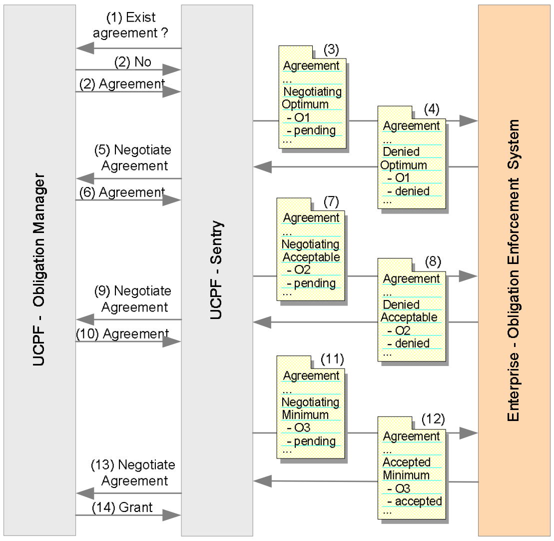

Figure 13: Agreement Negotiation Protocol Sketch

tor elements. The first is in charge of creating agreements and the negotiation of selected obligations. The Obligation Monitor monitors the state of obligations based on notifications and data logs. It also compiles notifications to be sent by the Sentry to the service, which can just confirm a received notification, request a data log, or inform of an occurred violation including sanctions to be carried out, e.g. unregistering the service. More information about the management of obligations in the UCPF can be found in [AZF08].

5.3 Providing User Manageability

A main goal of the UCPF is to allow non-expert users to specify their privacy criteria in an easy-to-use fashion by applications like our Privacy Manager Interface. This includes the definition of constrains on data collection, the desired obfuscation level and obligations to be negotiated, as well as providing interfaces for awareness and feedback. The core idea of our third design principle, User Manageability, is then to allow common users to manage privacy. But due to the lack of general knowledge of individuals of how to deal with privacy issues and the intrinsic complexity of semantic policy languages, the task of designing appropriate user interfaces is not trivial.

We analyzed how users can understand and thereby manage the privacy concepts implemented in our framework. As a first step, we addressed the following issues: How can a user understand the concept of privacy policies for passive and active interactions? How can a user specify restrictions on context data of him or even on the recipient context? How may a user still apply plausible deniability in pervasive computing? How can

Agreement Schema (XSD)

```
<Agreement id = 20080223>
        <Entity 1>          UCPF/00000555
        <Entity 2>          Buddy Finder
        <Priority>          (Mandatory, Middle, Optional, Request)
<Resource>
        <Owner>             Bob
        <Scope>             Location
<URL Notifications>         Bob@Sentry
<Timestamp>                 23.02.2008 11:00
<Negotiation>
        <State>             (Negotiating, Denied, Accepted)
        <Round>             (Optimum, Accetable, Minimum)
<List Obligations>
        <Obligation id = 008>
          <State>             (Pending, Rejected, Accepted)
          <Event Parameter>
              <Times>         5
              <Recipient>     Alice
        <Obligation id = 010>
            ...
<If Violated>
        <Sanctions>
```

Notification Schema (XSD)

```
<Notification id = 20080555 >
        <to >               00000555
        <from>              Buddy Finder
<Resource>
        <Owner>             Bob
        <Scope>             Location
        <Date>              23.02.2008 11:00
<Agrgemment ref>            20080223
        <Obligation Id>     008
<Notification Type>         Granted Disclosure
        <Recipient>         Alice
        <from>              23.02.2008
        <to>                25.02.2008
        <Times>             5
--------------------------------------------------------------
//only in notifications UCPF – Service
<Violation>
        <EnforcedSanction>
        ...
```

Figure 14: Agreement and Notification Template

he deal with organization policies? How can a user negotiate obligations on secondary use or set up the granularity level of information to be disclosed? Based on our analysis we designed in a second step a set of applications, which constitute our current version of the Privacy Manager Interface. The third step is to test our user interface, the Privacy Manager Interface, with a focus group of users and evaluate this against our initial questions.

We have developed the Privacy Manager Interface based on simple window-based control elements such as list, buttons, etc. Our goal was not to develop a new graphical environment to manage privacy. In [BRS05], studies showed that most users still prefer to use conventional interfaces, 24 out of 34 participants selected a "traditional" browser interface rather than a more sophisticated "Virtual City". We believe that users need to learn how to manage privacy prior to introducing "fancy" interfaces. It is more important to provide easy-to-use concepts for describing and managing privacy together. Our framework, we describe here, obviously can be accessed by all kinds of different GUIs (e.g. such Virtual City) later.

5.4 Privacy Manager Interface

The Privacy Manager interface incorporates a set of application parts designed especially to meet the requirements of *User Friendliness* and *Privacy Awareness*. Our first prototype allows to: i) customize permissions for the disclosure of users' personal data, ii) control active and passive interactions with services, iii) define obligations to be negotiated on the usage of the data upon transmission, iv) be aware of privacy related issues such as granted and denied permissions, v) apply alternative privacy mechanisms to access control, as white lying and obfuscation, vi) adhere to enterprise privacy poli-

Figure 15: Access and Home Windows

cies based on a contractual relationship with an enterprise or organization.

The access screen, shown on the left hand side in Figure 15, is the first window that users find after starting the Privacy Manager Interface. This window is used to login but also to register a new user of the system. Once a user is registered, he or she can access the offered applications, by just introducing "username" and "password". The button "login" takes a user to the next window, the "Home" window. The "Home" window, shown on the right hand side in Figure 15, displays six different options to manage personal privacy. Two of them, the "White Lying" and "Organization Policies" are already designed but under development at the moment of writing this article. The other four: "Customize your Permissions", "Customize your Services", "Privacy State", and "Obligations" are being tested by targeted users. Each of the applications is presented including a picture and a brief explanation to help users to understand the functionality of the application.

Customizing your Permissions. The first selectable option, depicted at the upper left corner of the home screen, is the application that allows users to manage positive permissions. A special feature of the "Customize your Permissions" application, shown in Figure 16, is that it provides a default configuration for all its users, even if a user does not introduce any rule at all, the user is still protected, the system returns always a "deny" value. A user only needs to create positive permissions to deal with passive interactions. With this application a user can add new permissions, by using a set of four different constraints: on the recipient, on the user context, on the recipient context, and on the accuracy of the data. A user can also check, update and delete those rules previously created. It provides also a special tool to manage contact groups, which can be used to group individuals into roles.

Customizing your Services. The second option to create personal policies is the application "Customize your Services", depicted at the upper right corner of the home

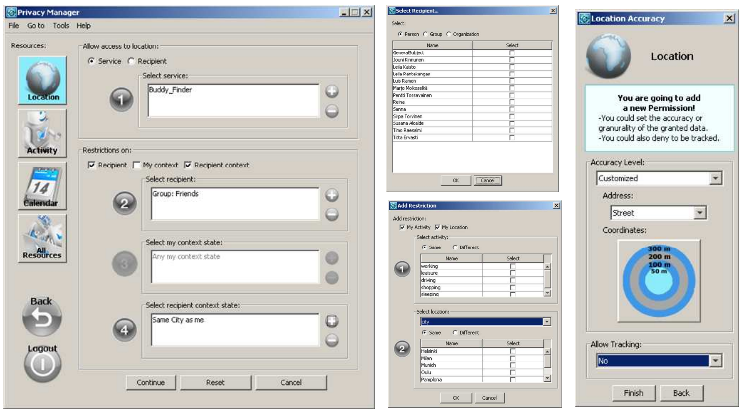

Figure 16: Screen Shots of the Customize your Permissions Application

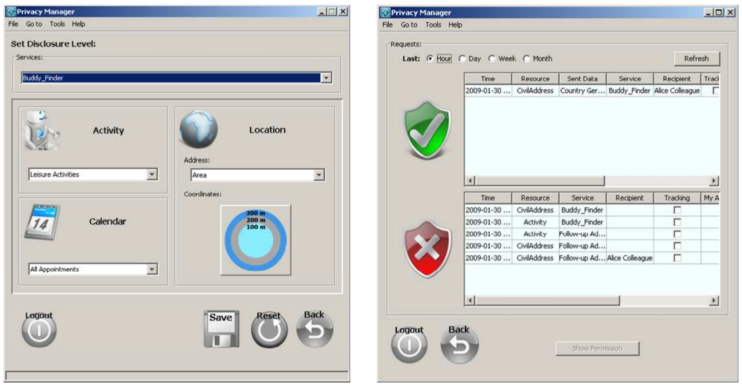

Figure 17: Customize Services and Privacy Status Application Screens

screen. This application is used only for active interactions. Here users just need to configure the granularity of each resource for each of the registered services, see the screen shown on the right hand side in Figure 17. Theses rules do not include other type of constraints. As a result, if one of the rules is enforced, the service always gets a positive permission. Since, the rule only controls the applicable transformation.

Privacy Status Application. The application “Privacy Status” was created to meet the system requirement of Awareness of users’ privacy status. The application should display all the information related to privacy issues. In the current version (left hand side on Figure 17), it lists granted and denied permissions sorted by date, and allows to check enforced rules. In the future, it will provide functionalities to track existing agreements on obligation sets, check the state of unfulfilled obligations, and monitor notifications sent to and received from a service. We would like also to extend this application by

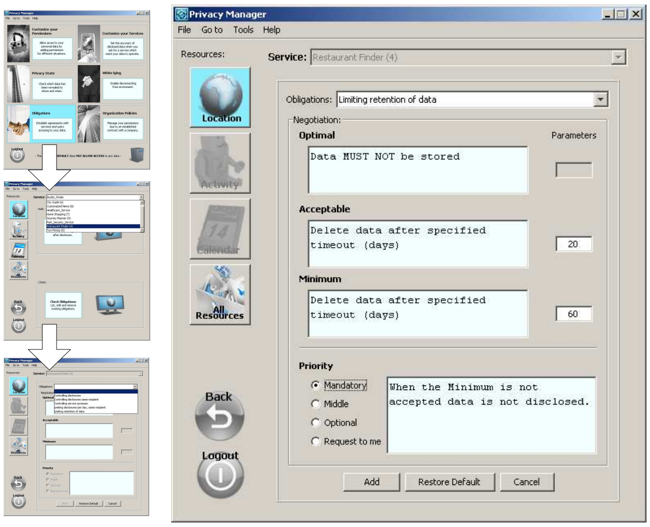

Figure 18: Obligation Application

incorporating alarms on potential privacy risk and check enterprises privacy policies.

Obligation Application. The group of negotiable obligations, as was mentioned, should be selected by users to specify their preferences according usage of data. For that we have included the application Obligations, shown in Figure 18. In that application, negotiable obligations are grouped in sets of three obligations, optimal, acceptable and minimum. One for each of the three negotiation rounds allowed. Per each set of obligations a user can also add a priority value to set up the action to be performed in case that the minimum obligation is not accepted.

6 Conclusions and Outlook

In this article, we presented a novel architecture that extends privacy control in a substantial matter and complements enterprise privacy with the incorporation of personal privacy. We elaborated the set of requirements for personal privacy enforcement aimed at maintaining the balance between *Privacy Protection*, *Service Usability*, and User *Manageability*. Based on these requirements, we designed and implemented a *Trusted Privacy Manager*, called UCPF, to control and monitor the context disclosure of a user to third parties during collection, access and secondary use. Furthermore, the UCPF provides a set of graphic tools to enable personal privacy management for non-expert users.

Also, we were showcasing some typical sample applications in the field of mobile services as they may be common in pervasive environments. Obviously, our scope is not limited to such applications but can easily be extended to other services as well because of the flexible and extensible nature of our framework, together with the use of a service-oriented approach for deployment of new services, rules and sets of obligations.

As part of our ongoing research, we were integrating the UCPF already into the demonstrator of the IST project CONNECT where mobile healthcare personnel are managing their privacy settings using our system model. Also current work includes the creation of demonstrators and questionnaires for deploying privacy-enabled tools at the Public University of Navarra for showcasing applications in the context of services found at a University, such like a blackboard application for finding homework partners.

Our future work includes to apply our framework to privacy-sensitive applications in the context of patient-monitoring and -supervision, as being found in integrated hospital IT landscapes. There we are about to start work in the context of the ITEA2 project AIMES.

Altogether we are convinced that especially the dynamic and flexible nature together with the explicit user-focus of our work is a great contribution to the field of privacy protection in pervasive computing.

References

[AHK07] D. Anthony, T. Henderson, D. Kotz, Privacy in Location-Aware Computing Environments, IEEE Pervasive Computing, vol. 6, no. 4, Oct-Dec 2007.

[And05] Anne Anderson, A Comparison of Two Privacy Policy Languages: EPAL and XACML, Sun Microsystems Laboratories, Technical Report TR-2005-147, 2005.

[APP02] M. Langheinrich, L. Cranor, M. Marchiori. Appel: A P3P Preference Exchange Language. W3C Working Draft, April 2002.

[AME07] S. Alcalde Bagüés, J. Mitic, E. Emberger. The CONNECT Platform: An Architecture for Context-Aware Privacy in Pervasive Environments. IEEE CS, Workshop on Secure and Multimodal Pervasive Environments 2007.

[AZF07] S. Alcalde Bagüés, A. Zeidler, C. Fernandez Valdivielso, I. R. Matias. Sentry@Home - Leveraging the Smart Home for Privacy in Pervasive Computing. International Journal of Smart Home Vol. 1 No. 2, 2007.

[AZF08] S. Alcalde Bagüés et al. Obligations: Building a Bridge between Personal and Enterprise Privacy in Pervasive Computing. In Proceedings of the 6th International Conference Trust, Privacy, Security in Digital Business, LNCS 5185, 2008.

[AZV07] Towards Personal Privacy Control, S. Alcalde Bagüés, A. Zeidler, C. Fernandez Valdivielso andI. R. Matias, On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, LNCS, vol. 4806, 2007.

[BRS05] Mike Bergmann, Martin Rost and John Sören Pettersson, Exploring the Feasibility of a Spatial User Interface Paradigm for Privacy-Enhancing Technology, Advances in Information Systems Development, 2005, pag. 437-448,Springer.

[CAT06] M. Casassa Mont, R. Thyne. A Systemic Approach to Automate Privacy Policy Enforcement in Enterprises. In Proceedings of the 6th Workshop on Privacy Enhancing Technologies. LNCS, Springer-Verlag, 2006.

[Cue02] Jorge R. Cuellar,Geographic Location in the Internet, Location Information Privacy, Kluwer Academic Publishers, pag. 179–212, 2002.

[Dek07] Marnix Dekker, Sandro Etalle, and Jerry den Hartog. Security, Privacy and Trust in Modern Data Management. Chapter 25: Privacy Policies. Springer-Verlag, 2007.

[EHG07] A. Esquivel, P. A. Haya, M. García-Herranz, X. Alamán, Managing Pervasive Environment Privacy Using the fair trade Metaphor. In proceedings of the OTM 2007 Workshops, LNCS vol. 4806, 2007.

[EPA03] P. Ashley and S. Hada and G. Karjoth and C. Powers and M. Schunter, Enterprise Privacy Authorization Language(EPAL 1.2) Specification, `http://www.zurich.ibm.com/security/enterprise-privacy/epal/`, November, 2003.

[HBP05] M. Hilty et al. On Obligations. In Proceedings of the 10th European Symposium on Research in Computer Security, 2005.

[HoL04] J. I. Hong, J. A. Landay. An architecture for privacy-sensitive ubiquitous computing. In the ACM proceedings of the 2nd international conference on Mobile systems, applications, and services, pg. 177-189, 2004.

[JHL02] Jiang, X., Hong, J. I., and Landay, J. A. Approximate Information Flows: Socially-based Modeling of Privacy in Ubiquitous Computing. In proceedings of the 4th International Conference on Ubiquitous Computing, 2002.

[KS02b] Günter Karjoth and Matthias Schunter, A Privacy Policy Model for Enterprises, CSFW '02: Proceedings of the 15th IEEE workshop on Computer Security Foundations, 2002, IEEE Computer Society.

[KFJ03] L. Kagal, T. Finin, A. Joshi. A policy language for a pervasive computing environment. In Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks, September 2003.

[Knopf] Knopflerfish Open Source OSGi, `http://www.knopflerfish.org/`

[Lan01] Marc Langheinrich, Privacy by Design – Principles of Privacy-Aware Ubiquitous Systems, Ubicomp 2001 Proceedings, LNCS, vol. 2201, pag. 273–291, 2001.

[Lan02] M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In Proceedings of the 4th International Conference on Ubiquitous Computing. LNCS No. 2498, Springer-Verlag, September 2002.

[LMD03] Lederer, S., Mankoff, J., Dey, A. K., Beckmann, C. P. Managing personal information disclosure in ubiquitous computing environments. Technical Report CB-CSD-03-1257, University of California, Berkeley, 2003.

[LUH79] Niklas Luhmann, Trust and Power, Chichester: Wiley, 1979.

[MCC07] U. M. Mbanaso, G. S. Cooper, D. W. Chadwick, A. Anderson. Obligations for privacy and confidentiality in distributed transactions. In EUC Workshops, pages 6981, 2007.

[MFD03] G. Myles, A. Friday, N. Davies. Preserving privacy in environments with location-based applications. IEEE Pervasive Computing, 2(1):5664, 2003.

[OAB07] Paul N. Otto, Annie I. Antón, David L. Baumer, The ChoicePoint Dilemma: How Data Brokers Should Handle the Privacy of Personal Information, IEEE Security and Privacy, vol. 5, no. 5, Sept./Oct. 2007.

[OECD] Daniel E. O'Leary, Some Privacy Issues in Knowledge Discovery, The OECD Personal Privacy Guidelines, IEEE Expert: Intelligent Systems and Their Applications, vol. 10, no. 2, Apr. 1995.

[OMA07] Open Mobile Alliance OMA, Privacy Requirements for Mobile Services, OMA-RD-Privacy-V1_0_1-20070807-A, August, 2007.

[P3P02] L. Cranor, M. Langheinrich, M. Marchiori, J. Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification. W3C Recommendation, Apr. 2002.

[PaS04] J. Park, R. Sandhu. The UCONABC usage control model. ACM Transactions. Inf. Syst. Secur., 7(1):128174, 2004.

[SSA06] S. Sackmann, J. Strüker, R. Accorsi, Personalization in privacy-aware highly dynamic systems. Communications of the ACM, vol. 49, no. 9, 2006.

[STM07] H. Schulzrinne, H. Tschofenig, J. Morris, J. Cuellar,J. Polk. Geolocation Policy: A Document Format for Expressing Privacy Preferences for Location Information. IETF Internet Draft. January 2007.

[WLI07] S. Alcalde Bagüés, A. Zeidler, C. Fernandez Valdivielso, I. R. Matias. Disappearing for a while - using White Lies in pervasive computing. In Proceedings of the 2007 ACM workshop on Privacy in Electronic Society 2007.

[XACML] OASIS standard,eXtensible Access Control Markup Language. Version 2, February, 2005.