A formal privacy management framework

Daniel Le Métayer
INRIA

Abstract Privacy is a complex issue which cannot be handled by exclusively technical means. The work described in this paper results from a multidisciplinary project involving lawyers and computer scientists with the double goal to (1) reconsider the fundamental values motivating privacy protection and (2) study the conditions for a better protection of these values by a combination of legal and technical means. One of these conditions is to provide to the individuals effective ways to convey their consent to the disclosure of their personal data. This paper focuses on the formal framework proposed in the project to deliver this consent through software agents.

To appear in: Proceedings of FAST'2008, Formal Aspects of Security and Trust, Springer Verlag, LNCS

1 Context and Motivations

In the same way as the growing use of photography at the end of the 19th century prompted Warren and Brandeis seminal paper [31], the changes imposed nowadays by information and communication technologies require a deep reflection on the fundamental values underlying privacy and the best way to achieve their protection [15, 27]. Furthermore a multidisciplinary approach is necessary to tackle this challenge because privacy can neither be apprehended nor guaranteed by exclusively legal or technical means. As a step in this direction, the collaborative project PRIAM[1] gathers lawyers and computer scientists with the goal of putting forward a common view of privacy for pervasive computing and effective (legal and technical) instruments to protect it.

Daniel Le Métayer, Inria Grenoble Rhône-Alpes, 655 venue de l'Europe, Montbonnot e-mail: Daniel.Le-Metayer@inrialpes.fr

[1] Privacy Issues in Ambient Intelligence

One of the greatest challenges posed by pervasive computing or ambient intelligence to privacy is the fact that communications and computations can occur without the user's notice ("invisibility principle"). Indeed, most legal instruments for privacy protection explicitly refer to the unambiguous consent of the person as one of the conditions for the collection of his/her personal data. But requiring that the user provides his consent before each single data communication would not only be ineffective in terms of privacy protection (or even counterproductive, as it already is on the Internet, because the harassed user would end up accepting all requests and relinquishing his privacy altogether), it would also defeat the very purpose of these systems. Possible ways to reconcile the principle of unambiguous consent and the essential features of ubiquitous computing have thus been central to the legal and technical studies conducted in the PRIAM project.

In this paper we start from the requirements and recommendations resulting from the legal study and focus on the technical aspects, more precisely on the definition of a formal framework for privacy management. The overall approach, which involves a natural language setting as well as informal and formal descriptions, is sketched in Section 2. The formal framework itself is defined in Sections 3, 4 and 5 which introduce respectively the language of events, compliance and global correctness. Section 6 is a review of related work and Section 7 draws some conclusions.

2 Approach

The fact that a person ("data subject" following the terminology of [28]) must provide his informed consent before his personal data may be collected (unless otherwise authorized by law) is the cornerstone of most data protection regulations [29]. For example, Article 7 of the EU Directive 95/46/EC [28] states that

> Personal data may be processed only if: (a) the data subject has unambiguously given his consent; or (b) processing is necessary for the performance of a contract to which the data subject is a party or in order to take steps at the request of the data subject prior to entering into a contract; or (c) processing is necessary for compliance with a legal obligation to which the controller is subject; or (d) processing is necessary in order to protect the vital interests of the data subject, ...

In addition, this consent must be *informed* in the sense that the entity collecting the data ("controller" following the terminology of [28])[2] must provide sufficient information to the data subject, including "the purposes of the processing for which the data are intended".

In situations such as pervasive computing where an action from the user before each disclosure of data is not practically feasible, the natural question for the computer scientist is then: why not using the technology itself to cure the problems caused by the technology? In other words, if privacy rights are jeopardized by the

[2] More precisely, [28] defines the controller as the legal entity which determines the purposes and means of the processing of personal data.

highest level of automation provided by pervasive computing, why not also increasing the level of automation on the side of the defense of these rights ? This idea leads to the notion of *Privacy Agent*, a dedicated software which would play the role of “representative” or “proxy” of the user and manage his personal data on his behalf [20, 21]. Not surprisingly, this possibility triggers a whole bunch of new questions from the legal side: to what extent should a consent delivered via a software agent be considered as legally valid? Are current regulations flexible enough to accept such kind of delegation to an automated system? If it is the case, what technical and legal constraints should be imposed on a software agent to be used as a valid representative of a subject? What would be the consequences of any error (bug, misunderstanding, etc.) in the process? The conclusions of our legal analysis of these issues are presented in [22]. The most important recommendations as far as the present paper is concerned are the following:

1. The technical framework should ensure, as much as possible, that the meaning and impact of the consent are defined without any ambiguity and properly understood by all the actors involved.
2. The actors involved are not only the data subject and the controller, but also the software agent providers and the Data Authority[3]. In particular, since software agents are not granted any legal personality (even though this issue is debated among lawyers [12, 10]), it is the software agent provider who should be liable for the correct implementation of the privacy policy of the data subject.
3. All actors should be held accountable for their actions and precise procedures should be put in place to ensure that liabilities can be established after the facts. Such procedures should be usable in a formal procedure in case of litigation[4].

In order to implement the above recommendations, the legal and technical framework put forward in the PRIAM project involves the following ingredients:

1. A restricted natural language (SIMPL: SIMple Privacy Language) used by data subjects and data controllers to express respectively their privacy requirements and commitments.
2. Specifications of a subject software agent and a controller software agent (“Subject Agent”, or “SA”, and “Controller Agent” or “CA” in the sequel). These specifications are mostly expressed in a formal framework, based on a trace semantics, complemented with informal requirements.
3. Link between SIMPL policies and software agent specifications.
4. Link between software agent specifications and their implementations.
5. Legal contracts between the Subject Agent provider (respectively the Controller Agent provider) and the data subject (respectively the controller) referring to the above items.

[3] In addition, other certification authorities may also be involved, e.g. to authentify or to certify software agents.

[4] As set forth in Article 23 of [28] the controller is liable for damages suffered by the data owner as a result of unlawful processing of personal data.

The PRIAM framework thus involves different languages (SIMPL, trace language, implementation language) which, we believe, is essential due to the variety of actors involved. The position taken in PRIAM is that, in order to reduce potential sources of ambiguities:

- The most appropriate language should be used for each purpose.
- Each of these languages should be kept minimal.
- The correspondences between these languages should be defined precisely.

These conditions are necessary to ensure that each actor has the proper understanding and that these understandings are consistent. For example, the aforementioned links should ensure that there is no gap between the wishes of a data subject (expressed through the SIMPL language) and the actual behaviour of his Subject Agent. In any circumstances, if a disagreement arises concerning the treatment of personal data by a controller (or the software agent acting on his behalf), then the proposed framework should make it possible to discover the origin of the problem and to identify the liable actor.

In this paper, we focus on the formal part of the specification and provide some hints on the other aspects.

3 Events Language

Before entering into the presentation of the events language used to define the semantics of software agents, it is necessary to start with a quick introduction to the SIMPL language. Let us call "disclosure policy" and "collection policy" the privacy policies defined by data subjects and controllers respectively. The following is an example disclosure policy statement in SIMPL:

> I consent to disclose data of category Cultural to a third party only if the aforementioned third party has provided the following pieces of information pursuant to this disclosure of data:
>
> - His identity and such identity belongs to Book Store.
> - His verification level and such verification level is at least 2.
> - His privacy policy with respect to the aforementioned category of data and such policy includes the following commitments :
>   - Use only this data for the following purpose(s): Order Processing.
>   - Delete this data within a delay of 1 month.
>   - Transfer this data always accompanied with the present privacy and only to third parties allowed to receive this data according the present privacy policy after commitment of such third party to respect this privacy policy provided I am previously informed of such disclosure and the identity of the third party.
>   - Ensure that any valid request from my side to access such data will be satisfied within a delay of 3 days.
>   - Ensure that any valid request from my side to delete such data will be satisfied within a delay of 3 days.
>   - Ensure that any valid request from my side to modify such data will be satisfied within a delay of 3 days.

Similarly, the controller can express privacy commitments of the form:

The management of data of category Book Order shall meet, if requested by their Sticky Policy, the following requirements:

- Use of the data shall be only for the following purpose(s): Order Processing.
- The data shall be deleted within a delay of 3 months after its collection.
- The data may be transferred or disclosed (i) always accompanied with its Sticky Policy; (ii) only in contexts allowed by this Sticky Policy; (iii) only to third parties allowed to receive the data according this Sticky Policy after commitment of such third party to comply with this Sticky Policy.
- Any Valid Request from the owner of the data to access the data will be satisfied within the delay of 1 week.

SIMPL has a slightly legal flavour because it is the language used to express policies as they are signed by individuals (subjects and controllers) but users can define their policies through a friendly interface which relieves them from the burden of writing the sentences by themselves. The above examples illustrate only some of the possibilities of the SIMPL language. A more complete account of the language is conveyed through the presentation of the semantics domains below.

Following the minimality principle stated in the previous section, we define the semantics of privacy agents (Subject Agents and Controller Agents) in terms of traces of events. In the following subsections, we introduce some of the most significant events for, successively, Subject Agents and Controller Agents.

3.1 Subject Agent Events

Subject Agents can communicate with Controller Agents and with the subject himself. By convention, events $E(Id_1, Id_2, \ldots)$ represent communications from Id_1 to Id_2:

- $DisclosureRequest(Id_1, Id_2, Category, Verification, Commitment)$ is a communication from a CA to a SA : the CA asks for the disclosure of information of category $Category$ of subject Id_2. Id_1 is the name of the controller, $Verification$ his verification level and $Commitment$ his commitments for the treatment of the requested data. The verification level can be seen as a trust level granted to the controller by a certification authority, which may come with a certificate from this authority. For this information to make sense for the subject, a standard (or widely accepted) ranking must be available[5]. The subject can also require that the certificate originates from specific authorities.
- $DataDisclosure(Id_1, Id_2, Category, Value, StickyPolicy)$ is the disclosure of data as a reply to the previous request. $Value$ is the requested value and $StickyPolicy$

[5] The ranking used in the PRIAM project includes level 1 (minimum level) which corresponds to controllers having committed to comply with automated auditor requests, level 2 which includes the same commitment for physical audits and level 3 which includes the certification of the Controller Agent.

its associated privacy policy. In our framework, a personal data should never be separated from its privacy policy. Id_1 is the name of the subject and Id_2 the name of the controller.
- $SDefineDisclosure(DisclosurePolicy, Id)$ is a communication from a subject to his SA : the subject defines a new disclosure policy and identity[6].

3.2 Controller Agent Events

In addition to the communications with Subject Agents introduced in the previous subsection, Controller Agents can interact with their controller (defining a new collection policy), with third parties (requesting the transfer of personal data collected by the controller) and with applications requesting access to the data. Applications represent accesses to the data which are local to the controller's site or device. The following are examples of Controller Agent events:

- $DefineCollection(CollectionPolicy)$ is a communication from the controller to the CA : the controller defines a new collection policy[7].
- $TransferRequest(Id_1, Id_2, Id_3, Category, Verification, Commitment)$ is a communication from the CA of name Id_1 to the CA of name Id_2. The controller Id_1 requests the transfer of data of category $Category$ pertaining to the subject of name Id_3. $Verification$ is the verification level of Id_1 and $Commitment$ his commitments.
- $TransferData(Id_1, Id_2, Id_3, Category, Value, StickyPolicy)$ is the reply to the previous request. $Value$ is the value of data $Category$ of the subject of name Id_3 and $StickyPolicy$ its sticky policy.

4 Compliance

The semantics of a software agent is defined in terms of compliant agent traces. An agent trace is a pair (E, S) with E a finite list of event values $E_1, \ldots, E_n$ and S a finite list of state values $S_1, \ldots, S_n$.

States are functions from variables to their domains. Any Subject Agent state includes at least the following variables:

- $MyData$: function of type $Categories \rightarrow Values$ representing the personal data of the subject.
- $MyDPolicy$: disclosure policy of the subject (belonging to $DisclosurePolicies$, as defined below).

[6] This feature allows the subject to use different identities (or pseudonyms) at different points of time.

[7] Note that, in contrast with subjects, controllers cannot change their identity. This limitation makes it easier to implement the accountability requirement set forth in Section 2.

- *MyIdentity*: identity of the subject.
- *MyTime*: local time of the subject.

It can also include context variables such as *MyLoc* (localization). Note that time is simply treated as a state variable here: the faithful implementation of a clock is typically a commitment which is left in the informal part of the specification.

Controller Agent states include at least the following variables:

- *MyImport*: function of type $(Identities \otimes Categories) \rightarrow (Times \otimes Values \otimes StickyPolicies)$ representing the personal data collected by the controller with their collection date and sticky policy.
- *MyCPolicy*: collection policy of the controller (belonging to *CollectionPolicies*, as defined below).
- *MyIdentity*: identity of the controller.
- *MyTime*: local time of the controller.
- *MyLevel*: verification level of the controller.

Similarly to Subject Agent states, Controller Agent states can also include context variables.

Events are tuples of values tagged by their event type. Events can either be internal or external. External events are events of the types introduced in the previous subsection. Internal events include other actions which can have an impact on the state of the agent. For the sake of simplicity (and without loss of generality), we consider only one type of internal event here: *internal*.

The parameters of external events take values in the following domains:

$$\begin{array}{l}
DisclosurePolicy \ : \ DisclosurePolicies \\
CollectionPolicy \ : \ CollectionPolicies \\
Id,\ Id_1,\ Id_2,\ Id_3 \ : \ Identities \\
Category \ : \ Categories \\
Value \ : \ Values \\
Application : Applications \\
Purpose : Purposes \\
Commitment : \mathrm{Commitments} \\
Verification : Verifications \\
StickyPolicy : StickyPolicies
\end{array}$$

The main domains are defined as follows[8]:

$$\begin{array}{l}
DisclosurePolicies \ = \ Categories \rightarrow StickyPolicies \\
CollectionPolicies \ = \ Categories \rightarrow Commitments \\
StickyPolicies = \{(X_1, X_2, X_3, X_4) \| \\
\qquad\qquad X_1 \in IdentityPolicies, \\
\qquad\qquad X_2 \in VerificationPolicies, \\
\qquad\qquad X_3 \in Commitments, \\
\qquad\qquad X_4 \in Contexts\} \\
IdentityPolicies = \{(X,Y) \parallel X = \nabla \ or \ X \subseteq Nat,\ Y = \nabla \ or \ Y \subseteq Authorities\}
\end{array}$$

[8] The other domains are pre-defined sets of basic values: for example, $Applications \ = \ Nat$.

$VerificationPolicies = \{(X,Y) \parallel X \in \{\nabla,1,2,3\}\,,\; Y = \nabla \textit{ or } Y \subseteq Authorities\}$
$Commitments = \{(X_1,X_2,X_3,X_4,X_5,X_6) \parallel$
$\quad X_1 \subseteq Purposes,$
$\quad X_2 \in Delays,$
$\quad X_3 \in \{\perp,\nabla,information,authorization\},$
$\quad X_4,\; X_5,\; X_6\; \in Delays\}$
$Delays \;=\; Nat_\nabla$
$Identities \;=\; Nat \otimes Certificates$
$Contexts \;=\; (Vars \rightarrow Bool)_\nabla$
$Certificates \;=\; (Nat \otimes Authorities)_\nabla$

We use the notation S_∇ to denote the set $S \bigcup \{\nabla\}$. The value ∇ represents the absence of constraint or commitment: for example if the deletion delay is equal to ∇ (X_2 in the definition of *Commitments*) for a given category in a disclosure policy, it means that no commitment is required from the controllers with respect to the deletion of data of that category; if the value of X in the definition of $IdentityPolicies$ is ∇, it means that no constraint is imposed on the identity of the controllers which are allowed to receive the data. A disclosure policy associates a sticky policy with each category of data. Sticky policies involve three constraints on the controllers (X_1, X_2 and X_3) and one constraint on the context (X_4). The first component of identity policies is the set of identities of controllers allowed to receive the data and the second component is the set of recognized certification authorities to certify this identity. The components of $Commitments$ represent respectively: the set of authorized purposes (X_1); the deletion delay (X_2); the commitment with respect to transfer of the data to a third party (X_3 where $\perp$ means no transfer right); and delays for complying with requests from the data subject (respectively X_4 for access requests, X_5 for deletion requests and X_6 for modification requests). Certificates contain the actual value of the certificate and the certification authority.

For the purpose of this paper, we use Definition 1 as the compliance property for Subject Agent traces, which is sufficient to convey the essence of the approach. The complete definitions of compliance include additional requirements to ensure, for example, that agents forward messages towards (and from) subjects and controllers.

Definition 1. A Subject Agent trace (E,S) is said to be compliant if the following conditions hold:

$\forall\, i,\; E_i = DataDisclosure(Id_1,Id_2,Ca,Va,Po) \;\Rightarrow$
$\quad \exists\, j < i,\; \exists\, Ve_2,\; \exists\, Co_2,\; E_j = DisclosureRequest(Id_2,Id_1,Ca,Ve_2,Co_2) \textit{ and}$
$\quad \forall\, k\; j < k < i,\; E_k \neq DataDisclosure(Id_1,Id_2,Ca,*,*) \textit{ and}$
$\quad S_i(MyIdentity) = Id_1 \textit{ and } S_i(MyData)(Ca) = Va \textit{ and}$
$\quad Po = S_i(MyDPolicy)(Ca) = (Id,Ve,Co_2,Cx) \textit{ and}$
$\quad Id_2 \triangleright Id \textit{ and } Ve_2 \triangleright Ve \textit{ and } S_i \triangleright Cx$
$\forall\, i, E_i = DefineDisclosure(Dp,Id) \;\Rightarrow$
$\quad S_i = S_{i-1}\,[MyDPolicy \mapsto Dp;\; MyIdentity \mapsto Id]$
$\forall\, i,\; E_i = Internal \;\Rightarrow$
$\quad S_i(MyDPolicy) = S_{i-1}(MyDPolicy) \textit{ and}$
$\quad S_i(MyIdentity) = S_{i-1}(MyIdentity)$
$\forall\, i,\; E_i = External \textit{ and } E_i \neq DefineDisclosure \;\Rightarrow S_i = S_{i-1}$

Definition 2. The "satisfies" relation $\triangleright$ is defined as follows for, respectively, identities, verification levels and states:

Let $Id = (Id_1, Cer_1)$, $Id' = (Id'_1, Au'_1)$, then $Id \triangleright Id'$ if and only if

$(Id'_1 = \nabla \; or \; Id_1 \in Id'_1) \; and \; (Au'_1 = \nabla \; or \; (Cer_1 = (*, Au_1) \; and \; Au_1 \in Au'_1))$

Let $Ve = (Ve_1, Cer_1)$, $Ve' = (Ve'_1, Au'_1)$, then $Ve \triangleright Ve'$ if and only if

$(Ve'_1 = \nabla \; or \; Ve_1 \geq Ve'_1) \; and \; (Au'_1 = \nabla \; or \; (Cer_1 = (*, Au_1) \; and \; Au_1 \in Au'_1))$

$S \triangleright Cx$ if and only if $Cx = \nabla \; or \; (\forall y \in Domain(Cx), \; S(y) \Rightarrow Cx(y))$

The most important rule for the compliance of Subject Agents is the rule defining the conditions for data disclosure. First a request for disclosure must have been received previously by the Subject Agent (and must not have been answered before). In addition, this request must come from an authorized controller for this category of data ($Id_2 \triangleright Id$), his verification level must be sufficient ($Ve_2 \triangleright Ve$) and the current state must satisfy the context requirement in the disclosure policy for this category of data ($S_i \triangleright Cx$). Last but not least, the controller must commit to the sticky policy for this category of data ($S_i(MyPolicy)(Ca) = (Id, Ve, Co_2, Cx)$ with Co_2 equal to the commitment in the *DisclosureRequest* event). Another important rule is the rule stating that the disclosure policy and identity must not be modified by internal events[9].

The compliance property for Controller Agents, which is not presented here for the sake of conciseness, characterizes honest behaviours of Controller Agents: for example, when requesting a data disclosure, the Controller Agent must provide its true identity, verification level and privacy policy for the category of data requested; the Controller Agent must ensure that collected data are not kept longer than permitted; it cannot modify sticky policies, etc.

The compliance properties apply to software agents individually. Not surprisingly, the semantics of a complete system (set of software agents) is defined as the sets of the traces of the software agents composing the system[10]. The only additional requirement for a set of traces to be compliant is consistency, which amounts to ensure the matching of communication events: any communication event $E(Id_1, Id_2, *, *, \ldots)$ must match with the corresponding (identical) event in exactly one other trace in the set.

5 Global Correctness

The compliance conditions stated in the previous section impose constraints on the behaviour of software agents. What remains to be shown however is that these constraints are sufficient to ensure that personal data are appropriately protected by the system. Property 1 and Property 2 state two desirable properties of the system:

[9] Note that internal events can change other parts of the state : typically, they can modify the current context (e.g. time or location) and the personal data of the subject.

[10] We assume a finite and fixed set of software agents in this paper.

- Property 1 expresses the fact that if a value of a subject is in the data space of a controller, then this value is associated with a sticky policy *Po* and the subject has defined at some stage a privacy policy allowing a controller with this identity to receive this data with this sticky policy.
- Property 2 states that if the value of a subject is contained in the data space of a controller and the subject has never defined a privacy policy allowing any controller to forward this data, then the subject must have disclosed this data to this controller directly.

Property 1. If Σ is a compliant set of traces of a system of software agents then

$$\begin{array}{l}
\forall (E,S) \in \Sigma, \\
\quad \exists\, i,\ \exists\, Id_2,\ \exists\, Ca,\ \exists\, Po,\ S_i(MyImport)(Id_2,Ca) = (*,*,Po) \\
\quad \Rightarrow \\
\quad \exists\, (E',S') \in \Sigma,\ \exists\, j, \\
\quad S'_j(MyIdentity) = Id_2 \ and \\
\quad Po = S'_j(MyDPolicy)(Ca) = (Id,*,*,*) \ and \\
\quad S_i(MyIdentity) \rhd Id
\end{array}$$

Property 2. If Σ is a compliant set of traces of a system of software agents then

$$\begin{array}{l}
\forall (E,S) \in \Sigma, \\
\quad \exists\, i,\ \exists\, Id_2,\ \exists\, Ca,\ S_i(MyImport)(Id_2,Ca) \neq \perp \ and \\
\quad (\forall (E',S') \in \Sigma,\ \forall\, j, \\
\qquad S'_j(MyIdentity) = Id_2 \ and \\
\qquad S'_j(MyDPolicy)(Ca) = (*,*,Co,*) \\
\qquad \Rightarrow Co = (*,*,\perp,*,*,*)) \\
\quad \Rightarrow \\
\quad \exists\, (E',S') \in \Sigma,\ \exists\, k,\ S'_k(MyIdentity) = Id_2 \ and \\
\quad E_k = DataDisclosure(Id_2, S_i(MyIdentity), Ca, *, *)
\end{array}$$

The complete definition of global correctness includes other properties which can be defined in a similar way such as the compliance with deletion delays and purpose restrictions. A significant benefit of the approach is that the compliance of the set of software agent traces is sufficient to establish global correctness. For example, Property 1 can be proven by recurrence on the length of the software agent traces and decomposition into two subcases corresponding respectively to (1) the collection of the data through direct disclosure from the subject and (2) the collection of the data through transfer by another controller[11]. The conclusion follows from the compliance of the Subject Agent in the first case and from the recurrence hypothesis and compliance of the sending Controller Agent in the second case. The second property can be proven in a similar way.

[11] The occurrence of a *DataDisclosure* or a *TransferData* event are the only possibilities to extend the *MyImport* data space of a compliant controller.

6 Related Work

Privacy policies have triggered a fair amount of interest during the last decade. The approach followed in [24] consists in extending the access control matrix model to deal with privacy rules. The extended model has been used to express the HIPAA [14] consent rules in a formal setting and to check properties of different versions of the HIPAA. The main extensions to the access control matrix model concern the introduction of specific operations for notification and logging. The motivations for this project are thus significantly different from our own goals: as a consequence [24] does not deal with sticky policies, agent compliance or future obligations (obligations used in [24] are conditions on the current context). The same access control matrix approach has been applied to the expression of privacy policies for location-based services [13] based on a Personal Digital Rights Management (PDRM) architecture. [25] has introduced a very generic framework encompassing several families of policies including usage control policies. These controls can be used to enforce some kinds of obligations at different points of time (which differ according to the families) but typical privacy obligations such as deletion or compliance with modification requests do not seem to be amenable to this model.

The Obligation Specification Language (OSL) put forward in [16] is mostly exemplified through DRM policies but can also be used to express privacy policies. It includes usage requirements such as duration, number of times, purpose, notification, etc. OSL is a rich language for obligations including different modalities (such as “must” and “may”) and temporal operators. The semantics of OSL is defined in terms of traces expressed in the Z notation. The work presented here shares with [16] the trace semantics approach but differs in terms of scope and focus: we start with a simple language dedicated to privacy (deriving from natural language statements) and provide a framework for the definition of compliant agents acting as representative of the individuals. In contrast, the objective of [16] was to propose “a language for expressing requirements from many application areas of usage control” and thus dos not include any specific provision for privacy management.

Other contributions aim at providing a formal semantics to existing languages or frameworks, such as EPAL (Enterprise Privacy Authorization Language [1]) in [3] and [2] or the P3P (Platform for Privacy Preferences [30]) in [23][12]. The semantics of EPAL presented in [3] is a decision procedure for the evaluation of access requests. The semantics is then used to define a notion of refinement of policies (satisfaction of a policy by another one) and the composition of policies. The approach followed in [23] and [32] is similar in spirit but uses a relational framework to define P3P privacy policies and APPEL (A P3P Preference Exchange Language) preferences. Similarly [2] proposes an embedding of EPAL into Prolog to benefit from the unification mechanisms for solving privacy queries. The formalism proposed in [19] is a first order predicate calculus built upon an XML framework. This

[12] The work reported in [18] is also related to this paper because it shows how to transform privacy practices expressed in E-P3P into privacy promises expressed in P3P. In contrast with our approach this transformation is bottom-up rather than top-down and it does not rely on a formal semantics.

formalism is used to support regulatory information management and compliance assistance. The main challenge tackled by these projects is the complexity and subtleties of languages which were not designed originally with a formal semantics and contain various sources of ambiguities. As a consequence, these efforts focus on the decision procedure to answer access requests and do not consider, as presented here, compliance properties of individual software agents or global correctness properties. Again, the approach followed here is more focused and top-down (defining a minimal setting to satisfy the legal and technical needs in order to secure the delegation of user's consent to software agents).

The semantics models proposed in [6] are based on RNLS (Restricted Natural Language Statements), a syntax for describing goals in terms of actors, actions and objects. Privacy related statements expressed in a natural language have first to be restated (manually) into RNLS. [6] suggests different kinds of quantitative and qualitative analyses which can then be performed on RNLS statements (for example to assess the level of privacy protection or to answer specific queries about the access to personal data) and [5] presents a method for generating natural language policy statements. Examples of analyses of privacy regulations derived from the HIPAA rules are presented in [7]. [26] also proposes a "semi-structured English syntax" which is used as an intermediate step to translate the Canadian FIPPA (Freedom of Information and Protection of Privacy Act) into EPAL. We share with this trend of work the use of natural language descriptions of privacy policies. However our main target is the definition of privacy policies (by subjects as well as controllers) and their correct implementation (as opposed to the analysis or translation of regulations in [6], [7] and [26]). We also share with the Sparcle project ([8] and [17]) the objective of assisting users to edit privacy policies expressed in a natural language but the user interface issues have been left outside the scope of this paper.

The interest of "a-posteriori" policy enforcement has been strongly advocated in [9] and [11], for example to cope with emergency actions that need to be taken in unexpected circumstances or to address the lack of control of the subjects in a distributed environment. The APPEL (A-Posteriori PoLicy Enforcement) core presented in [11] combines an audit logic with trust management techniques. As in our framework, it makes it possible to define sticky data policies; in addition, it includes provisions for constraining the join of documents and defining policy refinement rules. Trust and accountability are central in [11] and the formal setting is based on audit logs. Even though we have focused on the communications between subjects and controllers agents here (rather than the interactions with the auditor's agent), we also endorse the accountability principle and the introduction of automatic audits in the framework. The audit logic presented in [9] provides a general framework for defining agent accountability based on a proof system and proof obligations of the agents (when they are audited). The main difference between our approach and [9] and [11] is again a matter of focus: considering our goal to specify agents implementing the requirements of data subjects and controllers, we deal with specific obligations such as commitments on actions to be performed in the future (e.g., data deletion), purpose control, rights of the subject to be implemented by the controller (access, modification, deletion, etc.). In addition we consider subject as well as controller policies.

7 Conclusions

The work presented in this paper is part of a broader multidisciplinary project which follows a top-down approach, starting from the legal analysis and defining technical and legal requirements for the development of an effective solution to privacy issues in ambient intelligence environments. Due to space considerations, we have focused on one specific aspect in this paper, namely the specification of privacy agents and only the concepts necessary to convey the essence of the approach have been introduced. The complete framework includes, for example, provisions to define hierarchies of categories and purposes (with cumulative policy constraints), to deliver data with sticky policies stronger than the required policy (using implication rather than equality for compliance checking), to define the commitments of the controller in terms of audit (audit request and audit answer events), etc. Also the actual sets of events, compliance properties and global correctness properties are richer than the versions presented here.

As far as the legal framework is concerned, the roles of the different actors have been defined precisely and contract models have been proposed to formalize the commitments of the software agent providers with respect to the subjects and to the controllers. These commitments establish a double link between statements in SIMPL and software agent implementations: the first link is defined through the formal semantics introduced in this paper and a refinement relation between abstract execution traces and actual logs; the second link is expressed in terms of informal constraints. Typical constraints which have to be expressed informally concern the faithful implementation of the clock (variable $MyTime$ in the software agent states) and the delivery of personal data to applications consistent with the purpose stated in their sticky policy. As far as the formal path is concerned, the locality property put forward in Section 5 is significant both from the technical point of view and from the legal point of view: technically speaking, it makes it possible to reason at the level of individual software agents; legally speaking, it means that liabilities can be associated with software agent providers based on individual commitments.

We believe that the two most important features of the framework presented here are minimality and generality : we have focused on the needs arising from the legal analysis for the specific issue of privacy protection and used the minimal technical setting to reach our goal. Minimality is a pre-requisite in this context both at the level of the natural language used to communicate with the users (to minimize the risks of misunderstanding by a subject or controller) and with respect to the formalization (to minimize the risk of misunderstanding or rejection of the elements of proof by juridical experts in case of litigation). Another significant design choice made in the project is the separation of issues which also corresponds to the legal position to isolate privacy from economical issues: according to this view, personal data are not considered as assets for bargaining but values to be protected independently of any other consideration[13]. As a result, Subject Agents can be seen as a "Privacy Monitors" in charge of controlling all disclosures of data, but strictly limited to this role.

Another consequence of this choice of separation of issues is that security is seen as orthogonal to privacy here: in other words, we have defined privacy as a functional model which should be complemented by appropriate security measures[14]. Note that the framework presented here is flexible enough to refer to security issues though: for example authentication rules or trust policies can be linked to the framework through the management of appropriate sets of identities, certification authorities and protocols can be integrated as conditions before the disclosure of data.

Acknowledgements This work has been partially funded by the INRIA ARC (Action de Recherche Coopérative) PRIAM (Privacy Issues in Ambient Intelligence) and ANR (Agence Nationale de la Recherche) under the grant ANR-07-SESU-007 (project LISE: Liability Issues in Software Engineering).

References

1. P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorization language (EPAL). Technical Report 3485, IBM, 2003.
2. M. Backes, M. Durmuth, and G. Karjoth. Unification in privacy policy evaluation - translating EPAL into Prolog. In *Fifth IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, 2004.
3. M. Backes, B. Pfitzmann, and M. Schunter. A toolkit for managing enterprise privacy policies. In *8th European symposium on Research in Computer Security (ESORICS)*, volume 2808 of *Lecture Notes in Computer Science*. Springer, 2003.
4. S. A. Bibas. A contractual approach to data privacy. *Harvard Journal of Law and Public Policy*, 17, 1994.
5. T. D. Breaux and A. I. Anton. Analysing goal semantics for rights, permissions, and obligations. In *Sixth IEEE Int. Conference on Requirements Engineering (RE'05)*, 2005.
6. T. D. Breaux and A. I. Anton. Deriving semantics models from privacy principles. In *Sixth IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, 2005.
7. T. D. Breaux and A. I. Anton. Mining rule semantics to understand legislative compliance. In *ACM Workshop on Privacy in the Electronic Society (WPES'05)*, 2005.
8. C. A. Brodie, C.-M. Karat, and J. Karat. An empirical study of natural language parsing of privacy policy rules using the Sparcle policy workbench. In *Symposium On Usable Privacy and Security (SOUPS)*, 2006.
9. J. G. Cederquist, R. Corin, M. Dekker, S. Etalle, and J. den Hartog. An audit logic for accountability. In *Sixth IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, 2005.
10. E. Dehiyat. Intelligent agents and intentionality: should we begin to think outside the box? *Computer Law and Security Report*, 22(1):472–482, 2006.

[13] This view, which is widely adopted in Europe, is not accepted by all lawyers though, especially in the US [4].

[14] Indeed, the European Directive [28] explicitly sets forth the liability of the controller to ensure the security of the personal data but does not provide further details on this topic.

11. S. Etalle and W. Winsborough. A posteriori compliance control. In *ACM Symposium on Access Control Models And Technologies (SACMAT'07)*, 2007.
12. G. Finocchiaro. Electronic contracts and software agents. *Computer Law and Security Report*, 19(1):20–24, 2003.
13. C. Gunter, M. J. May, and S. Stubblebine. A formal privacy system and its application to location based services. In *Privacy Enhancing Technologies (PET)*, 2004.
14. Health Resources and Services Administration. Health Insurance Portability and Accountability Act. *Public Law, 104-191*, 1996.
15. M. Hildebrandt. Profiling: from data to knowledge. *DuD: Datenschutz und Datensicherheit*, 30(9):548–552, 2006.
16. M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter. A policy language for distributed usage control. In *12th European Symposium on Research in Computer Security (ESORICS)*, volume 4734 of *Lecture Notes in Computer Science*. Springer, 2007.
17. J. Karat, C.-M. Karat, C. A. Brodie, and J. Feng. Privacy in information technology: designing to enable privacy policy management in organizations. *Int. Journal on Human-Computer Studies*, 63, 2005.
18. G. Karjoth, M. Schunter, and E. Van Herreweghen. Translating privacy practices into privacy promises - How to promise what you can keep. In *Fourth IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, 2003.
19. S. Kerrigan and K. H. Law. Logic-based regulation compliance-assistance. In *ICAIL'03)*, 2003.
20. M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In *Ubicomp 2002 Proceedings*, volume 2498 of *Lecture Notes in Computer Science*, pages 237–245. Springer, 2002.
21. M. Langheinrich. Personal privacy in ubiquitous computing. tools and system support. *Dissertation Document ETH 16100*, 2005.
22. D. Le Métayer and S. Monteleone. Computer assisted consent for personal data processing. *3d LSPI Conference on Legal, Security and Privacy Issues in IT*, 2008.
23. N. Li, T. Yu, and A. Anton. A semantics-based approach to privacy languages. Technical Report 2003-28, CERIAS, 2003.
24. M. J. May, C. Gunter, and I. Lee. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *Computer Security Foundations Workshop (CSFW)*, 2006.
25. J. Park and R. Sandhu. The $UCON_{ABC}$ usage control model. *ACM Transactions on Information and System Security*, 2004.
26. C. Powers, S. Adler, and B. Wishart. EPAL translation of the freedom of information and protection of privacy act. Technical Report Version 1.1, Tivoli Software, IBM, 2004.
27. A. Rouvroy. Privacy, data protection, and the unprecedented challenges of ambient intelligence. *Social Science Research Network*, 2007.
28. The European Parliament and the Council of the European Union. UE directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities*, 1995.
29. A. Veldhuisen, M. Kohras, and et. al. Analysis of privacy principles: Making privacy operational. Technical Report Version 2.0, International Security Trust and Privacy Alliance, May 2007.
30. W3C. Platform for privacy preferences (P3P). W3C recommendation. www.w3.org. Technical report, W3C, 2002.
31. S. Warren and L. Brandeis. The right to privacy. *Harvard Law Review*, pages 193–220, 1890.
32. T. Yu, N. Li, and A. I. Anton. A formal semantics for P3P. In *ACM Workshop on Secure Web Services*, 2004.