A framework for privacy aware data management in relational databases

Pietro Colombo* and Elena Ferrari

Dipartimento di Scienze Teoriche e Applicate,
Università degli Studi dell’Insubria,
Varese, Italy
E-mail: pietro.colombo@uninsubria.it
E-mail: elena.ferrari@uninsubria.it
*Corresponding author

Abstract This paper is about MAPaS – modelling and analysis of privacy-aware systems – framework, which targets the development of privacy aware SQL queries operating on a given database. MAPaS supports the specification of purpose and role-based access control policies that regulate the access to data based on purpose compliance, role and purpose-based authorisations. The current version of MAPaS allows the definition of the scheme of the database whose data must be protected and the SQL queries that should be executed on such a database. A rich analysis toolkit allows user to assess the compliance of these queries with the specified privacy policies. The analysis can be done even before the database is populated. The use of MAPaS bring users to define SQL queries which are privacy aware by design.

Keywords: privacy aware data management; privacy policies; relational databases; model driven development; Object Constraint Language; OCL; Atlas Transformation Language; ATL.


Biographical notes: Pietro Colombo received his PhD in Computer Science from the University of Insubria (Italy) in 2009. Currently, he works as Research Associate within the STRICT SociaLab investigating the definition of privacy-aware data management systems. His research interests are mainly related to data privacy and model driven engineering.

Elena Ferrari is a Full Professor of Computer Science at the University of Insubria, Italy and Scientific Director of the K&SM Research Center. Her research activities are related to access control, privacy and trust. In 2009, she received the IEEE Computer Society’s Technical Achievement Award for ‘outstanding and innovative contributions to secure data management’. She is an IEEE fellow and an ACMD.

This paper is a revised and expanded version of a paper entitled ‘Towards a modeling and analysis framework for privacy-aware systems’ presented at the 4th IEEE/ASE International Conference on Information Privacy, Security, Risk and Trust, Amsterdam, The Netherlands, 3–5 September 2012.
1 Introduction

Most of database management systems (DBMSs) handle personal, identifiable and sensitive data without integrating proper mechanisms to protect data privacy. In contrast, the access to data should be granted only if it complies with specific privacy policies. Nowadays access control is included into numerous DBMSs. Although these mechanisms contribute to regulate the execution of SQL code, they are not sufficient to protect data privacy. Privacy aware data management requires considering privacy aspects in a comprehensive way. A framework is therefore required supporting the specification of privacy policies, the definition of data handling activities, and the analysis of the privacy awareness of these activities wrt the specified policies. The adoption of this framework on a large scale requires it is accessible, easy to use and cost effective.

Numerous frameworks proposed in the literature have been defined to support the development of secure systems. Qamar et al. (2011) compare some recent contributions. However, the majority of the frameworks focus on role-based access control, whereas the few of them that consider privacy in a more comprehensive way suffer from many shortcomings, such as for instance a low usability degree, which hinders their use in industrial processes (Section 2 presents some example). As a consequence, to the best of our knowledge, no comprehensive and easy to use development and analysis framework for privacy aware data management has been proposed yet.

In Colombo and Ferrari (2012) we made a first step to fill this void by proposing MAPaS – the modelling and analysis of privacy-aware systems – framework that we have built around the formalisation of privacy concepts proposed in Byun and Li (2008). MAPaS, which has been developed as a plugin (McAffer and Lemieux, 2005) of IBM Rational Software Architect (RSA) (http://www-01.ibm.com/software/awdtools/swarchitect), consists of the Privacy Aware Modelling Language (PaML), a notation supporting the specification of privacy policies and the design of privacy aware systems, and toolkits supporting the editing, validation and analysis of the designed systems wrt the specified policies.

The work described in this article is a revised and expanded version of Colombo and Ferrari (2012). The extension proposed in this paper consists in the extension of MAPaS to the definition of SQL privacy-aware queries, that is, queries that access a given database satisfying purpose and role-based privacy policies. This has required the extension of PaML and the enhancement of the editing, validation and analysis functionalities of the framework. More precisely, the extension concerned:

1. the introduction of new conceptual elements in the domain model of PaML, which allow the specification of a database scheme, SQL queries and the access purposes associated with the queries
2. definition of new stereotypes, tagged values and constraints for the newly introduced domain model elements
3. definition of the database toolkit, a new module of MAPaS that helps analysts to define a PaML model; this toolkit supports the definition of PaML elements that specify the scheme of the target database (i.e., the database accessed by the queries under definition) and the SQL queries that should be executed on such a database
the definition of editing and visualisation tools for the newly defined stereotypes
the definition of analysis functionalities to assess the compliance of SQL queries
with the purpose and role-based privacy policies specified in PaML models.

For space limitations, in this paper, we discuss selected MAPaS features by focusing on
the novel aspects only. The interested reader can refer to Colombo and Ferrari (2012)
for additional functionalities.

The rest of the paper is organised as follows. Section 2 discusses related work.
Section 3 shortly presents the technologies we have used to define MAPaS. Section 4
presents a running example which is used to ease the presentation of our framework.
Section 5 introduces MAPaS, discussing relevant aspects of its architecture, modelling
language, editing and analysis functionalities. Section 6 discusses the implementation
of selected components of our framework. Section 7 presents an early performance
evaluation of MAPaS functionalities. Finally, in Section 8 we draw some conclusions
and present ideas for future work.

2 Related work

The interest of industry to privacy is proved by the recent definition of specification
languages for privacy policies, such as EPAL (W3C, 2003) and XACML (OASIS,
2013), which have rapidly become de-facto standards. However, no comprehensive
development and analysis framework has been defined around these languages.

In the recent years, most of the research efforts in the privacy field have been
devoted to anonymity techniques for microdata releasing (see Bonchi and Ferrari, 2009
for a survey on these techniques).

A wider range of privacy aspects is also considered by logic frameworks, such as
those proposed in Basin et al. (2010), Cederquist et al. (2007) and Garg et al. (2011),
which support the specification and enforcement of privacy policies. However, the
formal background which is required to adopt these solutions hinders their application in
industry. In contrast, the rich toolkits integrated in MAPaS and its modelling language
make MAPaS an easy to use framework.

A seminal work by Agrawal et al. (2002), presents high level requirements for
the definition of privacy aware ‘Hippocratic’ DBMS. Although this work proposes
strategical development guidelines, these are too abstract for being directly used in
a systematic way. Based on the ideas proposed in Agrawal et al. (2002), Byun and
Li (2008) present a purpose-based access control (PBAC) model and an SQL queries
rewriting approach that enforces privacy policies based on PBAC. The model, which is
built around the concept of purpose, formalises a significant set of privacy concepts. The
work introduces purpose-based access control into existing DBMSs, making them, at
a given degree, privacy aware. However, it focuses on enforcement mechanisms based
on the proposed formal model, but it does not support the specification and analysis
of privacy policies. Several variants of the model presented in Byun and Li (2008)
have been proposed in the literature. For instance, Kabir and Wang (2009) first extends
the model by Byun and Li (2008) integrating the support for conditional intended
purposes, afterwards they also propose a further extension (Kabir et al., 2010) which
also integrates RBAC concepts. In a previous work (Colombo and Ferrari, 2014), we
have proposed an extension of the model by Byun and Li (2008) and mechanisms to
automatically enforce purpose-based policies at query execution time. Ni et al. (2010) propose to extend RBAC through the integration of privacy concepts such as purpose and obligation and related control mechanisms. Similar to Ni et al. (2010), Peng et al. (2008) propose to integrate into RBAC purpose-based access control. However, they also propose a mechanisms to derive the access purposes of user queries from the analysis of system and user attributes.

Different from Byun and Li (2008), Colombo and Ferrari (2014), Kabir and Wang (2009), Kabir et al. (2010), Ni et al. (2010) and Peng et al. (2008), MAPaS operates at design time and it does not enforce policies through query rewriting. It allows analysing if queries can actually access all table and columns that they are supposed to access.

In recent years, model driven development (MDD) has been successfully adopted by the software industry (France and Rumpe, 2007) since it allows handling the intrinsic complexity of systems by means of abstractions, and it is supported by easy to use tools. MDD has also been used for secure systems development. Most of the MDD security solutions in the literature in this field introduce Unified Modelling Language (UML) Profiles (OMG, 2013b) as domain specific modelling language (Kelly and Tolvanen, 2008) for secure system development. Several frameworks (see Qamar et al., 2011) have been proposed to this aim. Among the most cited papers, Jürjens (2005) proposes a UML Profile for secure systems development which supports the specification of security policies and the design of secure system models. Basin et al. (2006) propose another UML Profile supporting the specification of RBAC policies and the design of RBAC systems, and a framework that allows the generation of a security infrastructure capable of enforcing the policies, and analysing the designed systems. Probably due to the numerous aspects that characterise the privacy domain, no MDD framework has been proposed for privacy, yet. With MAPaS we make a step to fill this void.

3 Background

This section shortly describes the technologies that have been used to define MaPAS.

3.1 The UML and its extension mechanisms

The UML (OMG, 2013a, 2013b) is a notation that is largely used by software engineers for specifying and documenting the artefacts generated all throughout software development processes.

The UML specification describes the definition and use of this language by means of models belonging to a layered structure (OMG, 2013a). We focus here on two layers of this structure, denoted M2 and M1, which support the language specification and the user specification, respectively. More specifically, layer M2, which consists of the UML meta-model, introduces the basic elements of the UML notation (e.g., meta-class class), whereas layer M1 supports, at different levels of granularity, the specification of systems. User specifications are defined by means of a UML model that belongs to layer M1, which is composed of

1. instances of UML meta-model elements (e.g., the user defined Class A is an instance of meta-class class)
2. instances of user defined elements (e.g., x, y, z which are all instances of Class A).
Extension mechanisms allow one to extend UML to model elements and properties of special purpose systems. In particular, the profiling mechanism allows the definition of new UML elements through the extension of other elements of the UML meta-model. These new elements are defined as instances of the UML meta-class Stereotype. Each stereotype is mapped to an existing UML meta-class and extends it refining or modifying its semantics. More precisely, a Stereotype can introduce attributes, referred to as tagged-values, which specify properties that do not characterise the extended meta-classes. It can also introduce logic predicates, referred to as constraints, which constrain tagged values and/or properties of the extended meta-classes. The set of the newly defined stereotypes, their tagged-values and constraints, form a UML Profile.

The profiling mechanism is supported by most CASE tools. The definition of domain specific modelling languages as UML Profiles maximises the portability of these languages across several platforms and favours the development of analysis and editing tools on top of these languages using standard technologies. These reasons brought us to define PaML as a UML Profile.

3.2 The OCL

The Object Constraint Language (OCL) (OMG, 2012) is a formal language used to complement UML models with class invariants, pre and post conditions to the execution of operations, initialisation and derivation expressions. The language can also be used to specify queries on UML models.

OCL is used during the definition process of UML Profiles, to restrict the semantics of UML meta-classes making them suitable to represent domain specific elements. For instance, the OCL expression in Listing 3 – Section 6.1, restricts the specialisation relationships allowed for meta-class Interface to which Role is mapped. OCL expressions that specify invariants and pre/post conditions on operations are embedded into UML models (at level M1).

Although numerous OCL tools allow interpreting OCL expressions, most of them are only partially aligned with the OCL standard (Chimiak-Opoka et al., 2011). We have personally experimented this situation. Indeed, the OCL interpreter embedded in MAPaS, which is the native OCL interpreter of IBM RSA, does not allow evaluating OCL expressions defined on UML models (at M1). We address this issue evaluating OCL expressions with the Atlas Transformation Language (ATL) framework (Jouault et al., 2008).

3.3 The ATL

The ATL (Jouault et al., 2008) is a model transformation framework composed of a language and a toolkit. ATL is commonly used to derive a target model from source models by means of transformation rules. However, it can also be used to query models, by extracting model properties.

The model transformation language integrated into the ATL framework has been defined as an extension of a sub-core of OCL. A consequence of this choice is the capability of the ATL toolkit to evaluate OCL expressions. The models that embed the OCL expressions can belong to any layer of the layered structure presented in Section 3.1. As such, ATL can be used to evaluate OCL expressions defined in PaML.
A framework for privacy aware data management in relational databases

models at M1, thus addressing the weaknesses of the OCL interpreter of RSA. Moreover, it can also be used to query PaML models, deriving relevant information.

ATL queries and helpers are key elements that will be used in the development of MAPaS analysis functionalities. An ATL query is a sort of function that extracts data from the model under analysis based on what specified by an OCL expression which is denoted as the query’s body. Listing 5 in Section 6.2, shows a query that extracts from a PaML model all users that do not belong to any role. Non-trivial queries can be hardly specified within a unique OCL expression. To ease the definition of complex queries, one can use helpers, which are statements that factorise a set of instructions. The context of the helper specifies the model element on which this statement is executed. Helpers even allow the declaration of input/output parameters. The body of a helper is an OCL expression. An example of helper function is shown in Listing 6 – Section 6.2.1, which extracts from a PaML model those Purpose elements with given characteristics (thorough explanation will be given in Section 5). Helper are invoked within the body of other helpers or of ATL queries. For instance, the ATL code in Listing 6, shows the invocation of getAncestors and getAncestorsAndDescendants from the body of getImpliedPurposes.

4 Running example

A case study from the business domain is used all throughout the paper to ease the presentation of PaML concepts and the definition of the analysis functionalities of MAPaS.

The chosen example concerns the purpose and role-based access control system that regulates the processing of data stored in the databases of MyCompany, a small software company. The data processing is regulated based on the compliance of collection and access purposes and on the belonging of the users that require the access to authorised roles. Purposes and roles are hierarchically organised as tree structures. Figure 1 shows the purpose hierarchy, while the inverted tree in Figure 2 presents the role one. The concepts of purpose and role will be introduced in Section 5.

Table 1 shows the content of customer, a table of MyCompany’s database which collects personal and sensitive data of MyCompany’s customers.

5 The MAPaS framework

MAPaS has been implemented as a set of plug-ins to be integrated into IBM RSA.

MAPaS has been defined on top of the PaML, a domain specific modelling language for the privacy domain. The framework is composed of four modules denoted Editing toolkit, Database toolkit, Validation toolkit and Analysis toolkit, respectively. The graphical user interface of MAPaS allows users to select a desired functionality provided by the aforementioned modules. The architecture of MAPaS and the control flow of relevant activities performed by its modules is sketched in Figure 3.

The Editing toolkit allows for generating and visualising PaML models by means of dedicated editors, palettes and views. The whole module has been defined using the Profiling tool of IBM RSA, a component that generates the source code of editors, palettes and viewers tailored to the characteristics of PaML.
Figure 1  Purpose tree

```
Purpose tree

Admin  Purchase  Shipping  Marketing
Backup  Analysis  Direct  ThirdParty

DEmail  DPhone  TEmail  TPostal
SpecialOffer  ServiceUpdate
```

Figure 2  Role tree

```
Role tree

Employee

AdminDept  PurchaseDept  ShippingDept  MarketingDept

EMarketing  TeleMarketing

EAnalyst  Writer  TAnalyst  Operator
```

Table 1  Customer table

<table>
<thead>
<tr>
<th>id</th>
<th>Name</th>
<th>Street</th>
<th>City</th>
<th>Nation</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>John</td>
<td>543, 5th Avenue</td>
<td>New York</td>
<td>USA</td>
<td>110 K</td>
</tr>
<tr>
<td>02</td>
<td>Alice</td>
<td>432, 3rd Avenue E</td>
<td>Montreal</td>
<td>CA</td>
<td>100 K</td>
</tr>
<tr>
<td>03</td>
<td>Bob</td>
<td>321, 1st Avenue</td>
<td>London</td>
<td>UK</td>
<td>124 K</td>
</tr>
</tbody>
</table>

Figure 3  An overview of MAPaS architecture
The Database toolkit supports the definition of part of a PaML model that describes properties of the target database. The toolkit consists of wizards that extract meta-data of the target database and drive the analysts through the process of assigning policies to database elements (we will explain this in Section 5.2). The module is a Java application that uses the JDBC API to interact with the DBMS that hosts the target database and API of the Editing toolkit to edit the PaML model under definition.

The Validation toolkit validates PaML models against the OCL constraints included in the UML meta-model and those in the PaML profile. The verification exploits the OCL interpreter natively integrated into IBM RSA. No additional validation functionalities have been defined. After selecting the elements to be checked, the analyst invokes the verification. The interpreter selects the constraints of the UML meta-model and of the PaML Profile that are associated with the selected elements, and evaluate them. Detailed messages are sent back to the user in case of constraint violations. The control flow of the validation process is sketched in Figure 3.

The Analysis toolkit is used to assess properties of PaML models. A rich set of ATL queries allows extracting information from these models. After selecting the elements that must be evaluated, the analyst invokes the proper queries (see the control flow in Figure 3). Significant analysis functionalities supported by MAPaS are presented in Section 5.3.

In the rest of this section, we first introduce PaML, afterwords we discuss relevant editing and analysis functionalities of MAPaS.

5.1 Privacy-aware Modelling Language

In this section, we introduce the core conceptual elements allowing the specification of purpose and role-based privacy policies. These elements, whose semantics is based on the model presented in Byun and Li (2008), compose the domain model of PaML.

The domain model is presented as a UML model composed of classes, relationships and well formedness rules (we use UML since it is largely known, but other notations may be used for the same goal). Classes and relationships are used to represent domain elements and how these elements are connected among them at conceptual level, whereas well formedness rules are OCL invariant constraints that bind properties of these elements. The UML class diagram in Figure 4 shows an high level view on the domain model.

In the reminder of this section, we shortly introduce the main elements that compose this model, also exemplifying some well formedness rules.

The concept of purpose, which is modelled by element Purpose in Figure 4, represents the reasons for which data are collected and used. Purpose compliance, which states that data can be used if the purposes for which they are accessed comply with those for which they have been collected, basically represents the essence of the privacy policies considered in this work. The Purpose elements defined for a given application domain are connected among them and hierarchically organised as tree structures. The purpose tree that specifies the purposes involved in our running example is shown in Figure 1. Based on this hierarchical organisation, a Purpose is possibly specialised by multiple Purposes, but each Purpose can specialise only one more general Purpose. This constraint is an example of a well-formedness rule, whose OCL formalisation is presented in Listing 1.
The element IntendedPurpose is used to specify the purposes for which data can and cannot be accessed. IntendedPurposes explicitly specify prohibitions as required by several privacy laws and regulations. For instance, the Children’s Online Privacy Protection Act (http://epic.org/privacy/kids) states that any information about children of age under 13 cannot be used for marketing purpose. As such, IntendedPurpose aggregates allowed and prohibited purposes (modelled by the associations aip and pip in Figure 4).

Example 1: Let us suppose to introduce an IntendedPurpose element IP1 which, based on the set of purposes considered in our running example (see Figure 1), is defined as IP1 = ⟨{Admin, Direct}, {DEmail}⟩. The set {Admin, Direct} specifies the allowed purposes, whereas {DEmail} models the prohibited purpose.

We impose the obvious constraint that an element Purpose cannot belong to the set of aip and pip of the same IntendedPurpose. This rule is formalised in OCL in Listing 2.

context IntendedPurpose
defined by:
  self.aip->intersection(self.pip)->isEmpty() (2)

In our framework data organisation reflects the one of relational databases, thus data are organised as Tables, Attributes, Tuples, and Elements.

The assignment of IntendedPurpose elements to Data, which in Figure 4 is modelled by the association bp, can be specified at each of the above-mentioned granularity levels (i.e., Table, Attribute, Tuple or Element).

Example 2: Suppose that the IntendedPurpose IP1 = ⟨{Admin, Direct}, {DEmail}⟩ is assigned to Table Customer (see Table 1). Suppose that, a more restrictive privacy policy should be defined for Attribute income. This can be done by introducing IntendedPurpose IP2 = ⟨{Admin}, {DEmail, DPhone}⟩ and assigning it to Attribute income.
Element Query models an SQL query that accesses data for given access purposes. Association \( ap \) specifies the Purpose for which the Query processes data. The tables and attributes accessed by a Query are defined by the associations \( tb \) and \( at \), respectively.

Element Environment models contextual properties of the system under development.

**Example 3:** Suppose to define Environment \( Env \) characterised by the properties \( timeOfDay \) and \( dayOfWeek \), whereas \( envS = \langle \text{timeOfDay}=9, \text{dayOfWeek}='Monday' \rangle \) is an instance of \( Env \).

Element User has been introduced to specify stakeholders who request the access to data. User properties, such as personal information (e.g., name, surname, address) and capabilities within the organisation (e.g., known foreign languages), are modelled by means of a set of attributes. Our model allows the specification of users both at type and instance level. The specification at type level allows the definition of a group of stakeholders characterised by the same set of attributes, whereas at instance level, it defines a single stakeholder characterised by a set of attributes initialised to given values. This distinction represents an enhancement wrt the proposal by Byun and Li (2008) in which only single stakeholders are modelled.

**Example 4:** Let us introduce User \( \text{PersonnelOffice23} \) to model a group of stakeholders who work in the office 23 of the marketing department. \( \text{PersonnelOffice23} \) is characterised by the attributes \( \text{name, id, officeId, serviceType, expLevel, teamLeaderId, and yearsInCompany} \). Jack and Mary are two instances of \( \text{PersonnelOffice23} \) such that, \( \text{Jack=\langle name='Jack', id=123, officeId=23, serviceType='ReadInfo', managerId=2345, expLevel='low', teamLeaderId=234, yearsInCompany=2\rangle} \), whereas \( \text{Mary=\langle name='Mary', id=234, officeId=23, serviceType='UpdateInfo', managerId=1234, expLevel='high', teamLeaderId=234, yearsInCompany=7\rangle} \).

Users can cover one or multiple roles within an organisation. Element Role has been introduced to model functionalities and responsibilities that can be covered by users.

Roles can be assigned to Users both at type and at instance level. In the former case, all instances of the involved type cover the specified Role, while in latter case the specified role is assigned to a single stakeholder. The association \( br \) models the association of roles to users.

Similar to the case of User, element Role aggregates attributes that specify characteristics of the role. For instance, Role \( \text{Employee} \) is characterised by attributes \( \text{id, name, yearsInCompany} \) (see Figure 2).

When a Role \( R \) is assigned to \( UI \), an instance of User \( U \), the attribute values of \( R \) are initialised to the values of the homonymous attributes of \( UI \).

**Example 5:** Let us assign Role \( \text{Employee} \) to \( \text{PersonnelOffice23} \). Therefore, since \( \text{Employee} \) is assigned to \( \text{PersonnelOffice23} \), and Jack is an instance of \( \text{PersonnelOffice23} \) (see Example 4), the values of the \( \text{Employee} \)'s attributes based on Jack are \( \langle \text{id}=123, \text{name}=\text{Jack}, \text{yearsInCompany}=2\rangle \). Moreover, let us assign Role Writer to User \( \text{PersonnelOffice23} \). The derived values of Writer's attributes based on Mary are \( \langle \text{teamLeaderId}=234, \text{serviceType}=\text{UpdateInfo}, \text{id}=234, \text{name}=\text{Mary}, \text{yearsInCompany}=7, \text{managerId}=1234, \text{expLevel}=\text{high} \rangle \).
Roles are hierarchically organised as inverted tree structures. Within these structures, the attributes introduced by a Role R are inherited by all Roles that specialise R. For instance, let us consider the hierarchy of Roles defined for MyCompany, which is shown in Figure 2. According to that structure, Role EMarketing:

1. defines the attributes teamLeaderId and serviceType
2. inherits the attributes id, name and yearsInCompany defined by Employee
3. inherits the attributes managerId and expLevel of MarketingDept.

Based on Byun and Li (2008) the concept of conditional role is used to constrain the assignment of authorisations to users using a more fine-grained mechanism than the per-role one. Element ConditionalRole extends Role specifying a condition that must be satisfied to consent the assignment of the extended Role to a User. This condition is a propositional logic formula that is built starting from the attributes of Environment and the attributes inherited by the extended Role. For instance, ConditionalRole EMarketingCR extends Role EMarketing introducing the constraint yearsInCompany>5 and serviceType='UpdateInfo'.

An instance UI of a User U belongs to a ConditionalRole CR in an Environment state ES when the CR constraint is satisfied and UI has been authorised to play a Role R1 that dominates the Role R2 specialised by CR.

Example 6: Considering Example 4, the only instance of User that belongs to EMarketingCR is Mary. Indeed, Role Writer assigned to PersonnelOffice23 dominates Role EMarketing and, differently from the values of Jack’s attributes, those of Mary’s satisfy the constraint yearsInCompany>5 and serviceType='UpdateInfo'.

Access purposes are authorised to users on conditional role basis, meaning that they are granted to instances of User that belong to ConditionalRole only. In our model, authorisations are specified by instances of the association class AccessPurposeAuthorisation, which binds a ConditionalRole r to an access Purpose p.

Example 7: Let us authorise AccessPurpose AP1 = Marketing to ConditionalRole EMarketingCR. Due to this authorisation, all instances of Users that belong to EMarketingCR are allowed to access data for marketing purposes. Therefore, based on Example 6, Mary is authorised to access data with the Purpose Marketing.

5.2 Editing and database interaction capabilities

PaML supports the specification of privacy policies that regulate the processing of data stored in a target database. The essence of these policies is to allow the execution of SQL queries if the execution requests come from authorised users, and the access purpose of the queries comply with the purposes for which data that should be accessed have been collected.

Let us denote Data protection group, the set of PaML elements used to allocate policies to data stored in the database. This group includes the elements Table, Attribute, User, Query and AccessPurposeAuthorization and associations that involve these elements.
Even though the Editing toolkit allows security administrators to entirely edit a PaML model, it is not convenient to define elements of the Data protection group from scratch. In some cases, the database on which privacy policies should be enforced already exists (it can be already populated or not).

Elements of the data protection group refer to characteristics of the target database. In particular, elements Table and Attributes define its scheme. MAPaS allows for deriving them automatically. The Database analysis wizard (see Figure 3) allows system administrators to specify the interaction parameters (e.g., the URL of the host, the database name, etc.) with the DBMS that hosts the target database. The interaction exploits JDBC technology, which allows interfacing almost any type of existing relational DBMS with a JDBC driver. This wizard generates instances of Table and Attribute that model the scheme of the database. It also shows to administrators the list of currently defined IntendedPurposes that compose the PaML model under definition, allowing them to pick the one to be assigned to every Table and Attribute. Basically, this wizard modifies the PaML model instantiating Table and Attribute elements and defining associations between the involved instances of Table/Attribute and already defined IntendedPurpose elements.

The Database toolkit also includes the Query analysis wizard (see Figure 3) that helps administrators to define Query elements in PaML models. System administrators define the SQL code of the queries that they are interested in evaluating for compliance with the specified privacy policies. Such a code is provided as input to the wizard, which parses it deriving the list of accessed tables and attributes. The wizard identifies the instances of the elements Table and Attribute included in the PaML model that are accessed by the query. The wizard also shows the list of Purpose elements defined in the current PaML model, allowing the system administrator to select the access purpose to assign to the Query under definition. All these elements are used to instantiate new Query elements, contributing to the definition of the PaML model.

Example 8: Let us suppose that a system administrator defines the SQL query select name, income from customer which accesses data for the marketing purposes (see Table 1 and Figure 1). The Query analysis wizard instantiates a Query object Q which is connected to

1. customer, which is an instance of Table
2. name and income, which are Attributes of the customer table, and
3. marketing, which is an instance of Purpose.

In Section 5.3 we show how the generated Query elements that have been instantiated can be checked for compliance with the purpose-based policies defined in a PaML model.

5.3 Analysis capabilities

MaPAS allows system administrators to reason on properties of the privacy policies specified within a PaML model. It also allows them to evaluate the executability of SQL queries based on the specified policies. Indeed, the privacy policies included in a PaML model are defined to constrain the execution of SQL queries on the target database.
MAPaS analysis functionalities are completely based on a PaML model which is defined using visual tools of the Editing toolkit and wizards of the Database toolkit. Using MAPaS there is no need to practically define and populate the target database to check whether the SQL queries that should be executed on this database satisfy the specified privacy requirements. MAPaS allows one to evaluate the privacy awareness of queries without the need to interact with any DBMS.

Traditional access control systems evaluate the executability of given actions at runtime based on properties of the system under development. Even privacy aware access control systems for DBMSs, like PBAC (Byun and Li, 2008), handle the executability of SQL code when the queries are invoked for execution. Therefore, with traditional approaches to check whether given SQL queries comply with a set of policies it is first required to configure the target database and populate it with purposes, roles, users and data. Since data protection based on purpose and role-based privacy policies is not natively provided by any existing commercial DBMS, the management of these policies requires ad hoc programming in which numerous DBMS platform specific details are considered (e.g., the definition of control mechanisms uses different code with an Oracle platform and IBM one).

Using MAPaS, this effort is not required. The rich set of editing and analysis functionalities allow analysts to reason on privacy policies without considering any implementation or platform oriented detail. They can focus uniquely on privacy aspects. This approach allows them to identify possible shortcomings and mistakes before configuring and populating the target database.

Another advantage of the approach concerns the possibility to handle issues at design time. For instance, in case MAPaS finds out that an SQL query does not comply with the policies specified in a PaML model, the query or the database scheme specified in that model can be immediately modified. The general principle is to populate the target database only after verifying that the PaML model under development satisfies all privacy policies. In contrast, with traditional approaches it could be necessary to modify already populated tables, which is extremely time demanding and error prone. As such, the management of data privacy issues at design time operated by MAPaS has several advantages wrt the implementation-based traditional one. For these reasons, we believe that using MAPaS one can substantially speed-up the development of privacy aware SQL queries. Although the MAPaS modules and functionalities described in this paper are focused on the early stages of the development, we are working to define and integrate into MAPaS run-time monitoring and trace execution analysis capabilities.

A selection of some of the most relevant analysis functionalities supported by MAPaS is shown in Table 2. The implementation strategies we have followed to define these functionalities will be presented in Section 6.2. Some of the analysis functionalities in Table 2 are similar to functionalities integrated into the analysis frameworks of access control systems [e.g., the Review functions of RBAC systems (Ferraiolo et al., 2001)]. However, although carrying out similar functionalities, our analysis is performed on models, thus, it does not require the target database to be populated. Additionally, in our approach the analysis is anticipated in the early phases of the development. and MAPaS analysis is based on a purpose and role-based access control model.
Table 2  A selection of MAPaS assessment queries

1. **getDeniedPurposeToRole** it derives all purposes that cannot be assigned to a Role.
2. **getRolesAuthorizedToPurpose** it derives all roles that are authorised for an access purpose.
3. **getUsersWithAuthorizedRole** it derives all users at type and instance level that explicitly or implicitly belong to a Role.
4. **getUsersWithNoRole** it derives all users at type and instance level that do not explicitly or implicitly belong to any Role.
5. **getImpliedPurposes** it derives all purposes implied by an IntendedPurpose.
6. **getRolesAssignedToUser** it derives all roles that are associated with a user at instance or type level. This query takes into consideration both explicitly associated and inherited roles.
7. **getPurposeAuthorizedToConditionalRole** it derives all access purposes that are explicitly granted to or inherited by a conditional role.
8. **getPurposeGrantedToUser** it derives all access purposes that are granted to an instance of User in an environment state based on the conditional role the User belongs to.
9. **accessPurposeVerification** it verifies whether an AccessPurpose can be authorised based on sets of allowed and prohibited purposes of an IntendedPurpose element.
10. **roleIsAssignableToUser** it verifies whether a Role can be assigned to a User.
11. **userBelongsToConditionalRole** it verifies whether an instance of User belongs to a ConditionalRole in a given EnvironmentState.
12. **deriveUserRoleAttributes** it derives the value of Roles attributes starting from the attribute values of the instance of User to which Role is assigned.
13. **getUsersWithNoAuthorizedRole** it derives all users at type and instance level that do not explicitly or implicitly belong to any Role.
14. **userBasedAccessPurposeVerification** it evaluates if an access purpose can be granted to an instance of User with an authorised Role.
15. **intendedPurposeAssignmentPolicy** it evaluates if an IntendedPurpose assigned to a Tuple/Element is correct wrt to the purpose assignment policy specified for the involved Table.
16. **deriveIntendedPurposes** it derives all the IntendedPurposes that are associated with the Attribute and Table elements accessed by a Query Q.
17. **deriveQueryAttributes** it derives all the Attributes of the tables accessed by a Query Q whose data are actually accessed by Q.
18. **deriveQueryTables** it derives all the Tables that are accessed by a Query Q.
19. **compliesWith** it evaluates if an IntendedPurpose IP complies with a given Purpose P. This is achieved checking that P is included in the purposes implied by IP.
20. **queryPurposeCompliance** it evaluates if all the IntendedPurposes associated with data accessed by a Query Q comply with the access Purpose of Q.
21. **deriveNonCompliantTables** it derives the list of Tables among those accessed by a Query Q, which based on the purpose compliance analysis cannot be accessed by Q.
22. **deriveNonCompliantAttributes** it derives the list of Attributes among those accessed by a Query Q, which based on the purpose compliance analysis cannot be accessed by Q.
23. **complianceIndicator** it derives the percentage of Attribute and Table elements accessed by a Query Q whose IntendedPurposes do not comply with the access Purpose of Q.
For space limitations we cannot discuss all analysis functionalities of MAPaS. In our previous paper (Colombo and Ferrari, 2012) we focused on those used to derive the intended purposes associated with data based on assignments specified at multiple granularity levels (e.g., to tables and attributes). Since the focus of this work is on privacy aware data management, in the rest of this section we discuss functionalities supporting the analysis of SQL queries.

Let us consider a situation in which system administrators have introduced the Query elements in the PaML model they want to analyse, using the approach presented in Section 5.2. Once Query elements have been defined, MAPaS allows one to evaluate them for compliance with the privacy policies. Queries can be executed if the access purpose for which they aim to access data complies with the intended purposes for which data have been stored. MaPAS provides analysis functionalities that evaluate such compliance.

According to Byun and Li (2008), a Purpose \( p \) complies with an IntendedPurpose \( ip \), if \( p \) is included in the set of purposes implied by \( ip \). Compliance analysis is performed by query \( \text{compliesWith} \) (see Table 2). A query \( Q \) satisfies the specified purpose-based policies if all the IntendedPurposes associated with data accessed by \( Q \) comply with the access purpose of \( Q \). The compliance of the whole query is calculated through the conjunction of the compliance results for every accessed attribute and table. \( \text{queryPurposeCompliance} \) performs this calculation (see Table 2).

Example 9: Let us consider the purpose compliance analysis which is performed for the SQL query \( Q \) introduced in Example 7. \( \text{queryPurposeCompliance} \) checks for purpose compliance all the attributes accessed by \( Q \). As specified in Example 2, the IntendedPurpose \( IP1 \) is assigned to \( \text{name} \), while \( IP2 \) is assigned to \( \text{income} \). Therefore, \( \text{queryPurposeCompliance} \) invokes \( \text{compliesWith} \) for the pairs \( \langle IP1, \text{name} \rangle \) and \( \langle IP2, \text{income} \rangle \). Based on the purpose hierarchy in Figure 1, none of the checks is satisfied, and thus \( Q \) does not satisfy the policies specified within the current PaML model.

In case a query accesses multiple attributes/tables, the modeller may be interested in knowing which attributes cannot be accessed based on the stated access purposes. \( \text{deriveNonCompliantAttributes} \) selects from the list of Attribute/Table elements accessed by a Query under analysis \( Q \) those whose IntendedPurposes do not comply with the access Purpose of \( Q \).

MaPAS even allows the derivation of compliance measures. A purpose compliance indicator is calculated as the ratio between the sum of cardinality of the sets returned by \( \text{deriveNonCompliantAttributes} \) and \( \text{deriveNonCompliantTables} \), and the number of attributes accessed by the query.

A second requirement for the execution of the Query is that the user who requests the execution is actually authorised based on the existing access purpose authorisations. \( \text{userBasedAccessPurposeVerification} \) performs this check.

Example 10: Let us consider a simple scenario in which Peter, who is a system administrator, is modelling the purpose-based access control system of MyCompany. In particular, he has just defined:
A framework for privacy aware data management in relational databases

1. Role AdminDept (see Figure 2), to grant that role the permit to access all data in the Customer table

2. ConditionalRole AdminDeptCR, which extends AdminDept with the constraint 
   timeOfDay<9 and timeOfDay>17 and yearsInCompany>7

3. AccessPurpose AP2 = \{Backup\}.

Peter authorises AP2 for AdminDeptCR and erroneously assigns AdminDept to PersonnelOffice23 (see Example 4). Peter uses some MAPaS analysis mechanisms to assess his specification. The access to the data in Table Customer is regulated by IntendedPurposes IP1 and IP2 introduced in Example 2. He invokes accessPurposeVerification to check if AP2 is valid for each involved IntendedPurpose element. This is the case, since Backup is implied by all the involved IntendedPurposes. He also invokes roleIsAssignableToUser to see if AdminDept can be assigned to PersonnelOffice23. The check fails, in fact AdminDept specifies the attribute computerSkill, and this attribute is not part of those that characterise PersonnelOffice23. Based on this analysis, Peter realises that he erroneously assigned AdminDept to PersonnelOffice23. In fact, PersonnelOffice23 models stakeholders that work in the marketing department, while he should have assigned it to PersonnelOffice33 whose stakeholders work in the information system department. As a consequence, he replaces the wrong assignment with the correct one.

6. Implementation strategies

In this section, we present the strategies we have followed for the development of

1. PaML as a UML Profile
2. MAPaS analysis functionalities.

6.1 Building the PaML profile

According to the guidelines for the definition of UML Profiles proposed by Selic (2007), once the domain model has been defined, each domain model element must be mapped to an element of the UML meta-model. Then, it is required to introduce stereotypes, tagged values, and constraints that specify the characteristics of these elements.

The reminder of this section illustrates the process we have followed, exemplifying the afore mentioned activities for some PaML domain model elements.

The mapping phase is the most critical part of the whole definition process. It requires to analyse the UML meta-model and to identify the UML meta-classes most suited to model the characteristics of each domain element. The criticality is due to the lack of systematic selection criteria and to the possibility of identifying multiple candidate meta-classes (Salehi et al., 2010). The selection is essentially based on the comparison of the structural properties and semantics aspects of domain model elements with those of the UML meta-classes.

For instance, based on the domain model presented in Section 5.1, Role has been defined as an element that aggregates attributes. Numerous UML-meta-classes are defined with this capability, such as Class, Interface, and Component. Thus, to identify
the most suited UML meta-element we need to consider further semantics aspects. The attributes of Role specify a sort of abstract scheme that is realised only when Role is assigned to User. Indeed, Role attributes are initialised to the values of the corresponding User attributes only when the Role is assigned. Based on this reasoning, Interface is the most suitable candidate among the previously selected ones.

The next activity of the profile definition requires the introduction of new UML Stereotypes that extend the selected UML meta-classes. For instance, the Stereotypes User and Role are introduced as extensions of the meta-classes Class and Interface, respectively.

Similar mapping and definition criteria are applied to those elements of the domain model that are defined as relationships. For instance, for relationship br we define the Stereotype BelongsToRole as extension of meta-class Association. Table 3 shows the list of the defined Stereotypes and the corresponding extended meta-classes.

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML meta-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessPurpose</td>
<td>Package</td>
</tr>
<tr>
<td>AccessPurposeAuthorization</td>
<td>Association</td>
</tr>
<tr>
<td>BelongsToRole</td>
<td>Association</td>
</tr>
<tr>
<td>Query</td>
<td>Class</td>
</tr>
<tr>
<td>ConditionalRole</td>
<td>Interface</td>
</tr>
<tr>
<td>DerivableProperty</td>
<td>PrimitiveType</td>
</tr>
<tr>
<td>Environment</td>
<td>Class</td>
</tr>
<tr>
<td>IntendedPurpose</td>
<td>Package</td>
</tr>
<tr>
<td>Purpose</td>
<td>Interface</td>
</tr>
<tr>
<td>Table</td>
<td>Class</td>
</tr>
<tr>
<td>Attribute</td>
<td>Property</td>
</tr>
<tr>
<td>Role</td>
<td>Interface</td>
</tr>
<tr>
<td>Tuple</td>
<td>InstanceSpecification</td>
</tr>
<tr>
<td>Element</td>
<td>Slot</td>
</tr>
<tr>
<td>User</td>
<td>Class</td>
</tr>
</tbody>
</table>

The stereotypes definition activity can require to define tagged values, if domain elements properties are not modelled by the selected UML meta-classes. For instance, since domain element IntendedPurpose has the capability to aggregate purposes, it has been mapped to meta-class Package. Actually, also this meta-class has the capability to aggregate elements. However, IntendedPurpose aggregates allowed and prohibited purposes, but meta-class Package does not categorise the packaged elements. Therefore, we define the stereotype IntendedPurpose as characterised by the tagged values aip and pip, which model the allowed and prohibited purposes, respectively.

The final phase of the profile definition activity consists in the specification of constraints. The goal of this activity is the restriction of the UML meta-model in such a way that it can match the semantics of domain model elements.

For instance, as specified in the domain model, Roles are organised as inverted tree structures, within which each Role can extend a single Role element. Based on Table 3, Stereotype Role is defined by means of the extension of meta-class Interface, but this
A framework for privacy aware data management in relational databases

can specialise any number of other Interfaces. Therefore, as shown in Listing 3, we introduce a constraint that restricts the inheritance relationship.

\[
\text{self.general} \rightarrow \text{size()} = 0 \text{ or } (\text{self.general} \rightarrow \text{size()} = 1 \text{ and self.general} \rightarrow \text{at(1).isStereotypeApplied(PaML::Role.oclAsType(uml::Stereotype)))} \tag{3}
\]

The definition of stereotypes that model domain elements that have been defined as relationships uses the same process that we have applied to all the other elements. For example, let us consider the relation br, which models the covering of roles by users. Based on Table 3, Stereotype BelongsToRole has been defined as an extension of meta-class Association, which can link classifiers of any type. However, according to the domain model, at one end we must have User, at the other end a Role. Listing 4 exemplifies this constraint.

\[
\text{self.endType()} \rightarrow \text{at(1).isStereotypeApplied(PaML::User.oclAsType(uml::Stereotype)) and self.endType()} \rightarrow \text{at(2).isStereotypeApplied(PaML::Role.oclAsType(uml::Stereotype))} \tag{4}
\]

The collection of all the newly defined stereotypes, tagged values, and constraints forms the PaML profile.

6.2 Implementation of analysis functionalities

The Analysis toolkit of MAPaS consists of a collection of ATL queries and helper functions that allow analysts to assess different aspects of a given PaML model. Table 2 lists the most relevant analysis functionalities of MAPaS. Based on the mechanisms used for their interpretation, queries are classified into the simple and composite categories.

6.2.1 Simple queries

Queries belonging to this category can be evaluated in a single step. Simple queries can target the entire PaML model under development. For instance, an analyst may wonder whether the PaML model includes users that do not belong to any Role. This query, whose ATL implementation is shown in Listing 5, requires the analysis of the whole PaML model.

\[
\text{query getUserWithNoAuthorizedRole= (MM!Class} \rightarrow \text{allInstances()} \rightarrow \text{select(c|c.isStereotypeApplied(PaML::User.oclAsType(uml::Stereotype)) and c.ownedAttribute} \rightarrow \text{select(a|a.isStereotypeApplied(PaML::Role.oclAsType(uml::Stereotype))) ->size()} = 0))) \tag{5}
\]

Queries can also target selected elements of the PaML model. For instance, an analyst may be interested in deriving all Purposes implied by an IntendedPurpose \( IP \). Based on Byun and Li (2008), this requires to calculate:
AIP↓: a set which groups all Purpose elements that descend from the Purpose elements specified by the $aip$ field of IP

PIP↑: a set which groups all Purpose elements that either precede or descend from the Purpose elements specified by the $pip$ field of IP.

These sets are used to derive the implied purposes, which are those included into $\text{AIP} \downarrow \text{PIP} \uparrow$.

The ATL implementation proposed for this functionality uses helper functions. The helpers $\text{getDescendants/} \text{getAncestorsAndDescendants}$ are used to derive the purposes that descend from/descend from or precede a given Purpose. The helper $\text{getImpliedPurposes}$ shown in Listing 6 invokes $\text{getDescendants}$ and $\text{getAncestorsAndDescendants}$ calculating the sets $\text{AIP} \downarrow$ and $\text{PIP} \uparrow$, and derives the implied purposes as the difference of these sets.

```reasoning
helper def: getImpliedPurposes(src: MM!Package): Set(MM!Class) =
if(src.hasStereotype('PaML::IntendedPurpose'))
then let IMP: Set(MM!Class) = Set()
let AIP: Set(MM!Class) = src.getValue(src.getStereotype('IntendedPurpose'),
'\text{aip}')->iterate(el;locSet: Set(MM!Class)= Set()|locSet->union(el.base_Class.getAncestors())))
let PIP: Set(MM!Class) = (src.getValue(src.getStereotype('IntendedPurpose'),
'\text{pip}')->iterate(el;locSet: Set(MM!Class)= Set()|locSet->union(1.el.base_Class.getAncestorsAndDescendants()))))
in IMP->union(AIP->symetricDifference(AIP->intersection(PIP)))
else Set()
endif;
```

The helper $\text{getImpliedPurposes}$ receives the IntendedPurpose element under analysis (e.g., $\text{IP1}$) as input parameter.

6.2.2 Composite queries

Queries belonging to this category are interpreted by means of a stepwise process and are used to evaluate OCL expressions included in PaML models. The proposed technique allows interpreting OCL expressions defined in models that belong to any layer of the meta-layer structure (see Section 3). The interpretation is achieved through the composition of two ATL queries, denoted $\text{extractor}$ and $\text{evaluator}$, respectively.

The $\text{extractor}$ query selects the OCL expression to be evaluated and derives the values of the variables that compose the expression. These elements are composed to form the $\text{evaluator}$ query, whose interpretation returns the value of the original OCL expression.

Let us consider the composite query which allows for evaluating the condition specified by a ConditionalRole that constrains the assignment to users. The condition is an OCL Boolean expression built with attributes of Role and Environment.

The evaluation requires the initialisation of the constraint’s variables with the value of User and Environment attributes.
Example 11: Let us consider Example 4. A user of type PersonnelOffice23 belongs to ConditionalRole EMarketingCR in an Environment state envS, if attribute serviceType is set to 'UpdateInfo' and the value of attribute timeOfDay of envS is in the range [9..17] (see the OCL constraint in Listing 7).

\[(\text{serviceType} = \text{'UpdateInfo'} \land \text{timeOfDay} \geq 9 \land \text{timeOfDay} \leq 17)\] (7)

The extractor query derives the value of the attributes of the considered User and Environment, in our case Jack and envS (see Examples 3 and 4), as well as the constraint of the ConditionalRole EMarketingCR. The evaluator query is defined in such a way that:

1. it defines variables with the same name and type of the involved ConditionalRole attributes initialising them with the values extracted from the referred instances of User and Environment
2. it defines the constraint derived from the ConditionalRole as the target of the evaluation.

Listing 8 shows the evaluator query defined for our example.

```sql
query EvalConditionalRoleExpression =
let serviceType: String = 'ReadInfo'
let timeOfDay: Integer = 9
in (serviceType = 'UpdateInfo'
and timeOfDay >= 9
and timeOfDay <= 17) (8)
```

The execution of the evaluator query derives the value of the expression. In our case, based on the values of serviceType and timeOfDay, `EvalConditionalRoleExpression` is not satisfied.

7 Early performance assessment

In this section, we discuss an early performance evaluation of MAPaS analysis functionalities that target SQL queries. Our tests have been achieved with ten queries accessing up to four tables per query and five attributes per table. The target DB scheme is composed of five tables each including five attributes. The considered purpose set is the same presented in Section 4.

Most of the analysis execution time is spent for the parsing of SQL queries with the aim to derive the tables and the columns that are effectively accessed by a query. All MAPaS functionalities related to the analysis of privacy policies and SQL queries (i.e., MAPaS functions 15–23 in Table 2) rely on these parsing functionalities. The worst case that we have experimented is related to the parsing of a complex query including two sub-queries each in turn including a sub query. However, even in this case the analysis time is limited to few seconds, thus admissible performance are guaranteed. In addition, it is reasonable to assume that the great majority of queries do not include more than two layers of nested sub-queries. Although performance aspects of ATL queries are not comparable with those of SQL queries in relational DBMSs, which are optimised for speed, we have measured good performance results. Without
considering parsing activities, the execution time of the functions 15–23 in Table 2 has always been in the order of some fractions of seconds for all the considered queries. We believe that this is due to the limited number of elements that characterise a PaML model. The cardinality of the purpose set covers a fundamental role, as each policy compliance check requires to access the purpose tree. Our tests have been achieved with the purpose set specified in Byun and Li (2008), which represents a de-facto benchmark for purpose-based access control models as it is re-used in numerous other papers (e.g., Kabir and Wang, 2009; Kabir et al., 2010; Colombo and Ferrari, 2014). The considered purpose set includes 15 elements, and it is reasonable that even in realistic case studies the purpose set should not include more than few tens of elements and the purpose tree should not overcome four to five layers. In respect to these considerations, we believe that overall our tests have shown satisfactory performance and we do not expect that these results can be significantly modified by varying the application scenario.

This early assessment precedes a thorough evaluation of MAPaS performance that we plan to achieve as future work. In particular, we plan to analyse performance with respect to a variety of SQL queries, and database schemas.

8 Conclusions and future work

In this paper we have presented an extended version of MAPaS which supports the definition of purpose and role-based access control policies that regulate the execution of SQL queries in relational DBMSs, and the development of privacy aware SQL queries. MAPaS integrates a rich set of functionalities allowing to reason on privacy awareness of queries at database definition time before the database is populated. MAPaS functionalities presented in this work have been primarily developed with the goal to prove the feasibility of the analysis. However, we are currently working to the reengineering of the current prototype targeting the restyling of the graphical user interface finalised to maximise the framework usability, and the optimisation of performance and reliability aspects. Once completed this task, we will release the application to selected working groups from industry with the aim to get early users acceptance ratings and comments for possible future modifications of MAPaS functionalities.

References

A framework for privacy aware data management in relational databases


http://www.w3.org/Submission/EPAL/ (accessed 5 May 2014).

**Notes**

1. The attributes of *Writer* are inherited from the Roles *Employee*, *MarketingDept* and *EMarketing*.

2. The constraint is defined based on MyCompany’s internal regulations.

3. We have implemented them using the parsing engine provided by the SQL General Parser library.