

Introducing Privacy in a Hospital Information System

Stefano Braghin Alberto Coen-Porisini Pietro Colombo
Sabrina Sicari Alberto Trombetta

Dipartimento di Informatica e Comunicazione – Università degli Studi dell’Insubria
via Mazzini 5, 21100 Varese, Italy
{stefano.braghin, alberto.coenporisini, pietro.colombo,
sabrina.sicari, alberto.trombetta}@uninsubria.it

ABSTRACT

Security and privacy issues in healthcare data management play a fundamental role in the widespread adoption of medical information systems. As a consequence, it is very important to define the right means for expressing and managing policies in order to comply with privacy-related standards and regulations.

In this work, we extend an open source hospital information system in order to provide support for expressing and enforcing privacy-related policies, using as a starting point a conceptual model the authors developed in a previous work.

Categories and Subject Descriptors

J.3 [Life and Medical Sciences]: *Medical information systems.*

K.4.1 [Public Policy Issues]: *Privacy*

General Terms: Security

Keywords: Privacy policies, conceptual models, software engineering applications, hospital information system

1. INTRODUCTION

In the last few years, hospitals have increasingly adopted Information Technology-supported healthcare solutions in order to manage health-related information and to provide a (semi)automated administration of clinical functions.

Usually, healthcare-related data of patients are stored in a digital version of medical/health records, called Electronic Medical/Health Records (EMR/EHR) and managed by corresponding medical information systems that enable communication of patients’ data among healthcare professionals.

Sharing sensitive patients’ data in a large, distributed and heterogeneous environment introduces security and privacy risks. Such risks are further enhanced by the inherent openness of the web-based applications and interfaces through which medical information systems can be accessed. As a consequence, it is widely recognized that security and privacy concerns are the main obstacles to the deployment of medical information systems. On the other side, the relevance of such concerns is proved by the activities of regulatory bodies. For instance, in the US, the Health Insurance Portability and Accountability Act (HIPAA)[3] established standards for the

security of digital healthcare information, while in the European Union, the Directive for personal data [1] acts similarly.

In order to comply with such standards and regulations, healthcare organizations have to define suitable management processes, which often entail the publication of privacy policies (e.g., on websites) intended to inform patients about the management of their data. Such policies are expressed in a very high-level language and have to be translated into privacy policies expressed in a proper formal language before being applied for access control at the implementation level.

In order to address the shortcomings of traditional access control systems, which usually lack support for privacy-related policies, in [5] a privacy-oriented extension of the well-known RBAC model has been defined. Choosing RBAC as a starting point has some advantages since one can map roles – which are an important indirection between users and permissions – directly onto healthcare organizational positions such as physicians, nurses, administrative staff, etc. In [4] we presented an UML-based conceptual model providing a sound basis for the definition and enforcement of privacy policies

At the moment, however, there seems to be a gap between the functionalities offered by standard, off-the-shelf healthcare information systems and the requirements regarding health-related data privacy, as stated by regulatory bodies. In this paper we use the approach and the techniques presented by the authors in [4], for extending a well-known open source healthcare information system in order to express, manage and enforce privacy policies.

The paper is organized in the following way: Section 2 introduces the privacy model and discusses its main features; Section 3 presents an application scenario in the healthcare domain; Section 4 discusses the related works, while Section 5 draws some conclusions and provides hints on the future work.

2. MODELING PRIVACY

A privacy policy defines the way in which data referring to individuals can be collected, processed and diffused according to the rights that individuals are entitled to.

The rest of the paper adopts the terminology introduced by the EU directive [1], which is summarized in what follows:

- *personal data* means any information relating to an identified or identifiable natural person (referred to as *data subject* or *subject*).
- *processing of personal data (processing)* means any operation or set of operations which is performed upon personal data, whether or not by automatic means, such as collection, recording, organization, storage, adaptation or alteration,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SESS’08, May 17–18, 2008, Leipzig, Germany.

Copyright 2008 ACM 978-1-60558-042-5/08/05...\$5.00.

retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, blocking, erasure or destruction;

- *controller* means the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes and means of the processing of personal data;
- *processor* means a natural or legal person, public authority, agency or any other body which processes personal data on behalf of the controller;
- *the data subject's consent (consent)* means any freely given specific and informed indication of his/her wishes by which the data subject signifies his/her agreement to personal data relating to him/her being processed.

As a distinctive feature of a privacy policy, the processor is required to state for what *purpose* data are processed. A purpose can be defined either as a high-level activity (e.g., “marketing”, “customer satisfaction”) or as a set of actions (e.g., “compute the average price”, “evaluate the customer needs”). Moreover, an *obligation* is a set of actions that the processor guarantees to perform, after the data have been processed. Also obligations have to be stated by the processor. Subjects, whenever their data are collected, must be informed of the purposes and of the obligations related to any processing. Moreover, subjects must grant their consent before any processing can be done. Finally, the consent can be given selectively that is, a subject can grant the consent for one purpose while denying it for another one.

The conceptual model proposed in [4] is described by means of UML and it provides a clear and simple way for representing every concept occurring in a privacy policy along with their relationships. Furthermore, this approach provides a straightforward way towards the enforcement of privacy policies.

2.1 The UML Model

In the following we give a short overview of the basic features that are relevant for the present work. Due to space limitations, we omit the discussion on the behavioural features of our model, which can be found in [4].

The structural aspects are defined using UML classes and their relationships such as associations, dependencies and generalizations. Figure 1 depicts a class diagram that provides a high level view of the basic structural elements of the model.

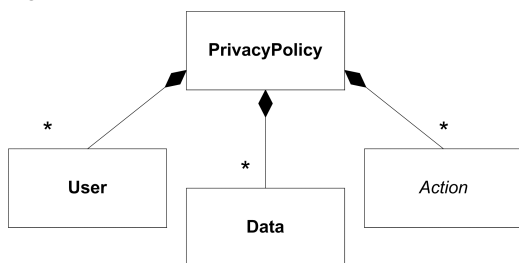


Figure 1: The Privacy Policy Class Diagram

A *PrivacyPolicy* is characterized by three types of classes: *User*, *Data* and *Action*. Users interact among them in order to perform some kind of action on data that refer to other users. Thus, an instance of *PrivacyPolicy* is characterized by specific instances of *User*, *Data* and *Action*, and by the relationships among such entities. Let us focus on the classes introduced by the diagram:

- *User* represents an actor either interested in processing data or involved by such processing. Since role [4] is a key concept of this approach, users are characterized depending on the role they play. *User* is extended by means of three distinct classes to represent the different roles: *Subject*, which is anyone whose data are referred to, *Processor*, which is anyone who asks for processing data by performing some kind of action on them and *Controller*, which defines the allowed actions that can be performed by the processors.
- *Data* represents the information referring to subjects that can be processed by processors. *Data* is extended by means of *Identifiable* data (e.g., name, address, phone n.) and *Sensible* data (e.g., health, religion). The former represents the information that can be used to uniquely identify subjects, while the latter represents information that deserves particular care and that should not be freely accessible.
- *Action* represents any operation performed by *User* (usually *Processor*). *Action* has been defined using an abstract class and it is extended by *Obligation*, *Processing* and *Purpose*. Moreover, each action can be recursively composed of purposes and obligations and therefore it is defined by means of an aggregation relationship between *Action* and both *Purpose* and *Obligation*.

Figure 2 depicts the aforementioned entities along with their relationships. For instance, the dependency relationship between *Action* and *Data* means that data are processed by actions, while the association between *Subject* and *Data* expresses data ownership.

Notice that this model can be extended in order to support the definition of policies related to different application domains. For example, to specify privacy policies compliant with the Italian privacy legislation [2], it is necessary to extend the model introducing the concept of “judicial data”. Such extension can be easily realized by introducing a class *Judicial* that extends the class *Data*.

All the above entities interact among them exchanging information through interaction points represented by means of interfaces. Thus, an interface defines the services that a class can either implement or use (invoke). The model introduces the following five interfaces:

- *ConsentRequest*, which specifies the method *notify()*, that taken an instance of *Obligation*, an instance of *Purpose* and the *id* of *Controller*, notifies the subject of both the purposes and the obligations of the data processing. *ConsentRequest* is implemented by class *Subject* and is used by class *Controller*.
- *ConsentAcquisition*, which provides the method *grant()*, that taken an instance of *Purpose*, an instance of *Obligation*, the *id* of *Subject* and a boolean value, specifies whether the subject has granted the consent for processing his/her data. *ConsentAcquisition* is implemented by class *Controller* and is used by class *Subject*.
- *Control*, which provides the method *verify()* that, taken an instance of *Action*, returns whether the performed action was authorized that is, the consent has been granted. *Control* is implemented by class *Controller* and is used by class *Action*.
- *FactoryAction*, which provides the services to instantiate an *Action*. It is implemented by class *Controller* and is used by class *Processor*.

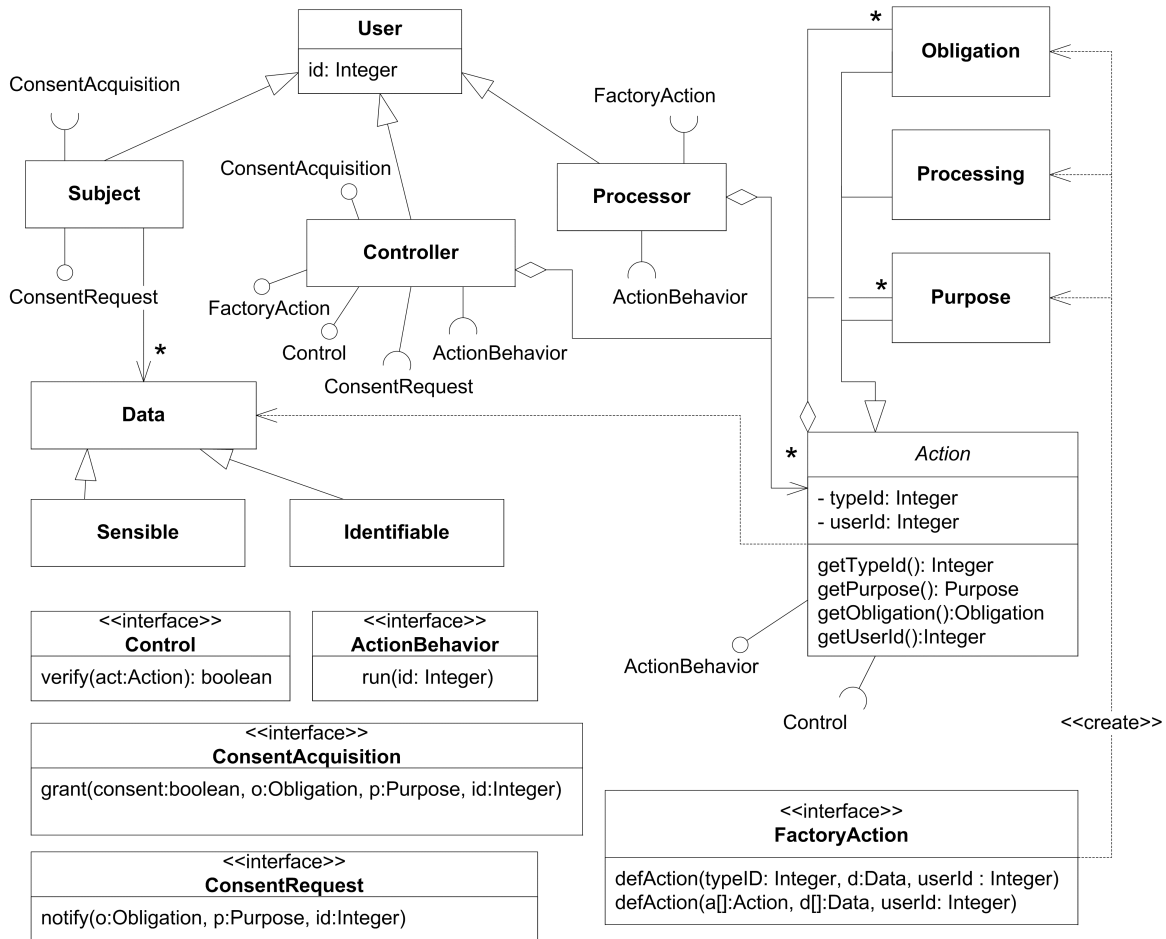


Figure 2: Privacy Class Diagram

- *ActionBehavior*, which provides the method *run()* that represents the execution of an action.

Notice that, since the interface *ActionBehavior*, which defines the method *run()*, is provided by *Action*, each class extending *Action* can provide a specific implementation of such method.

3. A CASE STUDY

Healthcare information systems are required to provide support for managing privacy policies, as prescribed by a growing body of *ad-hoc* legislation. In the following, we present a well-known open source hospital information system named Care2x [18], we discuss its weaknesses in dealing with privacy-related issues and we present a set of extensions in order to overcome such weaknesses.

3.1 Care2x

Care2x is an open source hospital information system created in year 2000 by the initiative of Mr. Elpidio Latorilla, a nurse interested in automating hospital information processes. The project was released to the open source community on May 2002 under the GNU General Public License. Care2x is currently supported by a team of more than three hundreds programmers, and its graphical user interface has been translated into 16 languages.

Care2x has been deployed in more than 20 countries. For instance, in 2004, the Malaysian Ministry of Health started experimenting the usage of Care2x in several hospitals, while the Brazilian Government financed the configuration of Care2x to support the national healthcare system. In Italy the Care2x experience involves relevant healthcare organizations, such as Policlinico Umberto I in Rome and the National Institute of Sport Medicine (INMS).

Care2x platform comprises the following four major independent components:

- HIS - Hospital/Healthservice Information System is a comprehensive, integrated information system designed to manage the administrative, financial and clinical aspects of a hospital.
- PM - Practice Management is an information system designed to support the management of practices of medical departments and clinics.
- CDS - Central Data Server is a central data repository that can be used by all the organizations involved in the healthcare environment. CDS is accessed via the Health Xchange Protocol (HXP).

- HXP - Health Xchange Protocol HXP is the standard data exchange protocol used by Care2x to communicate with other healthcare applications. HXP ensures data exchange among healthcare applications possibly operating into different environments.

HIS is the core application of Care2x and it provides the basic functionalities required by all healthcare systems and thus it is the ideal candidate for experimenting our privacy modelling approach.

3.2 Care2x HIS

Care2x HIS is a web based modular and scalable application that integrates different types of services, processes, and data used in a healthcare system.

Care2x is implemented in PHP 4 [19], using standard relational DBMSs and it can be deployed by means of common web servers such as Apache Web Server or Microsoft IIS. The default installation exploits XAMP, an easy to install platform composed of a pre configured version of Apache web server that includes a PHP4 engine and a MySQL DBMS. XAMP allows the application to be installed and used in most Win/Mac/Unix based system.

Care2x HIS is a client/server web based application. The client side of the application provides a communication layer through a highly configurable GUI. All the computational tasks are performed on the server side, where services, elaborations, data processing and data retrieving are executed. Notice that thanks to the standard communication infrastructure of XAMP, the server side can be configured as a distributed system.

The application is accessible through a common web browser supporting Css and JavaScript. Depending on configuration, HIS can be used either as a stand alone application or as a real distributed system operating on any TCP/IP compliant network infrastructure.

Care2x architecture can be thought as a 4 layers structure. The first layer is made up of a database that stores all the information used by the platform. The second layer provides DBMS-independent services to access and manipulate the data stored in the database. Care2x uses ADOdb [20], an interface that hides to the developer all the details concerning the currently used DBMS. All the services for accessing and manipulating data are handled through ADOdb and SQL. Notice that this solution may allow one to use any available relational DBMS. So far, known configurations of Care2x HIS experimented the usage of MySQL, PostgreSQL, DB2 and Oracle.

The third layer comprises several libraries providing the services required for the correct behavior of the fourth layer, such as user authentication, internationalization, GUI configuration.

The last layer includes a series of modules for managing processes and workflows. For instance, a module deals with patients admission, while another deals with laboratory analysis. Each module is completely independent from the others and it exploits the underneath layer in order to access and execute basic functionalities and services that may involve data access and manipulation.

HIS extensions generally involve the definition of innovative modules designed to satisfy the requirements of a specific department of a health care institute. The first three layers are stable and common to any Care2x setup, while the last one may

change from one configuration to another. For instance a new configuration may contain some new modules, developed *ad hoc* to satisfy some domain dependent specific requirements.

3.3 Weaknesses

Care2x is characterized by several weak points that need to be examined in order to identify possible exploitable paths for privacy violations. As a consequence Care2x cannot be considered a privacy-aware system. These vulnerabilities are mainly due to the lack of role based functionalities.

As mentioned before, the implementation of a privacy-aware system benefits from a sound role management mechanism, not yet provided by Care2x. In fact, although Care2x supports the definition and assignment of new roles, it does not provide any mean to define the actions for handling patients' data associated with each role.

According to [4] each role should be associated with a list of processing actions that can be performed along with the purposes of such actions. Moreover, roles can be used as filters for selecting actions that can be associated with actors belonging to the system. However, this kind of solution is not implemented in Care2x and, furthermore, Care2x does not provide any mechanism to verify the correct application of actions performed within the system and their compliance with any specific privacy policy nor there is any enforcement mechanism.

In the following, we present how to extend Care2x in order to overcome such deficiencies.

3.4 Extending Care2x HIS

This section introduces an approach for extending Care2x HIS, starting from the identified weaknesses and the privacy model proposed in a prior work [4].

More specifically, the extensions concern both the definition of a role-based privacy management mechanism, and the definition of enforcement techniques that aim at verifying the compliance of the actions performed with the existing privacy policies.

The preliminary analysis of the platform architecture suggests to focus on the first and the third layers, since the last one simply provides the applicative functionalities while the second one is a standard interface for accessing the database, and therefore it should not be modified. Therefore, the extensions of Care2x involve two distinct aspects:

1. Data-layer extensions: it is necessary to add concepts taken from the conceptual model (e.g., Role, Action), that are necessary to define any specific role based privacy mechanism. Notice that these extensions involve the first layer of the architecture.
2. Services extensions: new services are introduced in the libraries of the third layer of the architecture. Notice that these services exploit the extensions introduced in the first layer.

The rest of the section discusses the main extensions introduced.

3.4.1 Role-based privacy management

Role based management mainly requires the introduction of two different types of conceptual elements: roles and actions. According to the conceptual model proposed in [4], three main roles, named *Subject*, *Processor* or *Controller* are identified. Each of them is independent from any specific context.

Let us consider a complex organization such as a hospital, in which many employees work with different functions (e.g., doctors, nurses, administrative staff, etc.). The first step for the definition of roles consists in classifying the people that operate in the hospital according to the functions they perform therein [17]. Notice that, functions cannot be considered as roles, since the latter are related to privacy policies, while the former are related to the tasks performed by employees. As a consequence roles depend also on actions and are not exclusively related to functions. For example an employee having the function of doctor could play as *Controller* for a given action and as *Subject* for another one.

Therefore, it is necessary to analyze the relationships between the different functions (nurse, doctor, etc) in order to identify the role associated with each function interacting with the information system. More specifically it is necessary to define a function hierarchy in order to provide a dynamic role assignment.

In turn, functions are characterized by a set of actions. An action can be either enactive or declarative. The former includes actions that require to access and process data while the latter includes simple statements representing activities that do not require to interact with the system (e.g., send a letter, make a phone call, etc).

Notice that enactive actions are traceable since at run-time they interact with the information system, while declarative actions can not be traced.

For example, let us consider an hospital in which the employees may have one of the following functions: Head physician, Physician, Head nurse, Nurse.

The above function hierarchy imposes a corresponding hierarchy on the actions associated with functions. For instance, the actions performed by a head nurse can be done by a nurse, since function Head nurse extends function Nurse.

Moreover, let us consider a physician that during a medical examination identifies a new symptom in a patient. The physician performs the following actions: 1) Modify the health record (of the patient) and 2) Notify the Head physician (of the new symptom). Notice that action 1) requires to access the information system, while action 2) does not. Therefore, the former action is enactive, while the latter is declarative.

Extending the data-layer

Care2x HIS does not support role management nor the *Action* concept. In Care2x the term *role* simply refers to the function of employees. The *function* concept is introduced by means of an entity, named *care_role_person*, that contains a list of predefined functions such as physician, nurse, etc.

The function hierarchy is introduced by defining a new entity named *care_function_hierarchy*. The entity defines: 1) a hierarchical structure, and 2) a set of actions associated with each function.

According to the conceptual model (see Figure 2), actions are complex structures that may be decomposed into simpler ones. In order to model the concept of action structure the Composite design pattern is adopted (see Figure 3).

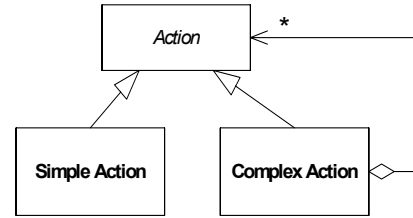


Figure 3: Relationships between different types of actions

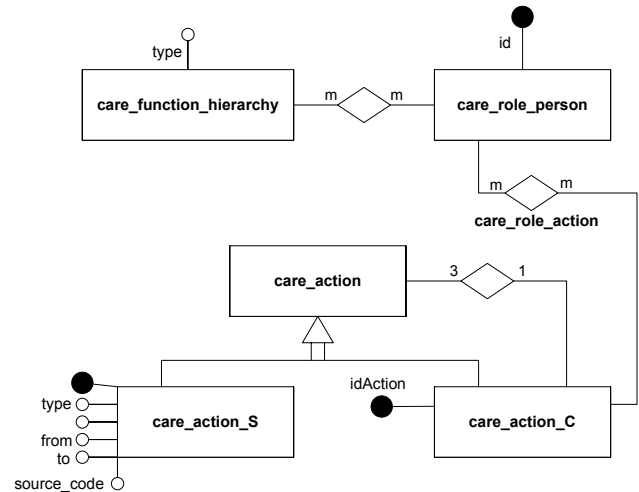


Figure 4: The extended data layer for supporting role management

At the data level, actions representation is based on the introduction of three new entities, named *care_action*, *care_action_S* and *care_action_C*, respectively.

- *care_action* represents the *Action* concept.
- *care_action_S* represents the simple actions that can be composed into more complex ones. Simple actions exploit the classification proposed in the conceptual model, that is actions are tagged as *Purpose*, *Processing* or *Obligation*.
- *care_action_C* represents complex actions and are represented using a recursive hierarchical definition, that is each action is composed of other actions that can be either simple or complex.

Referring to previous example, the physician performs the following actions:

1. *Purpose1*: Diagnostics (*care_action_S1*)
2. *Processing1*: Modify the health record (*care_action_S2*)
3. *Obligation1*: Notify the head physician (*care_action_S3*)

The resulting set of actions, *care_action_S1*, *care_action_S2* and *care_action_S3*, is represented by means of *care_action_C1*.

Finally, in order to complete the definition of the role concept, it is necessary to explicitly associate functions with actions.

As an example, the nurse function could be associated with the following actions: read patient personal data; modify and read medical/health records.

The relationship between functions and actions is expressed by means of the relation *care_role_action*.

Figure 4 shows a E-R diagram that presents the data layer enhancement.

Extending the control layer

The extension of the control layer starts from the analysis of the same conceptual elements introduced in the data layer and it is essentially focused on the introduction of classes *Role* and *Action*, and of some additional classes working as adapters for the currently existing interface.

Class *Action* reflects the characteristics of the homonymous element introduced in the conceptual model (see Figure 2). In particular, it implements the methods defined in the *ActionBehavior* interface. The implementation uses ADOdb for accessing the data required for executing the action. More specifically, the method *run*, which represents the core function, is implemented by accessing the relation *care_role_action* in which all the relations between functions and actions are stored. Therefore, entities *care_role_person*, *care_action_C*, and *care_action_S* are accessed in order to retrieve all the information required to check the execution of the current action.

Class *Role* represents another important extension to the existing control layer. More specifically, the instances of class *Role* specify a list of actions that defines the admitted or required behaviour for the instance.

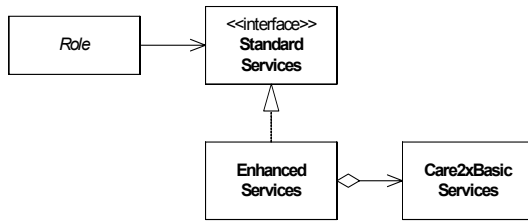


Figure 5: Extensions to the existing services

Both *Role* and *Action* may require to interact with the existent library in order to access some of the basic services provided by the system. Such access is filtered by software adapters that aim at keeping the existing interface extending the provided functionalities without modifying the existing code. Extensions are defined according to the design pattern Adapter as shown in Figure 5.

3.4.2 Supporting enforcement mechanisms

Privacy policy enforcement consists in verifying the compliance of the actions performed by users with a given privacy policy. According to the conceptual model, there are two different ways in which such verification can be carried out. The first one consists in providing *ex-post* enforcement mechanisms, that is, all the checks are done after all the actions have been performed (e.g., audit-based mechanisms). The second one consists in having run-time enforcement mechanisms, that is, the effect of every action is checked as soon as the action is executed.

In order to support both techniques, it is necessary to further extend the existing data and control layers.

The data layer extension concerns the introduction of logging mechanisms that keep track of all the actions that are executed, while the control layer extension concerns the verification of the compliance of such actions with the privacy policy.

Extending the data layer

In order to implement the logging mechanism required to support the enforcement of privacy policies, two entities named *care_log_action_S* and *care_log_action_C* are introduced.

- *care_log_action_S*, which traces the execution of simple actions and whether the executed action complies with the current policy;
- *care_log_action_C*, which provides logging support for complex actions.

Referring to the previously introduced example, when the physician interacts with the system to modify the health record of his/her patient a new log entry, *care_log_action_C₁*, is created. Such entry refers to the complex action *care_action_C₁* that is composed of three simple actions. Therefore a log entry, named *care_log_action_S* is created to represent that the physician acts according to *Purpose₁* (i.e., Diagnostics). In the same way when he/she actually modifies the health record another log entry, named *care_log_action_S₂* is created. Finally, when the system notifies the physician that he/she must comply with *Obligation₁*, the last log entry, *care_log_action_S₃* is created. Notice that each time a simple action (*care_action_S_{1...3}*) is logged, *care_log_action_C₁* is updated with the reference to the corresponding simple log entry (*care_log_action_S_{1...3}*).

The *ex-post* enforcement can be carried out by verifying that *care_log_action_C₁* correctly refers to the log entries associated with the corresponding simple actions

Figure 6 shows the E-R diagram that presents the data layer enhancement.

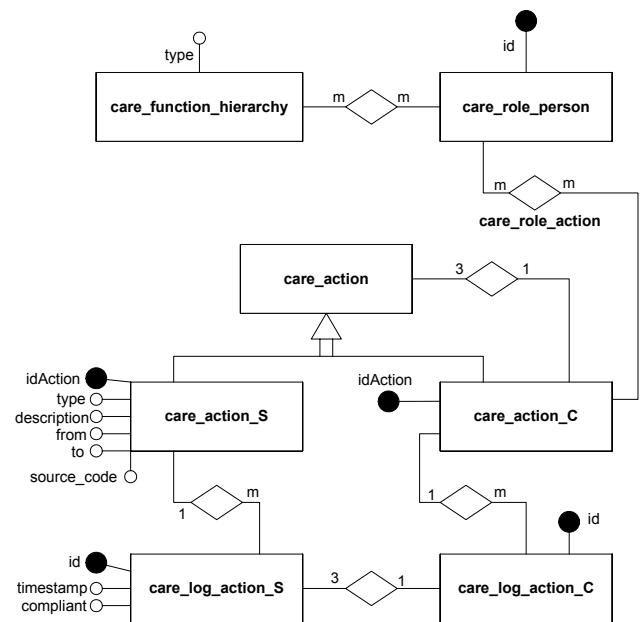


Figure 6: The extended data layer for supporting enforcement

Extending the control layer

According to the proposed conceptual model, all actions required by a privacy policy are defined as instances of classes *Purpose*, *Obligation* and *Processing*, which are extensions of the abstract class *Action* (see Figure 2).

Class *Action* requires the interface *Control* that, in turn, defines the method *verify()* to represent the verification of the compliance of any instance of *Action* with a given policy.

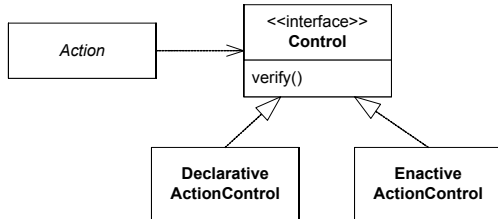


Figure 7: Extensions using the pattern Strategy

The interface *Control* is realized through different implementations. We initially provide a basic implementation for each type of action introduced in the previous sections. As shown in Figure 7, a dedicated implementation of the control service is defined according to the design pattern Strategy.

All the verification tasks are carried out by *Controller*, while actions are executed by *Processor*. Starting from these basic points, different processes can be defined by composing method calls provided by these classes in different order. As an example, when *Processor* executes an action (i.e., it invokes the method *run()*), the method *verify()* is invoked, thus allowing the *Controller* to verify that *Action* is compliant with the policy. As a consequence, in case of non-compliance, the *Controller* can prevent *Processor* from executing any further action.

4. RELATED WORKS

While research on security is a well-established field, the issues that arise when dealing with privacy have been under thorough investigation only in the recent years. The research efforts aiming at the protection of individuals privacy can be partitioned in two broad categories: Security-oriented Requirement Engineering (SRE) methodologies and Privacy Enhancing Technologies (PETs). The former focuses on methods for taking into account security issues (including privacy) during the early stages of systems development, while the latter describes techniques to ensure privacy.

Several existing requirement engineering methodologies, such as Kaos[6], Tropos[7][8][9], NFR[10][11] and GBRAM[12], can be used to take into account security issues at design level. In [13] the authors present a methodology, called PRIS, to incorporate privacy requirements into the system design process. PRIS is a requirement engineering methodology focused on privacy issues. It provides a set of concepts to model privacy requirements and a set of rules to transform requirements into implementation techniques.

All the above methodologies address the problem of how to state as clearly as possible the requirements that an information system must satisfy in order to be considered secure (with respect to a set of given security policies). This is different from our goal, which is to define a conceptual model for representing privacy policies. As explained in Section 2, this is achieved

through the deployment of a model that represents all the relevant concepts of privacy related policies.

In [14] extensions to a RDBMS are provided in order to express P3P privacy policies, at schema definition level. Furthermore, the authors define mechanisms for translating P3P privacy policies into a properly extended SQL-like data definition language. This is different from our approach, since what we propose is a conceptual model for the definition of privacy policies (not to be necessarily expressed in P3P language) and for the specification of the needed functional modules of an application in order to enforce such policies. Concerning medical information systems, to our knowledge there is no proposal aiming at a unified treatment of privacy-related policies, as presented in this work.

Finally, in the field of SRE methodologies, several techniques have been proposed in order to protect private data from unauthorized users. Typical examples are anonymizing techniques based on data suppression or randomization[15][16]. However, these techniques do not require the definition of any privacy policy; rather they can be used as building blocks for realizing them.

5. CONCLUSIONS

Privacy is a fundamental issue in hospital information systems. In fact most of the involved data are sensible and therefore their access and manipulation is strongly regulated.

In this work we have presented extensions to the Care2X medical information system suitable for expressing, managing and enforcing privacy policy, as specified by the conceptual model presented in [4]. Future work will address relevant features such as the possibility to express and manage complex hierarchies of actions and functions, along with their related policies.

REFERENCES

- [1] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 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 of 23 November 1995 No L. 281 p. 31*
- [2] Decreto Legislativo n. 196, 30 Giugno 2003, Codice in materia di protezione dei dati personali, *Gazzetta Ufficiale n. 174 del 29-7-2003 - Suppl. Ord. n. 123*
- [3] <http://www.hipaa.org>
- [4] A. Coen-Porisini, P. Colombo, S. Sicari, A. Trombetta. A Conceptual Model for Privacy Policies. In *Proc. of Software Engineering Application (SEA'07)*, Cambridge, Boston, 2007.
- [5] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo. Privacy-aware Role-Based Access Control. In *Proc. of ACM Symp. on Access Control Methods And Technologies (SACMAT'07)*, 2007.
- [6] A. V. Lamsweerde and E. Letier. Handling Obstacles in Goal-Oriented Requirement Engineering. *IEEE Trans. Soft. Eng.* 26:978–1005, 2000.

- [7] L. Liu, E. Yu, and J. Mylopoulos. Analyzing Security Requirements as Relationships among Strategic Actors. In *SREIS'02, e-proceedings*, Raleigh, 2002.
- [8] H. Mouratidis, P. Giorgini, and G. Mason. Integrating Security and Systems Engineering towards the Modelling of Secure Information System. In *15th Int. Conf. of Advanced Info. System Engineering (CAiSE'03)*, vol. 2681 of *LNCIS*, pages 63–78. Springer-Verlang, Berlin, 2003.
- [9] H. Mouratidis, P. Giorgini, and G. A. Manson. An Ontology for Modelling Security: The Tropos Approach. In V. Palade, R. J. Howlett, and L. C. Jain, editors, *KES*, vol. 2773 of *Lecture Notes in Computer Science*, pages 1387–1394. Springer, 2003.
- [10] L. Chung. Dealing with Security Requirements during the Development of Information System. In *5th Int. Conf. of Advanced Info. System Engineering (CAiSE'93)*, Paris (France).
- [11] J. Mylopoulos, L. Chung, and B. Nixon. Representing and Using non Functional Requirements: a Process Oriented Approach. *IEEE Trans. Soft. Eng.*, 18:483–497, 1992.
- [12] A. Anton. Goal-Based Requirements Analysis. In *2nd IEEE Int. Conf. on Requirements Engineering (ICRE'96)*, pages 136–144, Colorado Springs Co, 1996.
- [13] E. Kavakli, C. Kalloniatis, P. Loucopoulos, and S. Gritzalis. Incorporating Privacy Requirements into the System Design Process. The PRIS Conceptual Framework. *Internet research*, 16:978–1005, 2006.
- [14] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, and W. Rjaibi. Extending Relational Database Systems to Automatically Enforce Privacy Policies. In *ICDE*, pages 1013–1022. IEEE Computer Society, 2005.
- [15] T. Mielikinen. Privacy Problems with Anonymized Transaction Databases. In *7th Int. Conf. Discovery Science (DS 2004)*, Lecture Notes in Computer Science.
- [16] A. Narayanan and V. Shmatikov. Obfuscated Databases and Group Privacy. In *12th ACM conference on Computer and communications security (CCS '05)*, pages 102–111, New York, NY, USA, 2005. ACM Press.
- [17] Legislazione Sanitaria e Sociale, Edizione giuridiche Simone, 2006, ISBN 88-244-7728-3
- [18] <http://www.care2x.org/>
- [19] <http://www.php.net/>
- [20] <http://www.adodb.sourceforge.net/>