Privacy Preserving Event Driven Integration for Interoperating Social and Health Systems

Giampaolo Armellin[1], Dario Betti[1], Fabio Casati[2], Annamaria Chiasera[1,2], Gloria Martinez[2], and Jovan Stevovic[1,2]

[1] GPI SpA, Via Ragazzi del '99, 13,
Trento, Italy
{garmellin,dbetti,achiasera,jstevovic}@gpi.it
[2] Information Engineering and Computer Science Department,
University of Trento, Italy
casati@disi.unitn.eu, glomarin@gmail.com

Abstract. Processes in healthcare and socio-assistive domains typically span multiple institutions and require cooperation and information exchange among multiple IT systems. In most cases this cooperation today is handled "manually" via document exchange (by email, post, or fax) and in a point-to-point fashion. One of the reasons that makes it difficult to implement an integrated solution is that of privacy, as health information is often sensitive and there needs to be a tight control on which information is sent to who and on the purpose for which it is requested and used. In this paper we report on how we approached this problem and on the lessons learned from designing and deploying a solution for monitoring multi-organization healthcare processes in Italy. The key idea lies in combining a powerful monitoring and integration paradigm, that of event bus and publish/subscribe systems on top of service-oriented architectures, with a simple but flexible privacy mechanism based on publication of event summaries and then on explicit requests for details by all interested parties. This approach was the first to overcome the privacy limitations defined by the laws while allowing publish/subscribe event-based integration.

Keywords: interoperabilty, SOA, EDA, privacy enforcement.

1 Introduction

Recently we are assisting to a strong commitment of the public administrations toward e-government projects focused mainly on the dematerialization of the administrative processes, to monitor, control and trace the clinical and assistive processes with a fine-grained control on the access and dissemination of sensitive information. Such processes involve both private and public organizations from the social and healthcare domain providing a variety of services, from food delivery to house cleaning, telecare and nursing outside the hospitals. Typically these processes span multiple institutions and to support their integration or, as a minimum, to monitor their execution, some degree of integration among the IT systems of these institutions is necessary.

Application Integration poses many well-known problems in terms of bridging the different protocols and data models of the various systems involved. In this paper we study these problems from a privacy perspective. Specifically, our goal here is to report on the problems, solutions, and lessons learned from the development and deployment of a large integration project called CSS, aimed at monitoring healthcare and social processes across the different government and healthcare institutions in the region of Trentino, Italy. Trentino was used as a pilot region in Italy, and the results will next be applied at the national level.

The project allows the **cooperation** of applications from different agencies (both public and private) to provide high quality clinical and socio-assistive services to the citizens. The institutions to be integrated range from small civic centers to the regional government. As such, they are very high in number and very heterogeneous in nature. Furthermore, institutions progressively join the integrated CSS process monitoring ecosystem, so that an additional challenge lies in how to facilitate the addition of new institutions. From a data privacy perspective, these kinds of scenarios present tough challenges, such as:

– how to make it simple for all the various data sources to define the privacy constraints and enforce them. Simplicity here is the key as otherwise the complexity of the information system and in some cases also of the law make the problem already hard to tackle, and unless the privacy model remains really simple it is very hard to understand which information is protected and to what extent.
– how to achieve patient/citizen empowerment by supporting consent collection at data source level (opt-in, opt-out options to share the events and their content)
– how allow monitoring and tracing of the access request (who did the request and why/for which purpose?)
– how to support the testability and auditability of privacy requirements: owners of data sources often require that the privacy rules they are asked to define can be tested and audited so that they can be relieved of the responsibility of privacy breaches.

In the paper we show how we addressed these challenges, the lessons we learned, and the extent to which they can be generalized. In a nutshell, the solution we adopted is based on the following observations and decisions:

– First, we integrate systems based on events. Performing process analysis via a traditional data warehousing approach is not feasible as it would be too complex to dive into each of the information sources because of their number and diversity. Instead, we ask each source to model and distribute only the events which are relevant from the perspective of the process to be monitored. In this way the sources can focus only on getting the information needed for those events.
– Second, we realized that each data source owner wants to define their own privacy rules (even if they have to obey the same law). Hence, there cannot be a single institution acting as centralized data collector (e.g. the one that manages the CSS platform) that can define the privacy rules. However, defining the privacy rules on the source information system is complex and requires technical expertise that we cannot assume the data source owner (e.g. administrative and medical staff) has. We address this problem by defining privacy rules on events that, as we will see, not

only makes specifications relatively simple, they also avoid the danger of having over-constrained privacy specifications.

– Third, in many cases consumers do not need all the details that are generated by a data producer. We want to be able to define and control in a fine-grained manner which data is delivered to consumers. Privacy rules are dependent on which institutions is consuming the data. One of the problem in this case is who will decide which data can be transferred from the data sources to third party entities. We address this via a fine-grained, purpose-based, two-phase access control mechanism where each source initially exposes (publishes to the consumer) events that are only partially specified, just enough for the consumer to understand if they need more detail. Then, each interested consumer must explicitly request the details to the source (along with a purpose statement), via the CSS platform. This protocol allows sources to specify different visibility rules based on explicitly stated purposes, which makes the approach compliant with privacy regulations.

We have experienced that these ingredients taken together make the system easy to use and also legal from a privacy perspective, as well as easy to audit and test. In the following we describe the scenario in more detail and then present the solution that CSS provides.

2 The Scenario

Figure 1 shows the main actors involved in the social-health scenario and the flows of communication among them that today occurs mainly on paper. In this scenario, a patient needs healthcare and socio-assistive assistance (socio-health care services). The patient interacts with healthcare and social service providers (socio-health care

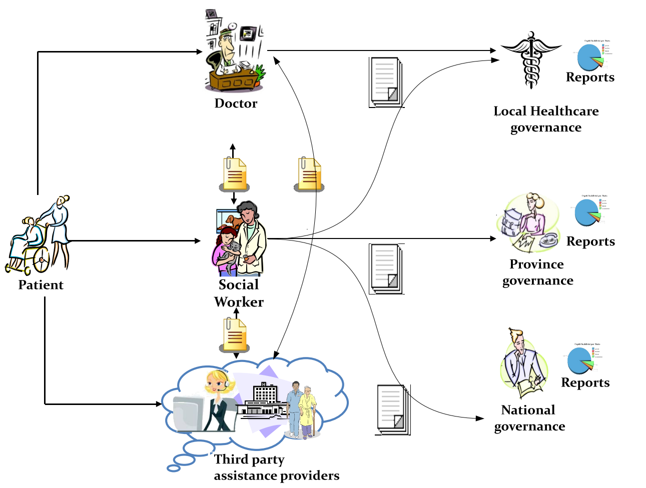


Fig. 1. Social and Health local scenario

service providers) such as doctors, social workers, or third parties service providers. The providers communicate mainly via documents or mail and, in some cases, by e-mail. Most of the times the patients themselves should bring their documents from office to office.

Typically each service provider has to provide data at different level of granularity (detailed vs aggregated data) to the governing body (province or ministry of health and finance) for accountability and reimbursement purposes. The governing body also uses the data to assess the efficiency of the services being delivered.

In this scenario is easy to have unintentional privacy breaches, as the data owners (doctors, social workers and third party assistance providers) that collect the data from the patients do not have any fine-grained control on the data they exchange or send to the governing body. Very often, either they make the data inaccessible (over-constraining approach) or they release more data than required to the others to complete their job in conflict with the principle of minimal usage [12].

Furthermore, as there is no central controller of the data access requests there is no way to trace how data is used by whom and for what purpose and to be able to answer to auditing inquiry by the privacy guarantor or the data subject herself.

3 State of the Art

In this section we will briefly analyze the state of the art related to business process interoperability, data interoperability and privacy management. We focus on the health-care field but the considerations we did are valid also in more general contexts. Service Oriented Architecture (SOA) [1] through web services are emerging as the standard for interoperability in intra-domain situations. One of the major problems of the SOA pattern is the point-to-point synchronous interaction that is established between involved actors. Event Driven Architectures (EDA) decouples service providers and consumers through asynchronous messaging and solves the point-to-point communication problem of SOA. In our project is adopted an Event Driven SOA approach [15] in which involved entities exchange the data through Web Service invocation and the Data Controller implements a Publish/Subscribe [10] event-driven model that allows many entities to subscribe to the same type of event.

One of the peculiarity of the Health domain is that it is characterized by extremely sensitive data and one of the main constrains in developing eHealth systems (EHR, PHR etc) is that the data should remain under the responsibility of the owners (i.e. producers). One possible solution, proposed by consortium like IHE - Integrating the Healthcare Enterprise [11] and applied in various eHealth projects [4] [18] [17], proposes a central registry with searching and data management functionalities over meta data that links and describes the data generated by producers. Various implementations of registries like UDDI, ebXML, XML/EDI [22] are available. The most appropriate in terms of flexibility, interoperability and adoption for the Health domain [9], that is adopted in our project is ebXML by OASIS [3].

The problem of privacy and security management in multi-domain environment where multiple heterogeneous entities are involved like in our scenario has a growing importance. The main problems are how to express and how to apply privacy constraint in a multi-domain environment. One possible approach of dealing with privacy is by defining constraints on the data at the moment the data is released to a third party [5]. Current privacy policy languages like P3P [19] and access control languages like IBM's EPAL or OASIS' XACML [2] allow to express privacy requirements in terms of the authorized purposes for the use of the data. In the SOA context various XML standards [23] that are more suitable in a distributed and inter-applications communication have been defined. The eXtensible Access Control Markup Language (XACML) [16] is a standardized security-policy language that allows an enterprise to manage the enforcement of all the elements of its security policy in all the components of its information systems.

XACML is mainly used for document-level access control that is too limited to support a fine-grained security enforcement. In some extensions like [6], XACML is used mainly to evaluate runtime requests on entire documents on an allow/deny model. In [14] is proposed a slightly different approach in which a query rewriting technique is used to provide to users with different access rights different views on data. In [8] is proposed a fine-grained solution that allows the definition and enforcement of access restrictions directly on the structure and content of the documents. For each request the system provides a specific XML response with the associated Document Type Definition (DTD). We apply the same idea but using XML schema (XSD) [21] instead of DTD because of more compliance with a Web Service based architecture. Furthermore our approach does not imply modification of Web Services interfaces and messages schema for each requests in order to provide a simpler solution for the involved entities that has heterogeneous level of ICT expertise.

However, all languages summarized before are not intuitive enough to be used by a privacy expert to express even simple constraints. They require a translation step to make the privacy constraints enforceable on the data schema. To overcome this problem we proposed a web application that provides an intuitive step-by-step policy configuration process. The definition process considers single events and it does not require any knowledge of underlying sources DB schema.

4 Event-Based Architecture

The CSS platform derives from a in-depth study of the healthcare and socio-assistive domains conducted in synergy with the Social welfare Department, the Health Care Agency, the two major municipalities and local companies providing telecare and elderly services in the Trentino region. In particular, we analyzed, together with the domain experts, the clinical and assistive processes to be monitored that involves all the partners to: identify the organizational and technological constraints of the IT systems in use, capture the business processes executed and the bits of data they produce and exchange with each other. This process-oriented analysis approach relieves us from the internal complexity of the single information sources as it focuses only on the 'visible' effects of the business processes captured by data events.

Events are the atomic pieces of information exchanged between data producers and data consumers according to event-based architectural pattern [15]. The infrastructure driving events from the sources to destinations is SOA based implemented with web services on top of an enterprise service bus that allows for event distribution across all interested parties.

A data event signals a change of state in a source system that is of interest to the other parties (e.g. the completion of a clinical exam by an hospital is of interest to the family doctor of the patient) and should be traced and notified to them. The composition of data events on the same person produced by different sources gives her social and health profile the caregivers needs to take care of her.

An event is described by two types of messages: notification and detailed message (in the rest of the paper we will use message and event interchangeably). Together they fully characterize the occurrence of an event but at a different levels of detail and sensitiveness:

- the notification message contains only the data necessary to identify a person (who), a description of the event occurred (what), the date and time of occurrence (when) and the source of the event (where). It contains the identifying information of a person but not sensitive information.
- the detail message contains all the data associated to an event generated during the assistance process (e.g. the result of a clinical trial or the report of a psychological analysis) that may be particularly sensitive and for this reason is maintained at the sources.

It is like if the profile of a person is represented by a sequence of “snapshots” (the events) and each snapshot has a short description of meta-data that explains where, when and by whom the “photo” was taken and what’s the picture about (i.e., notification message); the picture is the detail (the detail message) and you can see part of it only if the owner of the picture gives you the permission. A consumer will ask for the detail only if necessary based on the short information in the notification.

As shown in Figure 2 the central rooting node of the CSS platform is represented by the *data controller* that maintains an index of the events (*events index*, implemented according to the ebXML [3] standard) as it stores all the notification messages published by the producers and notifies them automatically to the interested parties that has previously subscribed to the corresponding classes of event. The detail messages are maintained only in the producer’s system as owner and responsible body for that sensitive information.

This dichotomy is what makes the architecture compliant to the national privacy regulations [13], [12] which prohibits the duplication of sensitive information outside the control of the data owner.

In the next section we will explain in more details how the data producer could define the privacy policies that regulates the use of the events and in particular the distribution of the notifications and the access to the details. Notifications are sent only to authorized consumers that can ask more details for specific purposes. This allows a fine-grained access control and allows the data source to hide part of the details to certain consumers depending on their functional role in the organization (e.g. social welfare department or radiology division) and purposes of use. The data controller is in charge of applying

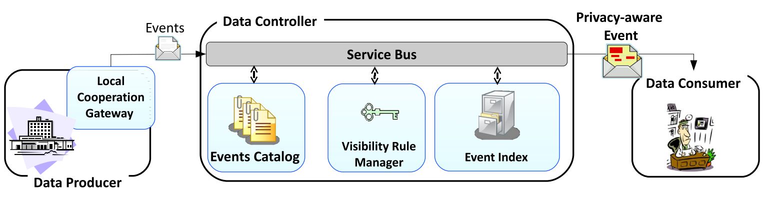


Fig. 2. General event-based architecture for the cooperation of healthcare and socio-assistive systems

the privacy policies to retrieve only what the consumer is authorized to see from the producer. It also offers the following services and functionalities:

– support both data producers and data consumers in joining the CSS platform and in particular: the data producer declares the classes of events it will generate in the event catalog and defines the privacy policies for their use by means of the visibility rule manager; the data consumer subscribes to the classes of events it is interested in;
– receive and store the notification messages and deliver them to the subscribers by means of a service bus (in the current prototype we customized the open source ESB *ServiceMix* [20]). The identifying information of the person specified in the notification is stored in encrypted form to comply with the privacy regulations;
– resolve request for details from the data consumer by enforcing the privacy policies and retrieving from the source the required and accessible information;
– resolve events index inquiry;
– maintains logs of the access request for auditing purposes;

The data controller acts as a mediator and broker between data sources and consumers and is the guarantor for the correct application of the privacy policy for retrieving the details and exploring the notifications. A data consumers can query the events index to get the list of notifications it is authorized to see without necessarily subscribe to the corresponding notification messages.

The decoupling between notification messages and detail messages is not only 'structural', as they are carried in different XML messages, but also temporal: typically a data

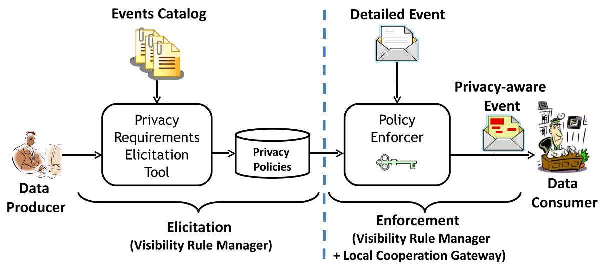


Fig. 3. Privacy Constraints elicitation and enforcement approach

consumer gets from the events index (either by automatic notification or by querying the index) an event and only at a later time it asks for the corresponding details. Requests for details may arrive to the data controller even months after the publication of the notification. This requires the data producer the capability to retrieve at any time the details associated to a past notification. These functionalities are encapsulated in the *local cooperation gateway* provided as part of the CSS platform to further facilitate the connection with the existing source systems. This module persists each detail message notified so that they can be retrieved even when the source systems are un-accessible.

5 Event-Based Privacy Constraints

The participation of an entity to the architecture (as data producer or data consumer) is conditioned to the definition of precise contractual agreements with the data controller. The contract between a data source and the data controller constraints how the data could be accessed by a third party and in particular:

- which data consumers could receive notifications (routing rules)
- how many details the data consumer could obtain from a request for details (details rules)

The data controller is not able to define such rules as it does not know which part of the event detail is really sensitive and which instead are its safe usages. On the other hand for the data source the definition of privacy rules that can be directly enforced in the system (e.g. in XACML [23]) is a complex and tedious task as it has to do it for each class of event details and requires technical expertise the typical privacy expert does not have.

To facilitate the data sources in this task we support the whole lifecycle of an event from the definition of the privacy policies (routing and detail rules) to their enforcement to resolve details requests.

In particular, we provide a GUI for the intuitive definition of the privacy policies on each class of events (Privacy Requirements Elicitation Tool) that produces policies that are directly enforceable in the system, a module that matches a detail request with the corresponding privacy policy (*Policy Enforcer*), and a module to be installed at the sources for the enforcement of the detail rules of a privacy policy in case a request is authorized (*Local Cooperation Gateway*).

The data producer declares the ability to generate a certain type of event (the *Event Details*). The structure of the event is specified by an XSD that is ‘installed’ in an *event catalog* module. The event catalog, as the structure of its events, is visible to any candidate data consumer that has previously signed a contract with the data controller to join to the cooperation architecture. In order to subscribe to a class of event (e.g. a *blood test*) or to access to its data content, the data consumer (e.g. a *family doctor*) should have the authorization by the data producer. If there is not already a privacy policy defined for that particular data consumer the data producer (that in that case could be the hospital) is notified of the pending access request and it is guided by the Privacy Requirements Elicitation Tool to define a privacy policy. Such a privacy policy defines if the *family doctor* has access to the event *blood test* and for which purpose (e.g. for healthcare treatment provisioning) and which part of the event he/she can access (e.g. the results regarding an AIDS test should be obfuscated).

It is important to note that the whole privacy policy elicitation process for the definition and storage of the privacy policies is centralized in the data controller but the resolution of a detail request is in part delegated to the source. More specifically, it is up to the data controller to find a privacy policy that matches a detail request (*policy matching phase*) and to apply the policy but it is let to the data source to return to the data controller a detailed event that contains only the data allowed by the policy (*policy enforcement*).

This approach has the following advantages:

– once the data sources have defined the privacy policies they do not need to keep track of the data consumers and data usage purposes as this is done by the data controller in charge of them. The data controller acts as guarantor and as certificated repository of the privacy policies
– it is never the case that data not accessible by a certain data consumer leaves the data producer
– the architecture is robust and flexible to changes in the representation of the privacy policies as the policy enforcement at the data controller is decoupled from the event details retrieval at the data producer
– it does not require any expertise to the data producers to define the privacy policies.

We assume that the partners are trusted parties and so we do not deal in this work with identity management. However, we plan to include as future extension of the infrastructure identity management mechanisms that are currently under development at the national level [7] for the identification of the specific users accessing the information, to validate their credentials and roles and to manage changes and revocation of authorizations in a policy. The security management mechanisms is also defined at the national level (e.g. PdD [7]) for message encryption and identity management and will be adopted when our solution will pass from a testing phase to the final release.

In the next two sections we describe more in details how the privacy policy elicitation and enforcement phases are performed.

5.1 Elicitation

As explain above the data consumer defines a privacy policy for each type of event (i.e. Event Details). We are using XACML to model internally to the Policy Enforcer module the privacy policies. According to the XACML notation [16], a *policy* is a set of *rules* with *obligations* where a rule specifies which *actions* a certain *subject* can perform on a specific *resource*. The *obligations* specifies which operations of the triggered policy should be executed at enforcing time (e.g. to obfuscate part of the resource). In our architecture an action correspond to a *purpose* of use (e.g. healthcare treatment, statistical analysis, administration).

A subject is an *actor* reflecting the particular hierarchical structure of the organization. For example, an actor could be a top level organization (e.g. '*Hospital S. Maria*') or a specific department inside it (e.g. '*Laboratory*', '*Dermatology*').

We consider an *event details* as a list of fields $e = \{f_1, \ldots f_k\}$.

Definition 1. *Let $E(D_i) = \{D_i.e_1, \ldots, D_i.e_n\}$ be the classes of event details generated by the data producer D_i for $i = 1, \ldots N$ and let $M_N(D_i.e_j)$ and $M_D(D_i.e_j)$ be respectively the notification and detailed messages that are instances of the event e_j generated by D_i.*

We define *Events Catalog*, $E = \cup_{D_1,\ldots D_N} E_{D_i}$. Basically, the event catalog contains all the types of event details that the data producers could generate. Furthermore, we call *Events Index* the set of all the notification messages generated by the data producers $D_1, \ldots, D_N$, that is $I = \cup_{i=1}^{N} M_N D_i.e_j$.

For each type of event details and type of usage the data producer D defines a privacy policy.

Definition 2. *Let $E(D) = \{D.e_1, \ldots, D.e_n\}$ be the set of events a data source D could produce. We define $P_D = \{p_1, \ldots, p_n\}$ the set of privacy policies defined by D where $p_i = \{A, e_j, S, F\}$ such that:*

- *A is an actor that can ask for an event details*
- *$e_j \in E$ is a type of event details*
- *S is a set of purposes*
- *F is a set of fields where $F \subseteq e_j$*

Intuitively, a privacy policy indicates which fields F of an event details of type e_j could be accessed by actor A, for the purposes S. For example, the privacy policy $p = \{$ *National Governance, autonomy test, statistical analysis,* $\langle age, sex, autonomy\_score \rangle\}$ allows the statistical department of the *National Governance* to access to *age*, *sex* and *autonomy_score* for the event details of type *autonomy test* to perform statistical analysis on the needs of elderly people.

We apply the *deny-by default* approach so that unless permitted by some privacy policy an Event Details cannot be accessed by any subject. With this rule semantics the obligations specify which part of the Event Details is accessible by a certain subject for some purposes. Notice also that a subject can issue only read requests for an event type.

5.2 Enforcement

Definition 3. *Given a privacy policy $p = \{A, e_j, S, F\}$ and an event request $r = \{A_r, \tau_e, S_r\}$ we say that p is a* matching policy *for r if $e_j = \tau_e \wedge A_r = A \wedge S_r \in S$.*

Intuitively a policy matches a certain request if it refers to the same type of event details and actor and if the requested purpose of usage is allowed by the policy.

Definition 4. *Given the privacy policy $p = \{A, e_j, S, F\}$ and the event instance e of type τ_e we say that e is* privacy safe *for p wrt to the request $r = \{A_r, \tau_e, S_r\}$, i.e. $e \models_r p$, if p is a matching policy for r and $\nexists f \in \tau_e$ such that $(e[f]$ is not empty $\wedge f \notin F)$ where $e[f]$ is the value of f in e.*

Intuitively an event satisfies a privacy policy if it does not expose any field that is not allowed by the policy.

If an event instance e is privacy safe wrt to a request r for all the policies in a set P we write $e \models_r P$ meaning that $e \models_r p_i, \forall p_i \in P$.

```
**Algorithm 1:** GETEVENTDETAILS($R$) $\mapsto e$
Require: $R = \{a, \tau_e, eID, s\} \neq \emptyset$
    $P$ set of policies defined by the data consumers
Ensure: $e \models_r P$
 1: $src\_eID \Leftarrow retrieveEventProducerId(eID)$
 2: $\langle A, e_j, S, F\rangle \Leftarrow matchingPolicy(R)$
 3: if $evaluate(\langle A, e_j, S, F\rangle, R) \equiv permit$ then
 4:     return $getResponse(src\_eID, F)$
 5: end if
 6: return $deny$
```

```
**Algorithm 2:** GETRESPONSE($src\_eID, F$) $\mapsto e$
Require: $src\_eID \neq NULL, F \neq \emptyset$
Ensure: $e \models_r P$
 1: $d \Leftarrow getEventDetails(src\_eID)$
 2: return $parse(d, F)$
```

Privacy policies comes into play in two distinct moments of the events life-cycle and in particular at subscription time and at access time (*request for details* and *event index inquiry*).

In order to subscribe to a class of notification events the data consumer should be authorized by the data producer that means there should be a privacy policy regulating the access to the corresponding event details for that particular data consumer. If such a privacy policy is not defined then, according with the deny by default semantics, the subscription request is rejected. The inquiry of the event index is managed in the same way as also in this case the data consumer is asking for notifications.

The request for details resolution is more articulated and we will describe it more in depth with a focus on the specific architectural components involved.

A request for details requires to specify the type and identifier of the event to be obtained from the source. This information is contained in the notification message that is a pre-requisite to issue the request for details and grant that only the data consumer notified by a data producer can access the details. The notification is obtained either automatically by means of the pub/sub service offered by the infrastructure or by direct inquiry of the event index.

Figure 4 shows the internal components of the Policy Enforcer module in the data controller which are in charge of receive the request for details from the data consumer, retrieve the matching privacy policy associated to the Event Type and Event ID specified in the request, apply and evaluate the policy against the request and finally return a response with the result of the authorization decision. The result is an event details with values only for the fields authorized by the matching policy.

The components which constitute the Policy Enforcer are based on the XACML Specification.

Algorithm 1 ($getEventDetails(R)$) shows the actions performed by the policy enforcer to resolve an authorization request R issued by a data consumer a to access to the event with identifier eID and type τ_e for purpose s. The main steps are described below:

1. The authorization request is received by the Policy Enforcement Point (PEP). Through the Policy Information Point (PIP) it retrieves the corresponding local event ID (src_eID) valid in the data producer of the event. This mapping step is necessary as the event identifier distributed in the notification messages (eID) is a global artificial identifier generated by the data controller to identify the events independently from their data producers.

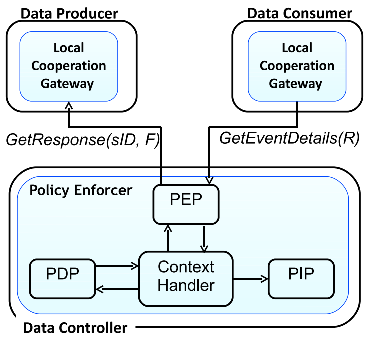


Fig. 4. Detail request resolution and privacy policy enforcement

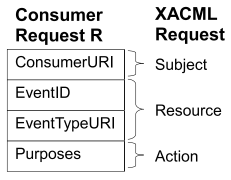


Fig. 5. Mapping in XACML request notation

2. The PEP sends the request to the Policy Development Point (PDP). The PDP retrieves the matching policy associated to the data producer, the data consumer and the resource: $\langle A, e_j, S, F \rangle$.
3. The PDP evaluates the matching policy and sends the result to the PEP. If there is no matching policy for the request or the evaluation fails the response will be *deny* and an *Access Denied message* is sent to the data consumer.
4. If the matching policy successfully evaluates the request (*permit decision*), the PEP asks the allowed part of the event details (F) to the data producer (i.e. the owner of the resource). Basically the producer applies the obligations in the policy.

Algorithm 2 shows the actions performed by the data producer in the $getResponse(src\_eID, F)$ method and in particular:

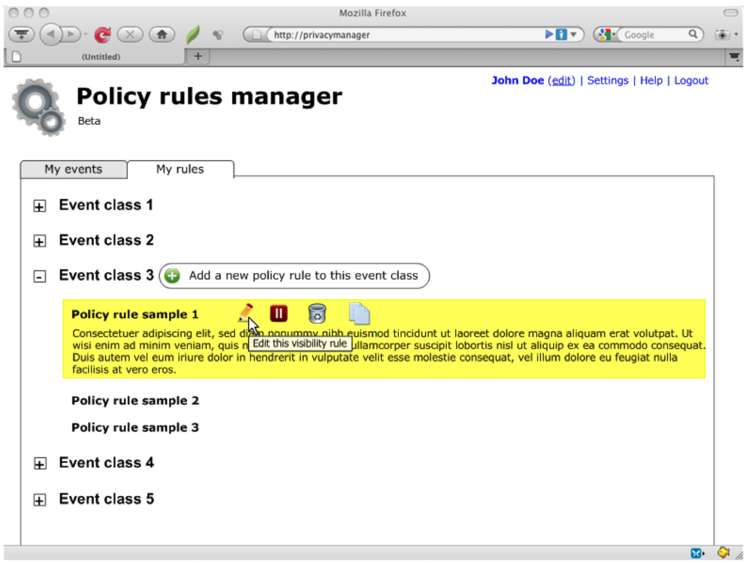
1. retrieves the Event Details from the internal events repository at the Local Cooperation Gateway;
2. parses the Event Details to filter out the values of the fields that are not allowed and produces the Privacy-Aware Event to be sent to the data consumer.

Notice that only the data accessible to the data consumer leaves the data producer. Fields that are not authorized are left empty. The data controller is a trusted entity which performs the application of policies, can trace the request of access and does the message routing between data producers and data consumers.

The architecture of the policy enforcer reflects XACML but the way we interact with the data producer and data consumer is independent from the underlying notation and enforcement strategy. As shown in Figure 5 the request for details of the data consumer is mapped to an XACML request by the policy enforcer. Similarly, the $getResponse$ method does not depend on the policy but only on the fields to be included in the event details.

6 Privacy Requirements Elicitation Tool

Figure 6 shows the Dashboard of the Privacy Rules Manager the data owner will use to define the privacy policies. She can define one or more privacy policy rule for each type



John Doe (edit) | Settings | Help | Logout

Policy rules manager
Beta

My events | My rules

Event class 1

Event class 2

Event class 3 — Add a new policy rule to this event class

Policy rule sample 1

Consectetuer adipiscing elit, sed di[illegible] nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis n[illegible] ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros.

Edit this visibility rule

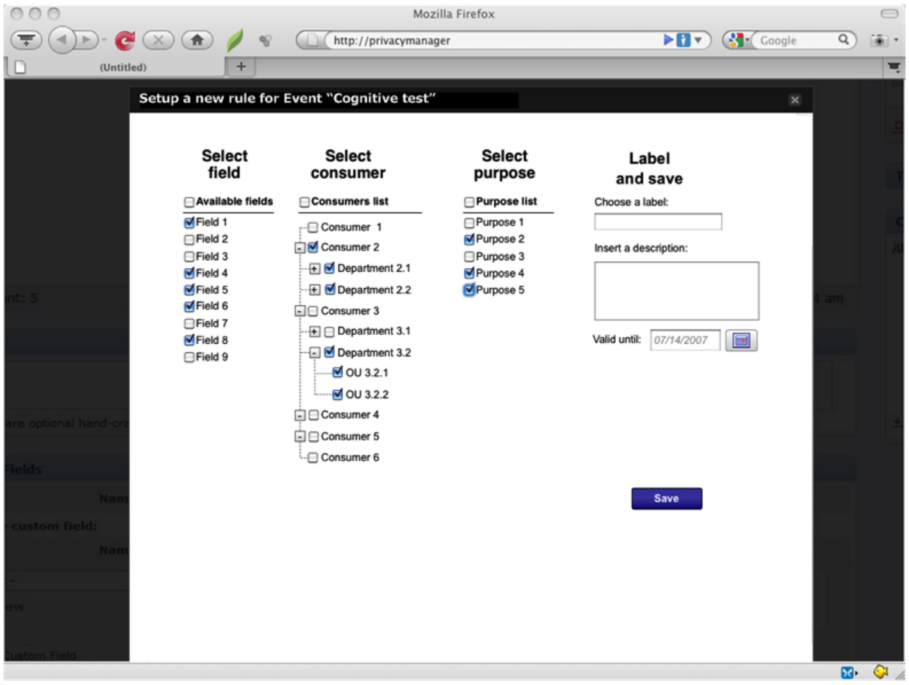
Policy rule sample 2

Policy rule sample 3

Event class 4

Event class 5

Fig. 6. Dashboard of the Privacy Rules Manager



Setup a new rule for Event "Cognitive test"

Select field	Select consumer	Select purpose	Label and save
☐ Available fields	☐ Consumers list	☐ Purpose list	Choose a label:
✓ Field 1	☐ Consumer 1	☐ Purpose 1	
☐ Field 2	✓ Consumer 2	✓ Purpose 2	Insert a description:
☐ Field 3	✓ Department 2.1	☐ Purpose 3	
✓ Field 4	✓ Department 2.2	✓ Purpose 4	
✓ Field 5	☐ Consumer 3	✓ Purpose 5	
✓ Field 6	☐ Department 3.1		Valid until: 07/14/2007
☐ Field 7	✓ Department 3.2		
✓ Field 8	✓ OU 3.2.1		
☐ Field 9	✓ OU 3.2.2		
	☐ Consumer 4		
	☐ Consumer 5		
	☐ Consumer 6		

Save

Fig. 7. Privacy Policy definition tool

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <Policy ... targetnamespace omissis>
 3     <Description> HomeCare Service Request Policy </Description>
 4     <Rule RuleId="HomeCareServiceReqRule" Effect="Permit">
 5         <Description> The Family Doctor can ask details of HomeCareService</Description>
 6         <Target><Subjects>
 7             <Subject><SubjectMatch MatchId="...string-equal">
 8                 <AttributeValue DataType="...string">FamilyDoctor</AttributeValue>
 9                 <SubjectAttributeDesignator AttributeId="...role" DataType="...string"/>
10             </SubjectMatch></Subject>
11         </Subjects>
12         <Resources>
13             <Resource><ResourceMatch MatchId="...string-equal">
14                 <AttributeValue DataType="...ssitring">urn:css:HomeCareServiceEvent</AttributeValue>
15                 <ResourceAttributeDesignator AttributeId="...resource-id" DataType="...string"/>
16             </ResourceMatch></Resource>
17         </Resources>
18         <Actions>
19             <Action><ActionMatch MatchId="...string-equal">
20                 <AttributeValue DataType="...string">HealthCareTreatment</AttributeValue>
21                 <ActionAttributeDesignator AttributeId="...action-id" DataType="...string"/>
22             </ActionMatch></Action>
23         </Actions></Target>
24     </Rule>
25     <Obligations><Obligation ObligationId="fieldsAvailable"  FulfillOn="Permit">
26             <AttributeAssignment
27                 AttributeId="...field1"
28                 DataType="...string">/md:DettaglioEvento/md:PatientID</AttributeAssignment>
29             <AttributeAssignment
30                 AttributeId="...field2"
31                 DataType="...string">/md:DettaglioEvento/md:Name</AttributeAssignment>
32             <AttributeAssignment
33                 AttributeId="...field3"
34                 DataType="...string">/md:DettaglioEvento/md:Surname</AttributeAssignment>
35         </Obligation>
36     </Obligations>
37 </Policy>
```

Fig. 8. Example of Privacy Policy

of event. Figure 7 shows the UI for the definition of an instance of privacy policy. The user can select (i) one or more items from the list of fields in the event details type, (ii) whom (i.e. ore or more OUs as consumers) and (iii) the admissible purposes. Privacy rules are saved with a name and a description. Optionally the user can specify a validity date that limits the application of the rule for a certain time window. This option is particularly useful when private companies are involved in the care process and should access to the events of their customers only for the duration of their contract.
Some of the advantages of the UI are listed below:

– it is very intuitive to use as it does not require any knowledge of XACML but it asks to the user to define a policy in terms of actor, type of event to protect and admissible purposes of use;

– it automatically generates and store in a policy repository the privacy policy in XACML format.

In Figure 8 we provide an example e of a privacy policy that allows a user with role family doctor (lines 7-10) to access the event of type HomeCareServiceEvent (lines 13-16) for HealthCareTreatment purpose (line 20). In particular, only the fields PatientId, Name and Surname of the details are accessible (line 25-36).

7 Conclusions and Future Work

In this paper we proposed and implemented a solution that respects national standards both in application cooperation and in privacy. The approach allows us to couple the benefits of a pub/sub event-based system (decoupling of publishers and subscribers) with a privacy approach that is compliant with the privacy laws typically adopted in managing healthcare information, as validated with privacy compliance experts. The system provides services for policy definition and application but does not dig into the data that travels from sources to destinations and improves the control on data exchanges and consumption by validating and logging all data requests and exchanges. The informatization of data flows permits also to avoid privacy violations caused by manual importing into systems (operators does not input anymore data into systems). The privacy requirements elicitation tool is very intuitive and can be used by privacy experts that does not have sufficient knowledge on underlying DB schema.

We already develop all the components of the infrastructure (routing, storage and privacy aware details retrieval of the events) and tested it with sample data given by the data providers. The Web Interfaces for the definition of the privacy rules are still under development to take better into account the requirements of accessibility of the partners as shown in the mockup implementation of the User Interface in Figure 6 and 7.

The system can be used also directly by the citizens to specify and control their consent on data exchanges. This possibility will acquire more importance considering that the CSS is the backbone for the implementation of a Personalized Health Records (PHR) in Trentino.

Acknowledgment

This research work comes from our experience in the project CSS founded by the Autonomous Province of Trento. A special thank to Dr. Andrea Nicolini, project manager of the ICAR[1] project.

References

[1] Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architecture and Applications. Springer, Heidelberg (2004)

[2] Anderson, A.H.: A comparison of two privacy policy languages: EPAL and XACML. In: SWS 2006: Proceedings of the 3rd ACM workshop on Secure web services, pp. 53–60. ACM, New York (2006)

[3] Breininger, K., McRae, M.: ebxml registry tc v3.0. Technical report, OASIS (2005)

[4] Canada health infoway, http://www.infoway-inforoute.ca/

[5] Chiasera, A., Casati, F., Florian, D., Velegrakis, Y.: Engineering privacy requirements in business intelligence applications. In: Jonker, W., Petković, M. (eds.) SDM 2008. LNCS, vol. 5159, pp. 219–228. Springer, Heidelberg (2008)

[6] Chou, S.-C., Huang, C.-H.: An extended xacml model to ensure secure information access for web services. J. Syst. Softw. 83(1), 77–84 (2010)

[1] http://www.progettoicar.it/

[7] CISIS. Inf-3: Sistema federato di autenticazione,
http://tinyurl.com/27yo92v
[8] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for xml documents. ACM Trans. Inf. Syst. Secur. 5(2), 169–202 (2002)
[9] Dogac, A., Laleci, G.B., Kabak, Y., Unal, S., Heard, S., Beale, T., Elkin, P.L., Najmi, F., Mattocks, C., Webber, D., Kernberg, M.: Exploiting ebxml registry semantic constructs for handling archetype metadata in healthcare informatics. Int. J. Metadata Semant. Ontologies 1(1), 21–36 (2006)
[10] Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Comput. Surv. 35(2), 114–131 (2003)
[11] IHE: IHE - integrating the healthcare enterprise xds profile,
http://www.ihe.net/profiles/
[12] Personal data protection code. Italian privacy guarantor, Legislative Decree no. 196 dated 30 June 2003 (2003)
[13] Guidelines on the electronic health record and the health file. Italian privacy guarantor, Italy's Official Journal 71 dated 26 March 2009 (2009)
[14] Luo, B., Lee, D., Lee, W.-C., Liu, P.: Qfilter: fine-grained run-time xml access control via nfa-based query rewriting. In: CIKM 2004: Proceedings of the thirteenth ACM international conference on Information and knowledge management, pp. 543–552. ACM, New York (2004)
[15] Michelson, B.M.: Event-driven architecture overview event-driven soa is just part of the eda. Patricia Seybold Group (2006)
[16] Moses, T.: Extensible access control markup language tc v2.0 (xacml). Technical report, OASIS (2005)
[17] NHS-UK: Nhs connecting for health,
http://www.connectingforhealth.nhs.uk/
[18] NICTIZ-AORTA: AORTA the dutch national infrastructure
[19] Schunter, M., Wenning, R. (eds.): The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Group Note (November 2006)
[20] ServiceMix, A.: Apache servicemix, http://servicemix.apache.org/
[21] W3C. Xmlschema (2001), http://www.w3.org/2001/XMLSchema
[22] Webber, D., Dutton, A.: Understanding ebxml, uddi, xml/edi. Technical report, XML Global Technologies Inc. (2000)
[23] Yagüe, M.: Survey on xml-based policy languages for open environments. Journal of Information Assurance and Security, 11–20 (2006)